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

NHẬP MÔN CNPM

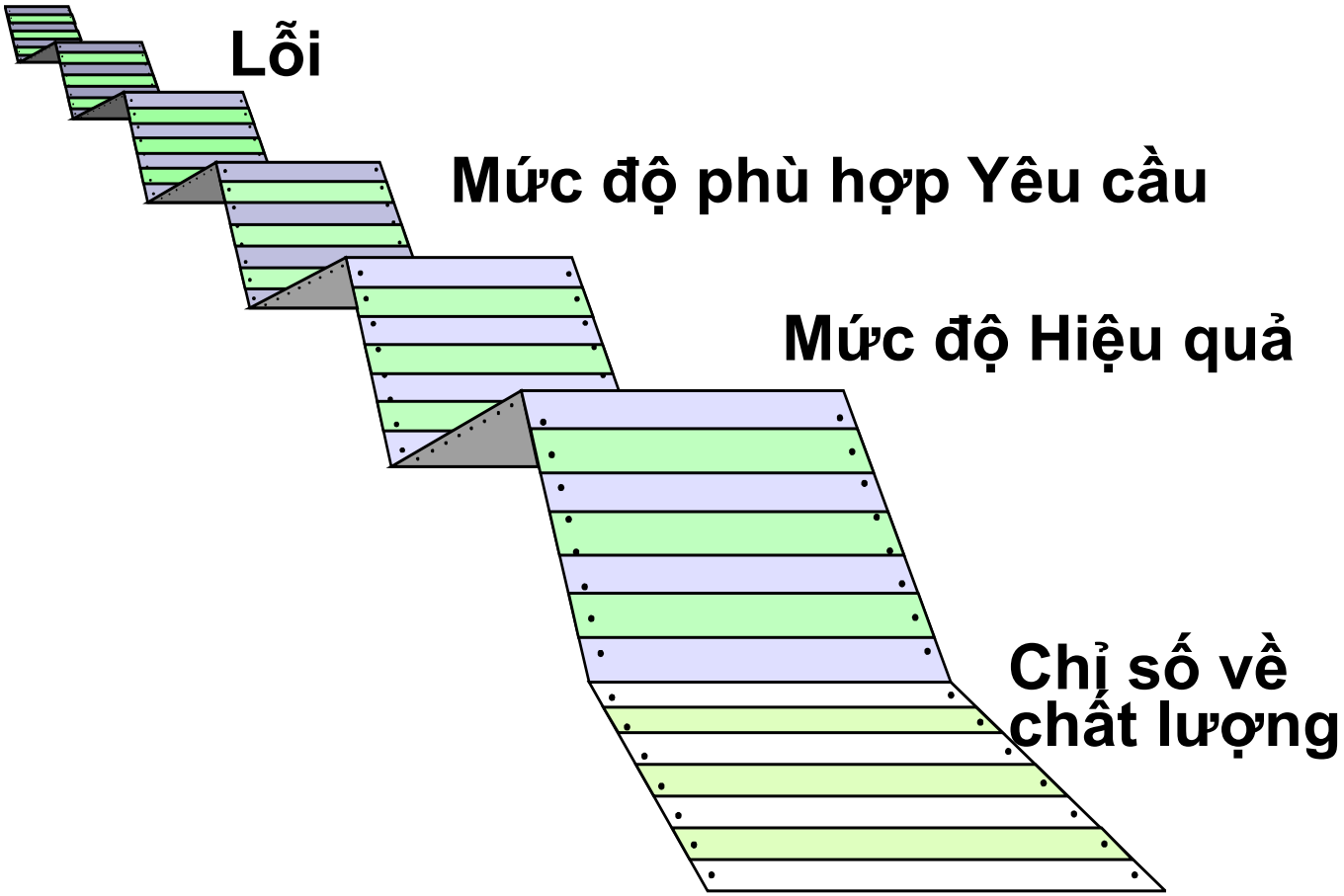
Nội dung / Chương 11-1: Giới thiệu chung về kiểm thử

Thông tin GV

Kiểm tra phần mềm

Kiểm tra là quá trình chạy một chương trình với mục đích cụ thể để tìm ra lỗi trước khi giao hàng cho người dùng cuối.

Kiểm tra cho thấy những gì



Kiểm thử chương trình

- Là mấu chốt của đảm bảo chất lượng phần mềm
- Là tiến trình (và là nghệ thuật) nhằm phát hiện lỗi bằng việc xem xét lại đặc tả, thiết kế và mã nguồn.
- Có thể chỉ ra lỗi, không thể khẳng định không còn lỗi
  - Có thể khẳng định hết lỗi bằng kiểm thử vét cạn, nhưng cách này không khả thi trên thực tế
- Một kiểm thử thành công là một kiểm thử phát hiện ra lỗi

Cách tiếp cận chiến lược

- Để thực hiện việc kiểm tra có hiệu quả, bạn nên tiến hành đánh giá kỹ thuật có hiệu quả trước. Bằng cách này, nhiều lỗi sẽ được loại bỏ trước khi kiểm tra bắt đầu.
- Kiểm tra bắt đầu ở cấp thành phần và tiến hành rộng ra đối với sự tích hợp của toàn bộ hệ thống máy tính.
- Các kỹ thuật kiểm thử khác nhau phù hợp với các phương pháp công nghệ phần mềm khác nhau và ở các thời điểm khác nhau.
- Kiểm tra được tiến hành bởi các nhà phát triển phần mềm và (đối với các dự án lớn) một nhóm kiểm tra độc lập.
- Kiểm tra và gỡ lỗi là những hoạt động khác nhau, nhưng gỡ lỗi phải được tiến hành trong bất kỳ chiến lược kiểm tra.

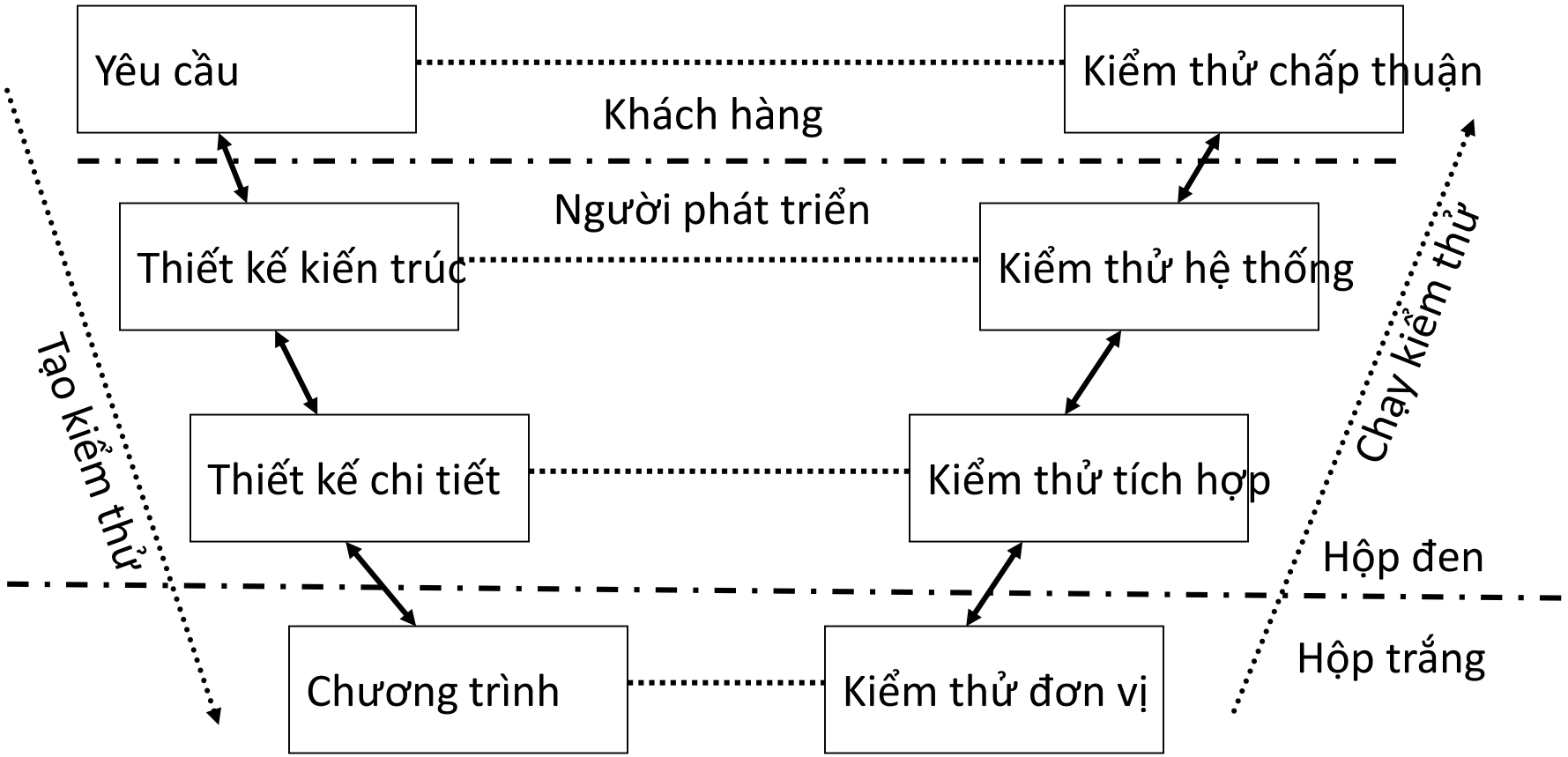
Khó khăn

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

Lưu ý khi kiểm thử

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

Kiểm thử trong mô hình chữ V



Các mức kiểm thử

- Đơn vị
  - Tìm lỗi trong từng đơn vị
- Tích hợp
  - Tìm lỗi khi ghép các đơn vị
- Hệ thống
  - Tìm lỗi khi hệ thống đã tích hợp xong, trước khi phát hành, chuyển giao
- Chấp thuận
  - Người sử dụng dùng thử xem hệ thống đáp ứng đúng mong muốn chưa.
  - Còn gọi là kiểm thử alpha.

