Các mức độ kiểm thử

Trường ĐHCN - ĐHQGHN

Nội dung

- Kiểm thử đơn vị
- Kiểm thử tích hợp
- Kiểm thử hệ thống
- Kiểm thử chấp nhận

Các mức kiểm thử

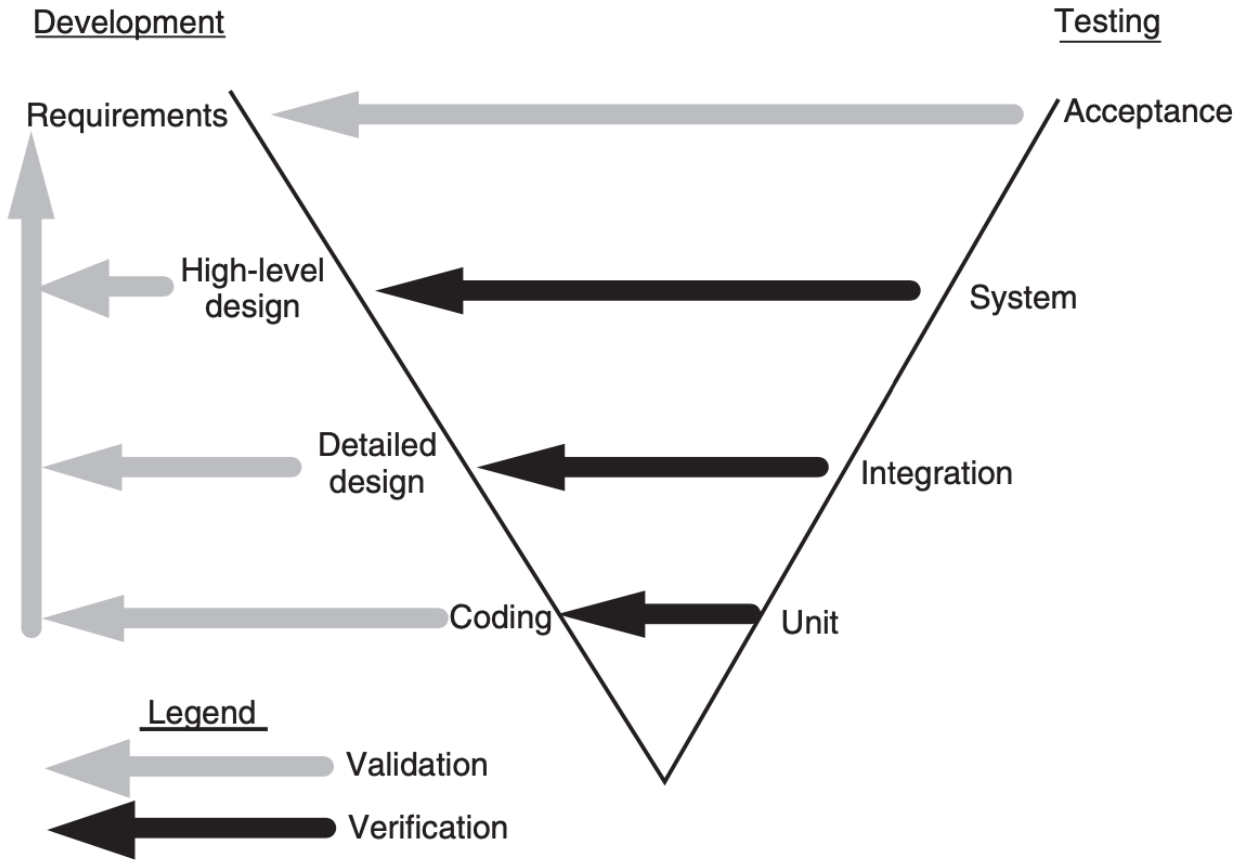