Verification & Validation

- Kiểm định đề cập đến các các nhiệm vụ đảm bảo rằng phần mềm thực hiện chính xác một chức năng cụ thể.
- Đánh giá đề cập đến một nhiệm vụ khác đó là đảm bảo rằng phần mềm đã được xây dựng đúng theo yêu cầu của khách hàng. Boehm [Boe81] phát biểu về vấn đề này theo một cách khác:
  - Kiểm định: "Chúng ta đang xây dựng sản phẩm theo cách đúng chưa?“
  - Đánh giá : "Chúng ta đang xây dựng đúng sản phẩm chưa?"

Ai kiểm tra các phần mềm?

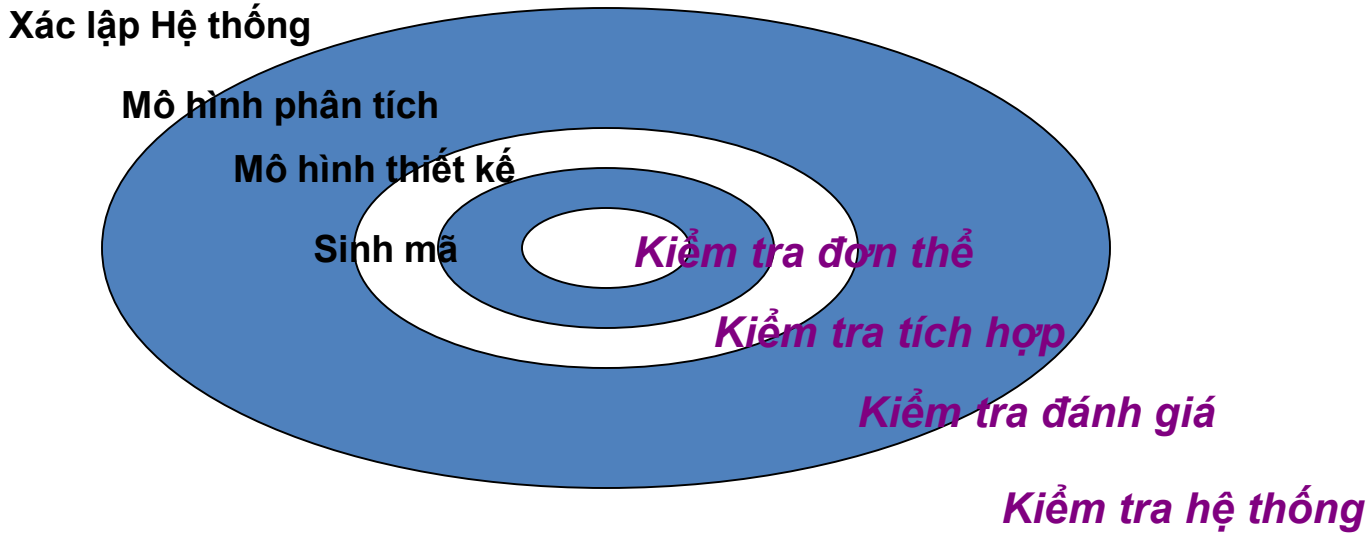
Nhà phát triển

Hiểu về hệ thống Nhưng sẽ chỉ kiểm tra tương đối và bị chi phối bởi việc giao hàng

Người kiểm tra độc lập

Phải học về hệ thống Nhưng sẽ cố gắng để hiểu sâu nó và bị chi phối bởi chất lượng

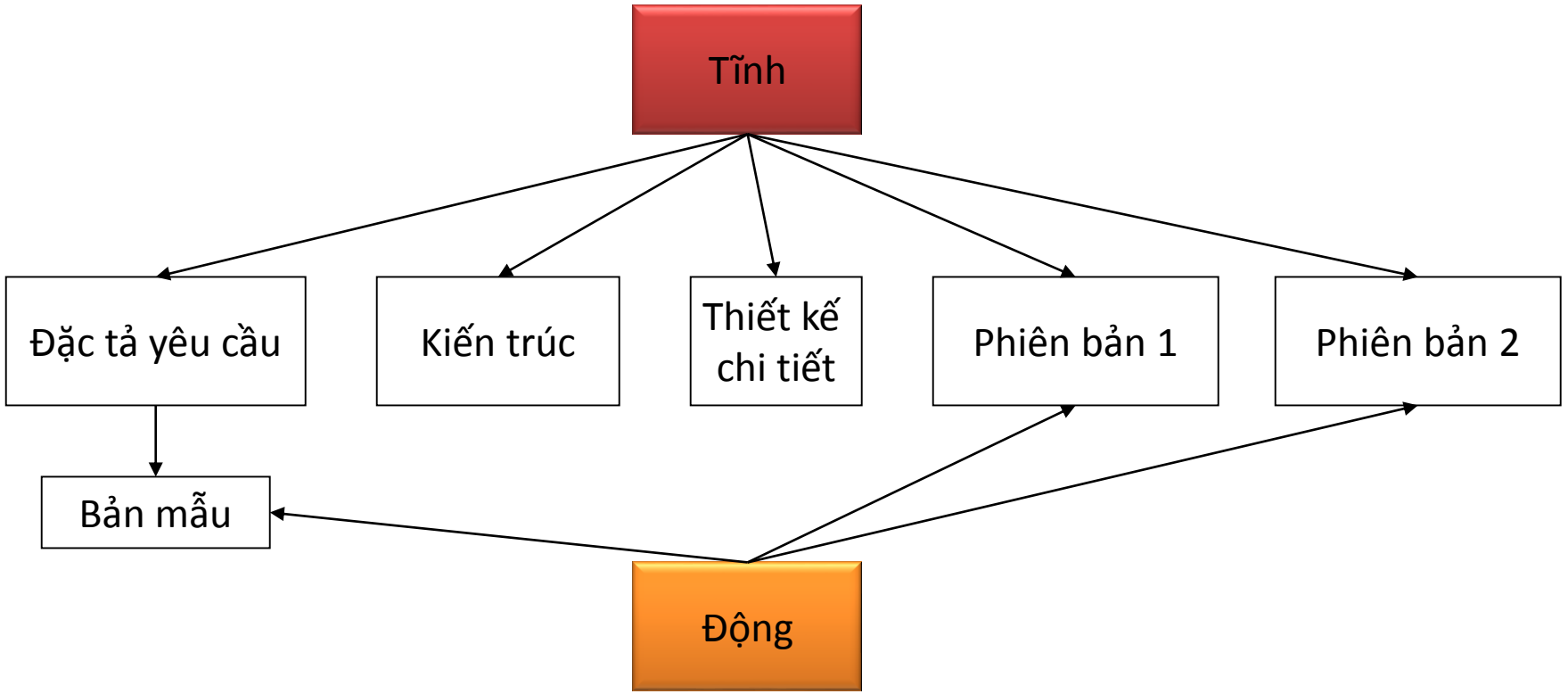
Chiến lược kiểm tra



Chiến lược kiểm tra

- Chúng ta bắt đầu với 'kiểm tra những phần nhỏ' và tiến dần tới 'kiểm tra những phần lớn'
- Đối với phần mềm truyền thống
  - Mỗi Mô-đun (Cấu phần) là quan tâm ban đầu của chúng ta
  - Sau đó thì đến việc Tích hợp các mô đun
- Đối với phần mềm OO
  - Trọng tâm khi 'kiểm tra những phần nhỏ' chuyển từ một mô đun (cách nhìn truyền thống) sang một lớp OO mà bao gồm các thuộc tính và các hoạt động và chuyển dần sang các lớp có giao tiếp và hợp tác

Kiểm thử tĩnh và động



Kiểm thử tĩnh và động

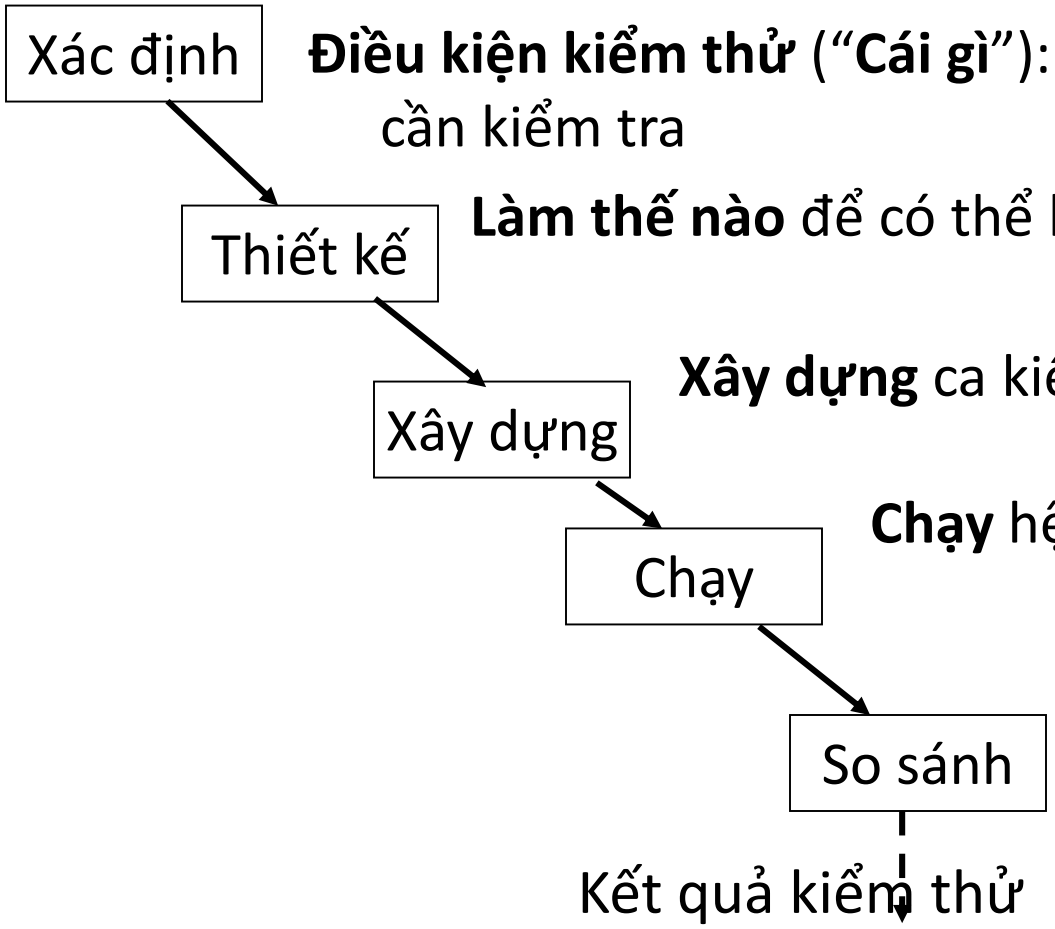
- Kiểm thử trên bàn: giấy và bút trên bàn, kiểm tra logic, lần từng chi tiết ngay sau khi lập trình xong.
- Đi xuyên suốt (walk through)
- Thanh tra (inspection)

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

Các vấn đề chiến lược

- Xác định yêu cầu sản phẩm một cách định lượng trước khi bắt đầu kiểm tra.
- Đặt mục tiêu kiểm tra một cách rõ ràng.
- Hiểu được những người sử dụng phần mềm và phát triển một hồ sơ cho từng nhóm người dùng.
- Xây dựng kế hoạch kiểm tra nhấn mạnh rằng "chu kỳ kiểm tra nhanh."
- Xây dựng phần mềm "mạnh mẽ" được thiết kế để tự kiểm tra
- Sử dụng các đánh giá kỹ thuật có hiệu quả như một bộ lọc trước khi kiểm tra
- Tiến hành đánh giá kỹ thuật để tự đánh giá các chiến lược kiểm thử và các ca kiểm thử
- Phát triển một phương pháp cải tiến liên tục cho quá trình thử nghiệm.

Các hoạt động kiểm thử



Điều kiện kiểm thử (“**Cái gì**”): một phần tử hoặc sự kiện cần kiểm tra

Làm thế nào để có thể kiểm tra được: thực hiện

Xây dựng ca kiểm thử (mã, dữ liệu)

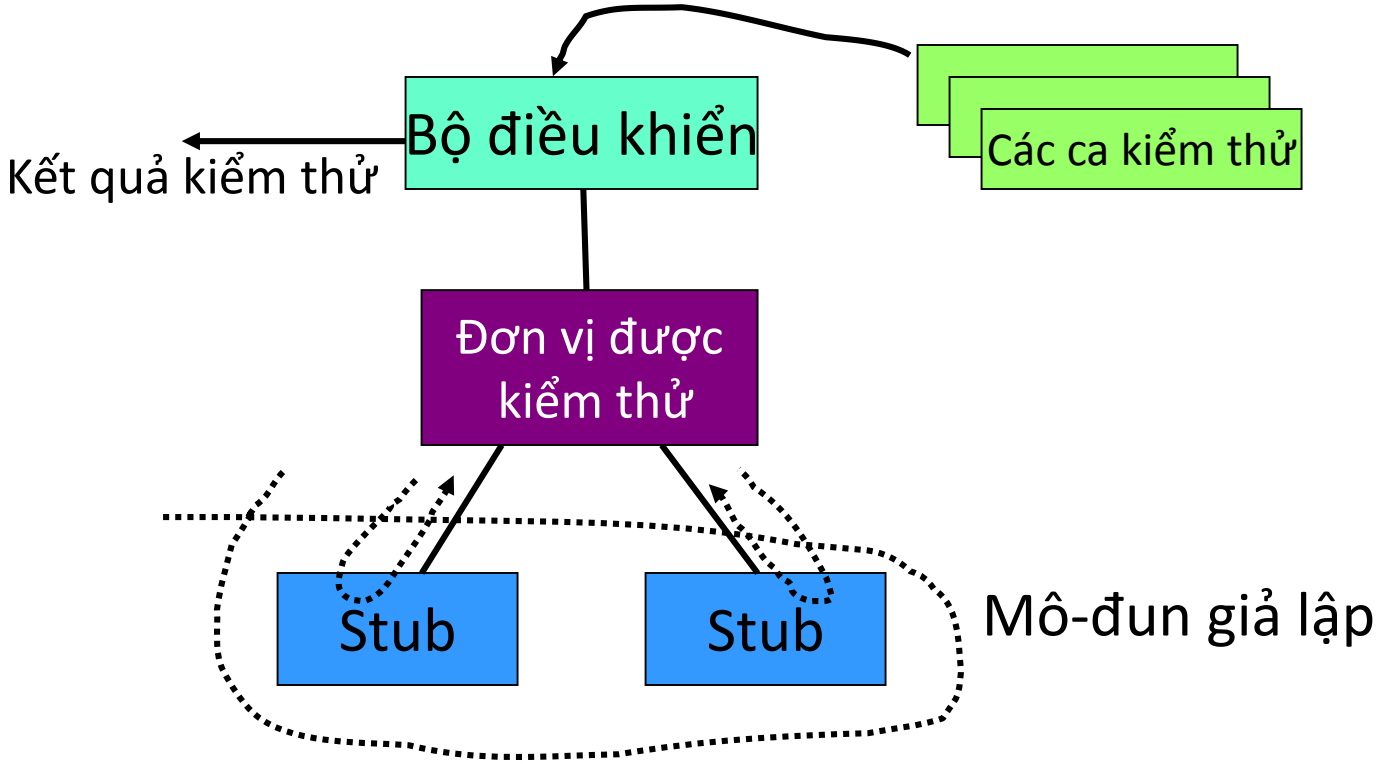
Chạy hệ thống

So sánh kết quả của hệ thống và kết quả mong đợi

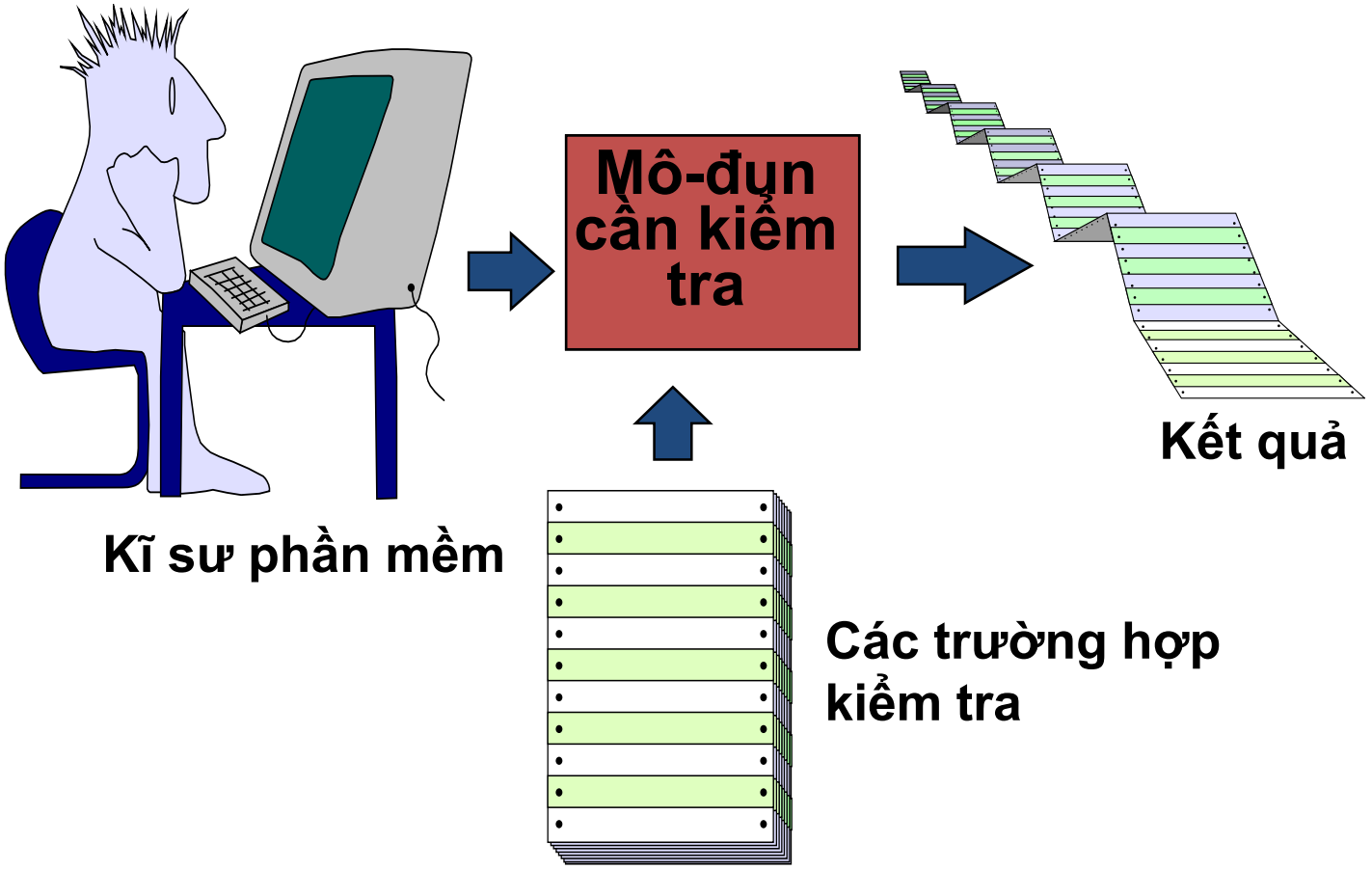
Kiểm thử đơn vị

Mục đích: Tìm sự khác biệt giữa đặc tả và cài đặt của đơn vị
Đơn vị: các lớp, hàm, đối tượng, gói, mô-đun

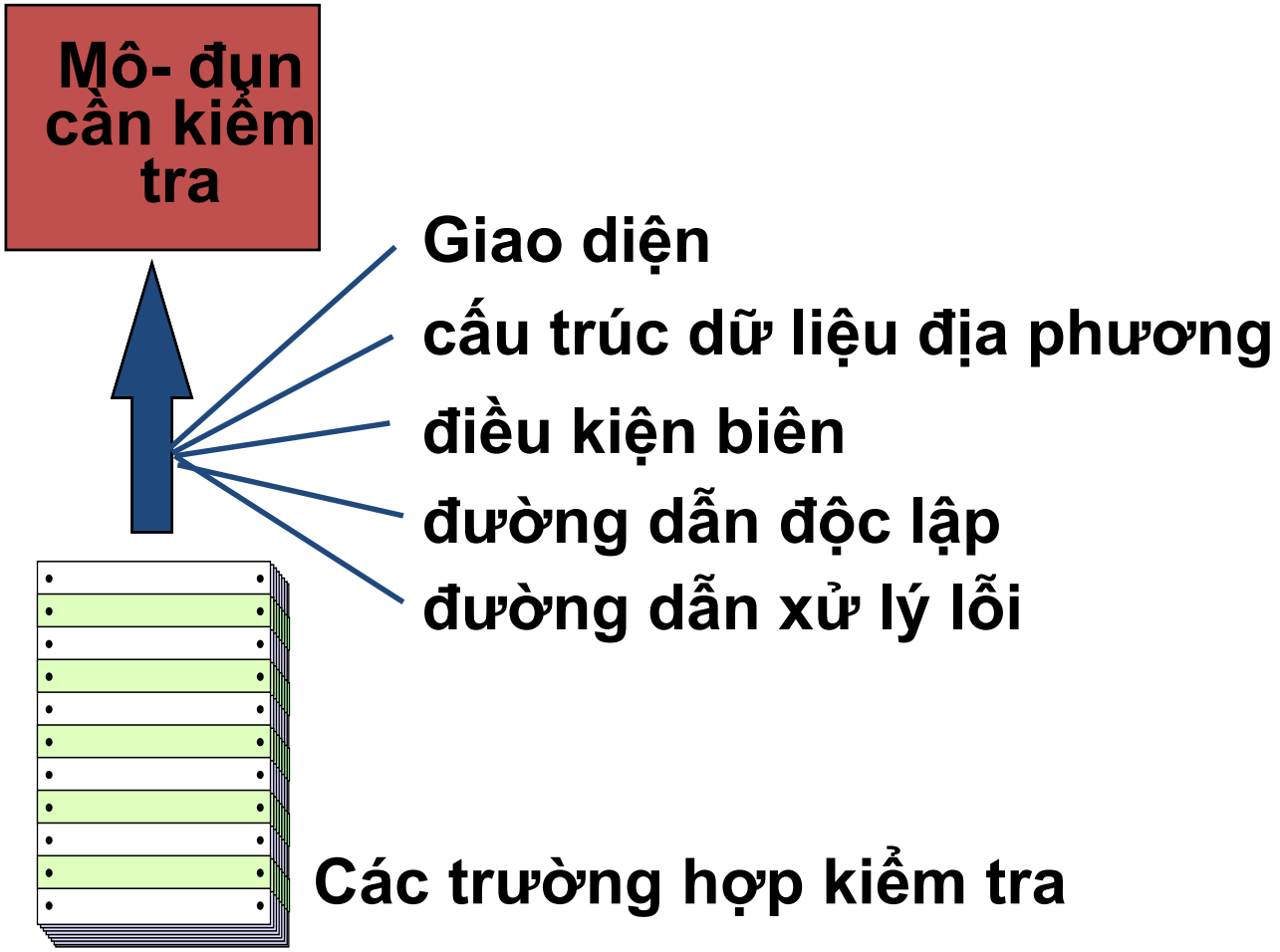
Môi trường kiểm thử đơn vị:



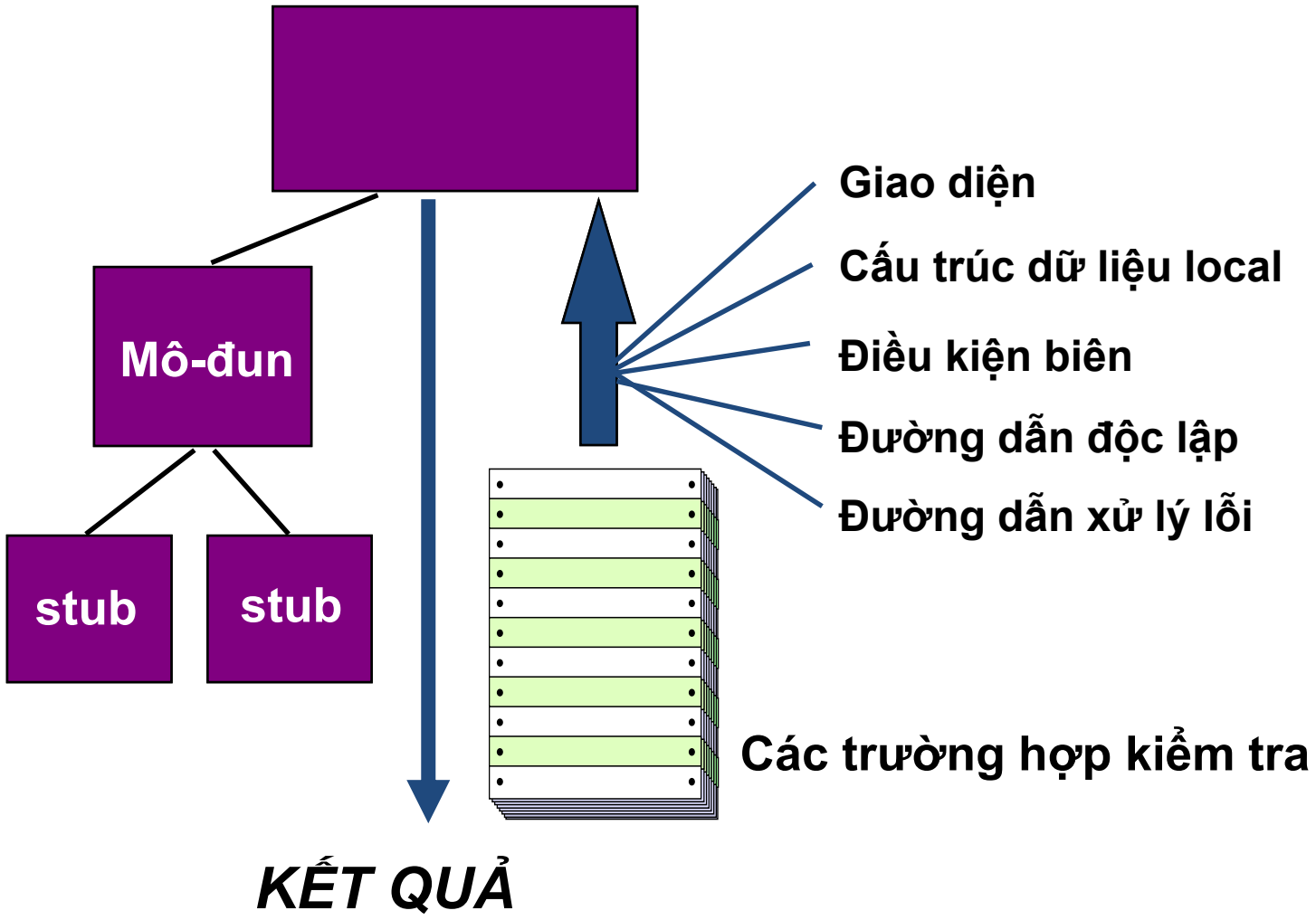
Kiểm tra đơn thể



Kiểm tra đơn thể



Môi trường kiểm tra đơn thể

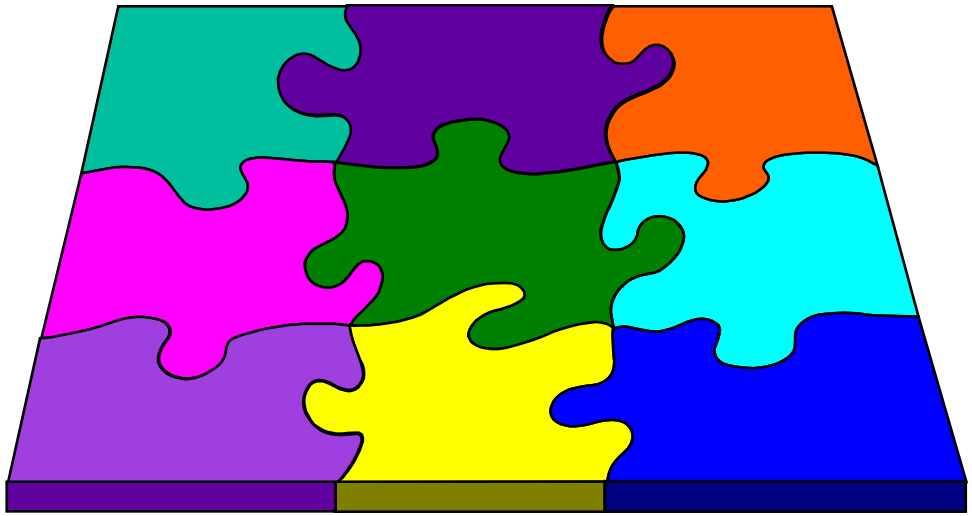


Kiểm thử tích hợp

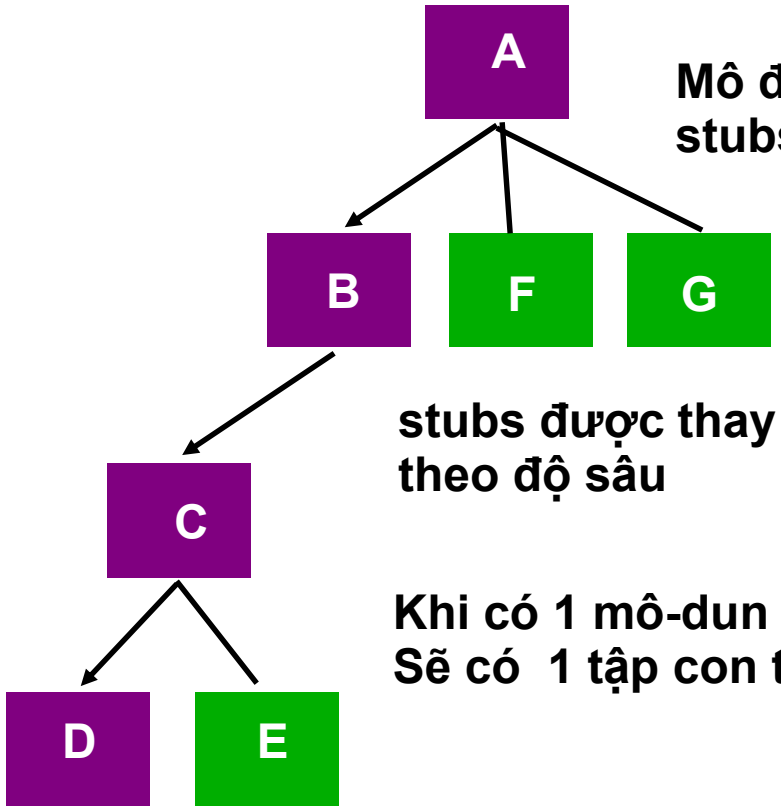
- **Mục tiêu:**
  - **Phát hiện vấn đề khi ghép các mô-đun/thành phần với nhau**
- **Các vấn đề**
  - **Bên trong: giữa các thành phần**
    - Gọi: call/message passing/…
    - Tham số: kiểu, số lượng, thứ tự, giá trị
    - Kết quả trả về: ai, kiểu, trình tự
  - **Bên ngoài:**
    - Ngắt (wrong handler?)
    - Thời gian vào ra

Chiến lược kiểm tra tích hợp

- Lựa chọn:
  - Phương pháp “big bang”
  - Chiến lược xây dựng incremental



Tích hợp từ trên xuống

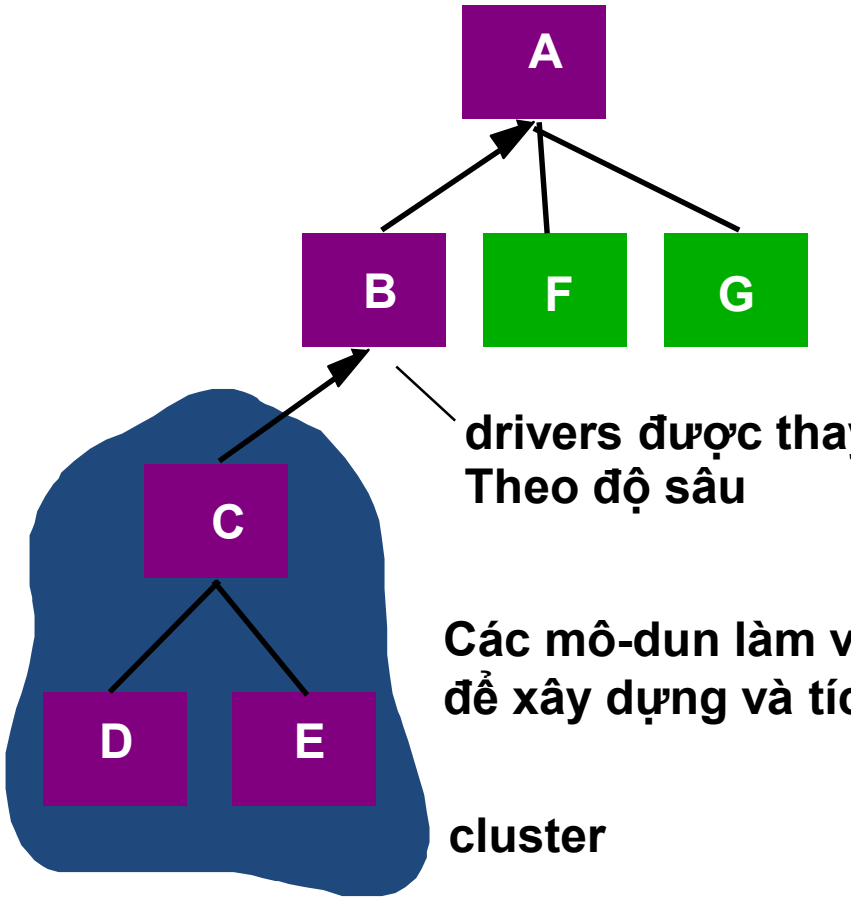


Mô đun trên cùng được kiểm tra với stubs

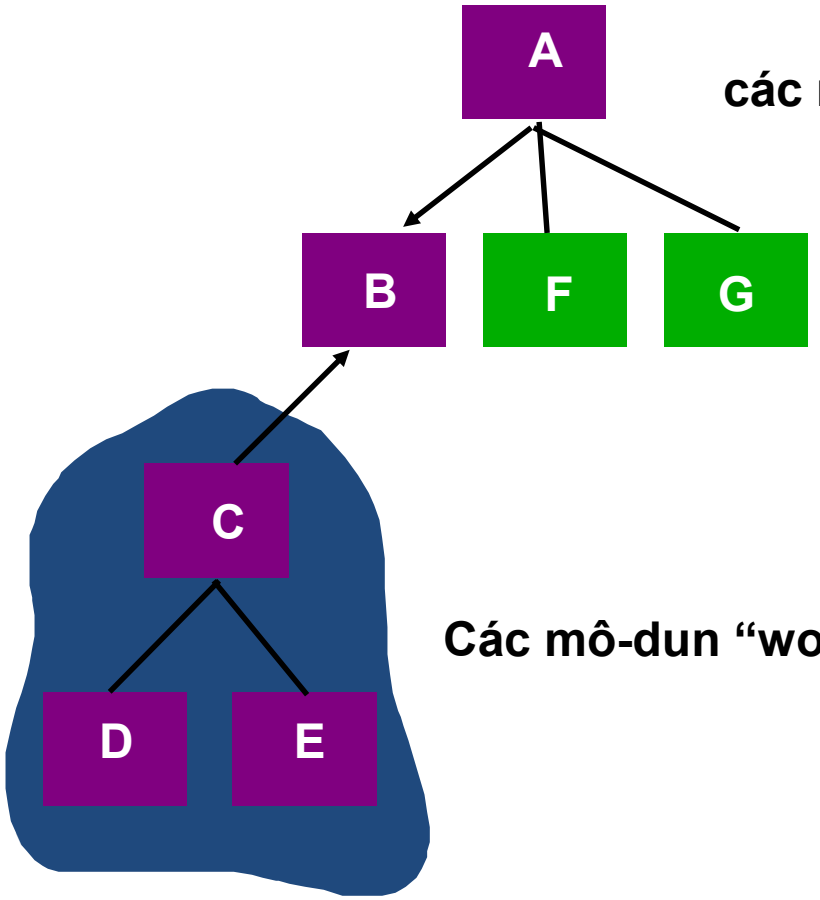
stubs được thay thế 1 lần theo độ sâu

Khi có 1 mô-dun được tích hợp,
Sẽ có 1 tập con test được chạy lại.

Tích hợp từ dưới lên



Sandwich Testing



các mô-dun trên được test

Các mô-dun "worker" được nhóm lại.

Kiểm tra hồi quy

- Kiểm tra hồi quy là việc tái thực hiện một số tập hợp con của các bài kiểm tra đã được tiến hành để đảm bảo rằng những thay đổi không phát sinh tác dụng phụ ngoài ý muốn
- Bất cứ khi nào phần mềm được hiệu chỉnh, một số khía cạnh của cấu hình phần mềm (các chương trình, tài liệu của nó, hoặc dữ liệu hỗ trợ nó) cũng được thay đổi.
- Kiểm tra hồi quy giúp đảm bảo rằng những thay đổi (do kiểm tra hoặc vì lý do khác) không xảy ra hành vi không mong muốn hoặc các lỗi bổ sung.
- Kiểm tra hồi quy có thể được tiến hành bằng tay, bằng cách tái thực hiện một tập hợp của tất cả các trường hợp thử nghiệm hoặc sử dụng các công cụ chụp / phát lại tự động.

Smoke Testing

- Một phương pháp phổ biến để tạo ra "xây dựng hàng ngày" cho sản phẩm phần mềm
- Các bước kiểm tra:
  - Các thành phần phần mềm đã được dịch sang mã được tích hợp vào một "xây dựng".
  - Một ‘xây dựng’ bao gồm tất cả tập tin dữ liệu, thư viện, mô-đun tái sử dụng, và các thành phần thiết kế được yêu cầu để thực hiện một hoặc nhiều chức năng sản phẩm.
  - Một loạt các bài kiểm tra được thiết kế để phát hiện lỗi sẽ giữ cho ‘xây dựng’ thực hiện đúng chức năng của nó.
  - Mục đích nhằm để phát hiện ra lỗi "show stopper" , lỗi có khả năng cao nhất trong việc khiến dự án phần mềm chậm tiến độ.
  - Việc xây dựng được tích hợp với các build khác và toàn bộ sản phẩm (trong hình thức hiện tại của nó) là khói kiểm tra hàng ngày.
  - Các phương pháp tiếp cận tích hợp có thể được trên xuống hoặc từ dưới lên.

Kiểm tra hướng đối tượng

- Bắt đầu bằng việc đánh giá tính đúng đắn và nhất quán của việc phân tích và thiết kế các mô hình
- Chiến lược kiểm tra làm thay đổi:
  - Khái niệm của "đơn vị" mở rộng do bao bọc
  - Tích hợp tập trung vào các lớp và sự thể hiện của chúng thông qua một "chủ đề" hay trong bối cảnh của một kịch bản sử dụng
  - Phê chuẩn sử dụng phương pháp hộp đen thông thường
- Thiết kế trường hợp kiểm tra dựa trên các phương pháp thông thường, nhưng cũng bao gồm các tính năng đặc biệt

Mở rộng cách nhìn "kiểm tra"

- Có thể lập luận rằng việc xem xét phân tích OO và mô hình thiết kế đặc biệt hữu ích vì các cấu trúc ngữ nghĩa tương tự (ví dụ: các lớp, các thuộc tính, các hoạt động, tin nhắn), xuất hiện tại các phân tích, thiết kế và cấp mã. Vì vậy, một vấn đề trong các định nghĩa của các thuộc tính lớp được phát hiện trong quá trình phân tích sẽ phá vỡ các tác dụng phụ có thể xảy ra nếu vấn đề không được phát hiện cho đến khi thiết kế hoặc mã (hoặc thậm chí các hệ tiếp theo của phân tích).

Chiến lược kiểm tra OO

- Kiểm tra lớp tương đương với kiểm tra đơn vị
  - Các hoạt động trong lớp được kiểm tra
  - Các hành vi trung ương của lớp được xem xét
- Tích hợp áp dụng 3 chiến lược khác nhau
  - Kiểm tra dựa trên chủ đề - tích hợp các tập hợp lớp được yêu cầu hồi đáp cho một đầu vào hoặc sự kiện
  - Kiểm tra dựa trên sử dụng - tích hợp tập hợp các lớp cần thiết để đáp ứng với một trường hợp sử dụng
  - Kiểm tra theo cụm - hợp nhất tập các lớp cần thiết để chứng minh có sự hợp tác giữa các thành phần

Kiểm thử hệ thống

- **Liên quan đến các yếu tố bên ngoài hệ thống**
- **Không chỉ là kiểm tra chức năng**
  - **Khả dụng (usability)**
    - Giao diện, thông báo, dễ học, dễ nhớ..
  - **Hiệu năng**
    - Khả năng đáp ứng/Tìm khả năng đáp ứng
  - **Tài nguyên sử dụng**

Kiểm thử chấp thuận

- Có hai loại kiểm thử chấp nhận
  - Bởi cơ quan phát triển gọi là BAT
  - Bởi người dùng gọi là UAT
- Mục đích: kiểm tra sự hài lòng của người sử dụng
- Cơ sở: mong muốn của người dùng (không xét đến tài liệu đặc tả)
- Môi trường: thật
- Người thực hiện: bởi và cho người sử dụng
- Các ca kiểm thử:
  - Sử dụng lại từ kiểm thử hệ thống
  - Do người dùng thiết kế

Khi nào nên dừng kiểm thử

- Hết thời gian, hết ngân sách
- Đạt mức độ bao phủ mong muốn
- Đạt tần suất hỏng hóc mong muốn

Ước lượng số lỗi

- Công thức dự đoán số lỗi của Halstead

$$((N_1+N_2)\log_2(n_1+n_2))/3000 \text{ (/KDSI)}$$

với

  - N_1: Số toán tử
  - N_2: Số toán hạng
  - n_1: Số toán tử khác nhau
  - n_2: Số toán hạng khác nhau

Ước lượng thời gian kiểm thử

- Công thức (Brettschneider) xác định thời gian kiểm thử

$$\frac{\ln\left(\frac{f_{target}}{0.5 + f_{target}}\right)}{\ln\left(\frac{0.5 + f_{target}}{f_{total} + f_{target}}\right)} \times t_h$$

với

- *ftarget* : Số lượng lỗi dự đoán
- *ftotal*: Số lỗi thực sự xảy ra sau đó
- *th*: Thời gian kiểm thử xảy ra lỗi

Ước lượng thời gian kiểm thử

- Giả sử sản phẩm có 50000 LOC, hợp đồng qui định mỗi KDSI có ít hơn 0.02 lỗi. Sản phẩm được kiểm thử 400 giờ, trong thời gian này có 20 lỗi xảy ra và đã thực thi 50 giờ kể từ lỗi cuối cùng. Xác định thời gian cần kiểm thêm?

Ước lượng thời gian kiểm thử

- Giả sử sản phẩm có 50000 LOC, hợp đồng qui định mỗi KDSI có ít hơn 0.02 lỗi. Sản phẩm được kiểm thử 400 giờ, trong thời gian này có 20 lỗi xảy ra và đã thực thi 50 giờ kể từ lỗi cuối cùng.
  - $ftarget = 0.02 \times 50 = 1$,
  - $ftotal = 20$,
  - $th = 400 - 50 = 350$ giờ

$$\frac{\ln\left(\frac{f_{target}}{0.5 + f_{target}}\right)}{\ln\left(\frac{0.5 + f_{target}}{f_{total} + f_{target}}\right)} \times t_h = (\ln(1/1.5)/\ln(1.5/21)) \times 350 = 54 \text{ giờ}$$

=> Phải kiểm thử thêm 4 giờ nữa

Kiểm tra ứng dụng web - I

- Các mô hình nội dung cho các ứng dụng web được xem xét để phát hiện ra lỗi.
- Các mô hình giao diện được xem xét để đảm bảo rằng tất cả các trường hợp sử dụng đều có thể được đáp ứng.
- Mô hình thiết kế cho các ứng dụng web được xem xét để phát hiện ra các lỗi điều hướng.
- Giao diện người dùng được kiểm tra để phát hiện ra các lỗi trong trình diễn và / hoặc cơ học điều hướng.
- Mỗi thành phần chức năng là đơn vị được kiểm tra

Kiểm tra ứng dụng web - II

- Sự điều hướng trong các kiến trúc được kiểm tra.
- Các ứng dụng web được thực hiện với một loạt các cấu hình môi trường khác nhau và được kiểm tra cho phù hợp với từng cấu hình.
- Kiểm tra an ninh được thực hiện trong một nỗ lực để ngăn chặn khai thác lỗ hổng trong các ứng dụng web hoặc trong môi trường của nó.
- Các bài kiểm tra hiệu năng được tiến hành.
- Các ứng dụng web được thử nghiệm bởi sự điều khiển và giám sát của một số lượng người dùng cuối. Các kết quả sự tương tác của họ với hệ thống được đánh giá về nội dung và lỗi điều hướng, quan tâm về sử dụng, quan tâm về tương thích, và độ tin cậy và hiệu suất của ứng dụng web.

Kiểm tra theo thứ tự

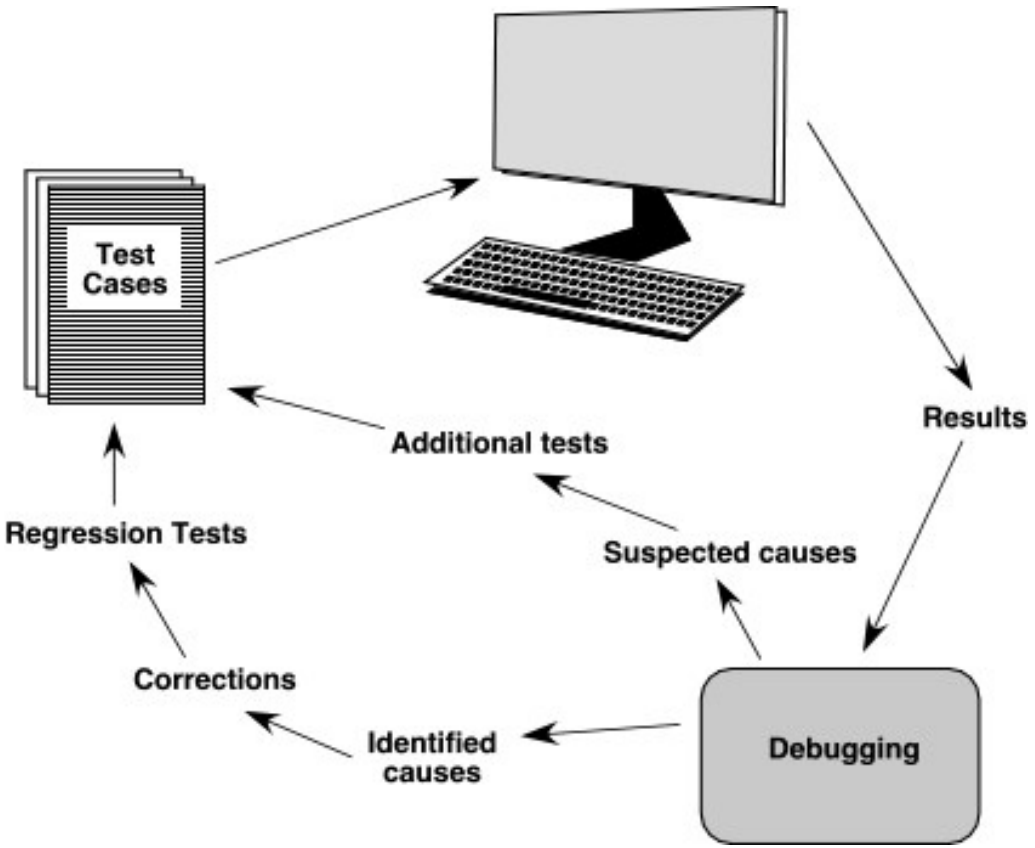
- Kiểm tra phê chuẩn
  - Tập trung vào các yêu cầu phần mềm
- Kiểm tra hệ thống
  - Tập trung vào tích hợp hệ thống
- Kiêm tra Alpha / Beta
  - Tập trung vào cách sử dụng của khách hàng
- Thử nghiệm phục hồi
  - Buộc các phần mềm sai theo nhiều cách khác nhau và xác minh rằng sự phục hồi được thực hiện đúng.
- Kiểm tra bảo mật
  - Xác nhận rằng cơ chế bảo vệ được xây dựng thành một hệ thống, trên thực tế, sẽ bảo vệ nó khỏi sự xâm nhập trái phép
- Thử nghiệm áp lực
  - Thực hiện một hệ thống mà đòi hỏi nguồn lực bất thường về số lượng, tần số, hoặc khối lượng
- Kiểm tra năng suất
  - Kiểm tra việc thực hiện thời gian chạy của phần mềm trong bối cảnh của một hệ thống tích hợp

Nhiều công cụ hỗ trợ các loại kiểm thử

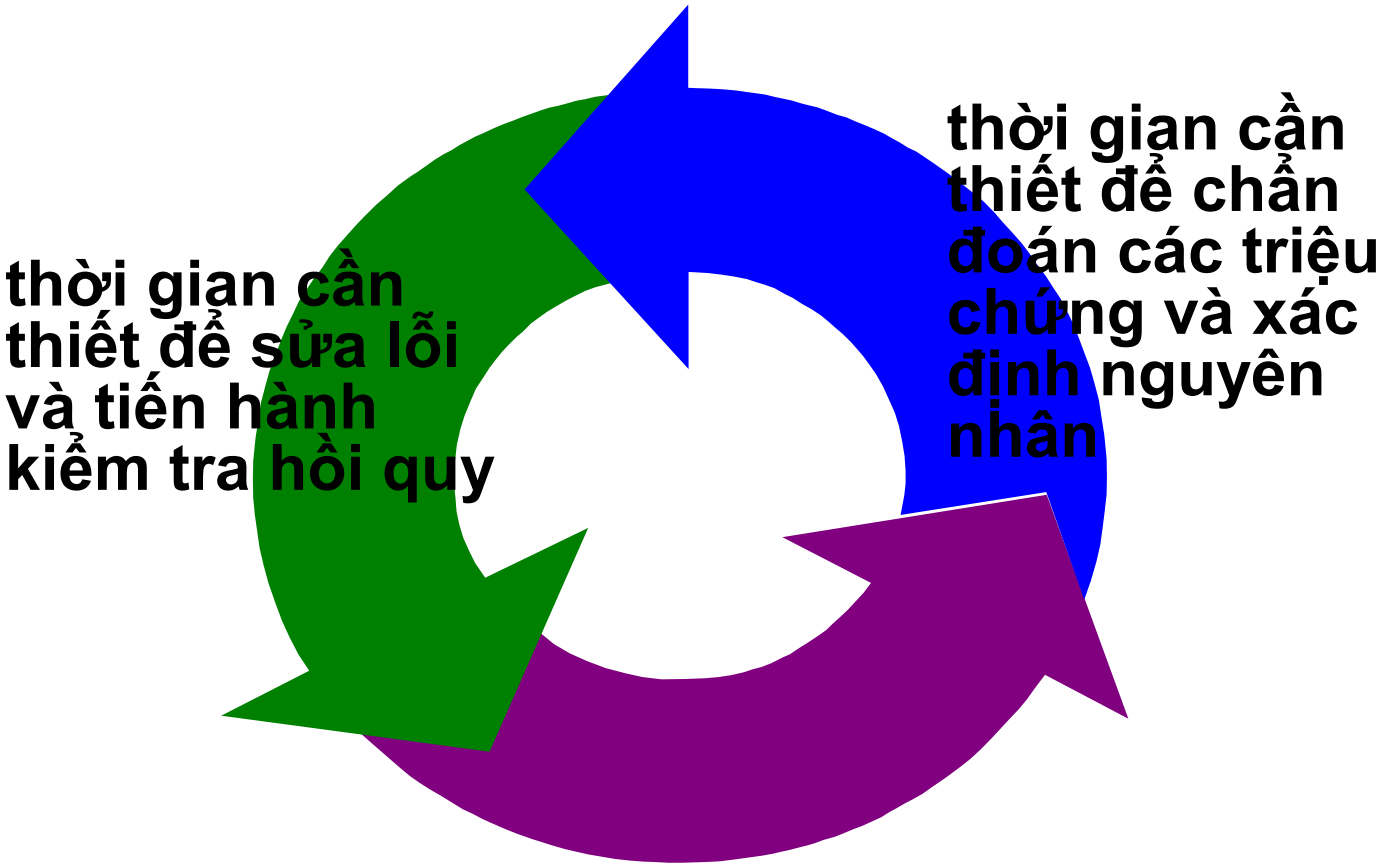
- Kiểm thử đơn vị: Achoo, JUnit, Pex/Moles, PyUnit
- Tự động kiểm thử: TestComplete, Selenium,…
- Kiểm thử hiệu năng và tải: JMeter
- Kiểm thử giao diện đồ họa (GUI): Abbot, Guitar
- Kiểm thử tổ hợp: AETG, FireEye
- Kiểm thử dựa trên mô hình: Spec Explorer
- Phân tích bao phủ: Corbertura
- Quản lý lỗi (defects): Bugzilla

Một quy trình chẩn đoán

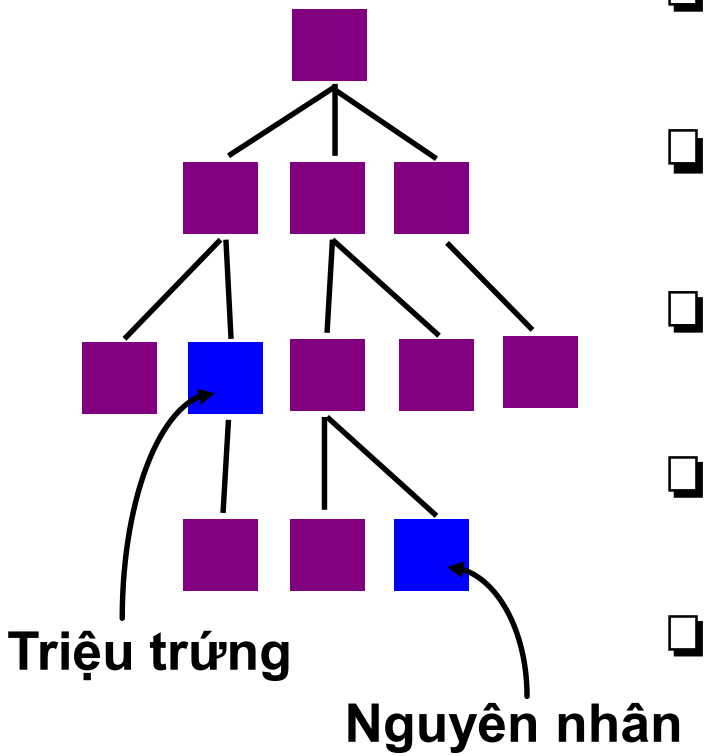
Quy trình gỡ lỗi



Debugging Effort

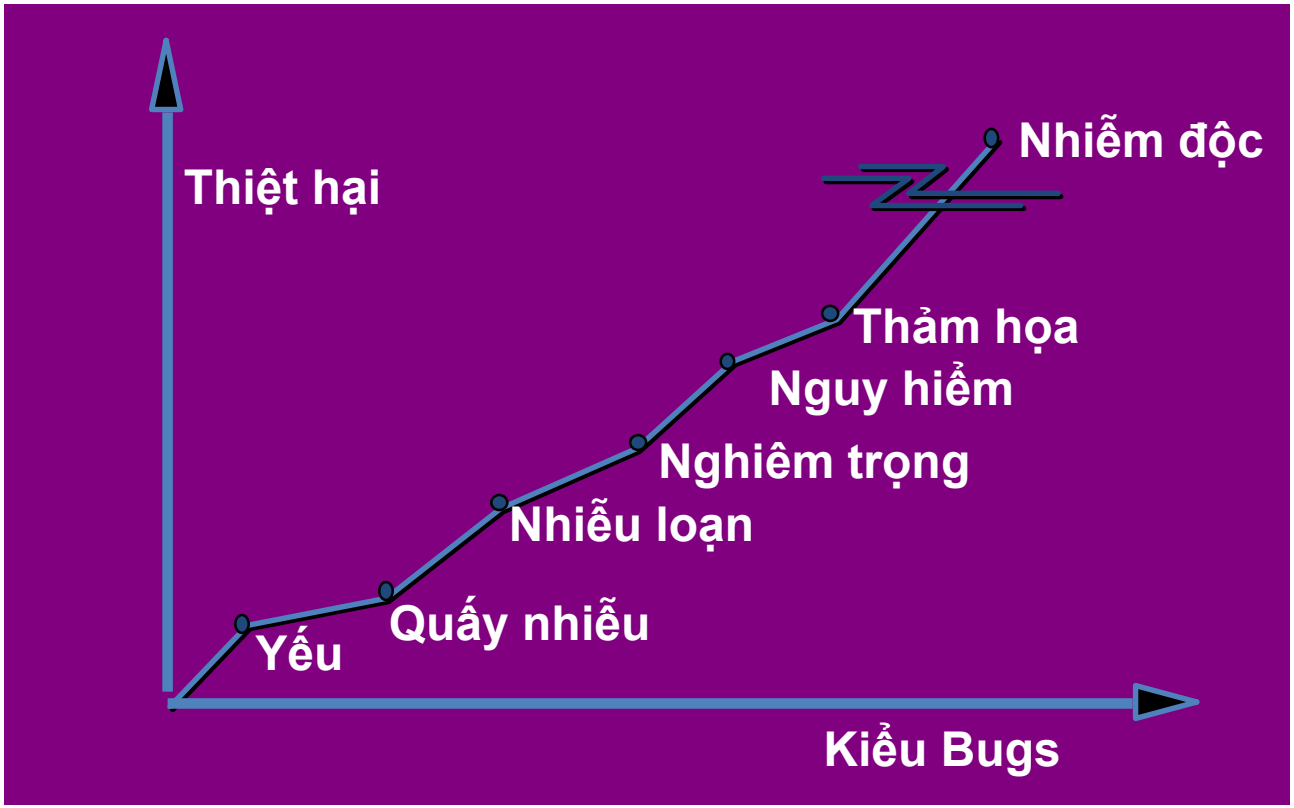


Triệu trứng và nguyên nhân



- Triệu chứng và nguyên nhân có thể tách biệt về vị trí
- Triệu chứng có thể biến mất khi một vấn đề khác được giải quyết
- Nguyên nhân có thể là do sự kết hợp của nhiều vấn đề không có lỗi
- Nguyên nhân có thể là do lỗi hệ thống hoặc lỗi biên dịch
- Nguyên nhân có thể là do giả định rằng tất cả mọi người tin tưởng
- Triệu chứng có thể xảy ra liên tục

Hậu quả của Bugs



Các loại Bug: Bug liên quan đến chức năng, bug liên quan đến hệ thống, bug dữ liệu, mã lỗi, bug thiết kế, bug tài liệu, vi phạm tiêu chuẩn, vv

Kỹ thuật gỡ lỗi

- brute force / Kiểm tra
- Tìm ngược
- Quy nạp
- Suy luận

Hiệu chỉnh lỗi

- Có phải nguyên nhân của lỗi tái phát sinh trong một phần khác của chương trình? Trong nhiều tình huống, một sai sót của chương trình bị gây ra bởi một mô hình lo gic sai lầm có thể bị sao chép ở nơi khác.
- "Lỗi tiếp theo" là gì có thể được giới thiệu bởi các sửa chữa Tôi thực hiện? Trước khi việc hiệu chỉnh được thực hiện, các mã nguồn (hoặc, tốt hơn, thiết kế) nên được đánh giá để khớp nối logic và cấu trúc dữ liệu.
- Điều gì chúng tôi đã thực hiện để ngăn chặn lỗi này ở nơi đầu tiên? Câu hỏi này là bước đầu tiên hướng tới việc thiết lập một phần mềm thống kê chất lượng đảm bảo cách tiếp cận. Nếu bạn sửa các quy trình cũng như các sản phẩm, các lỗi sẽ được gỡ bỏ từ các chương trình hiện tại và có thể được loại bỏ khỏi tất cả các chương trình

Tư tưởng cuối cùng

- Suy nghĩ – trước khi bạn thực hiện hiệu chỉnh
- Sử dụng các công cụ để có được cái nhìn sâu sắc
- Nếu bạn đang ở một tình trạng bế tắc, tìm kiếm sự giúp đỡ từ người khác
- Một khi bạn sửa lỗi, sử dụng thử nghiệm hồi quy để phát hiện ra bất kỳ tác dụng phụ

Tài liệu tham khảo

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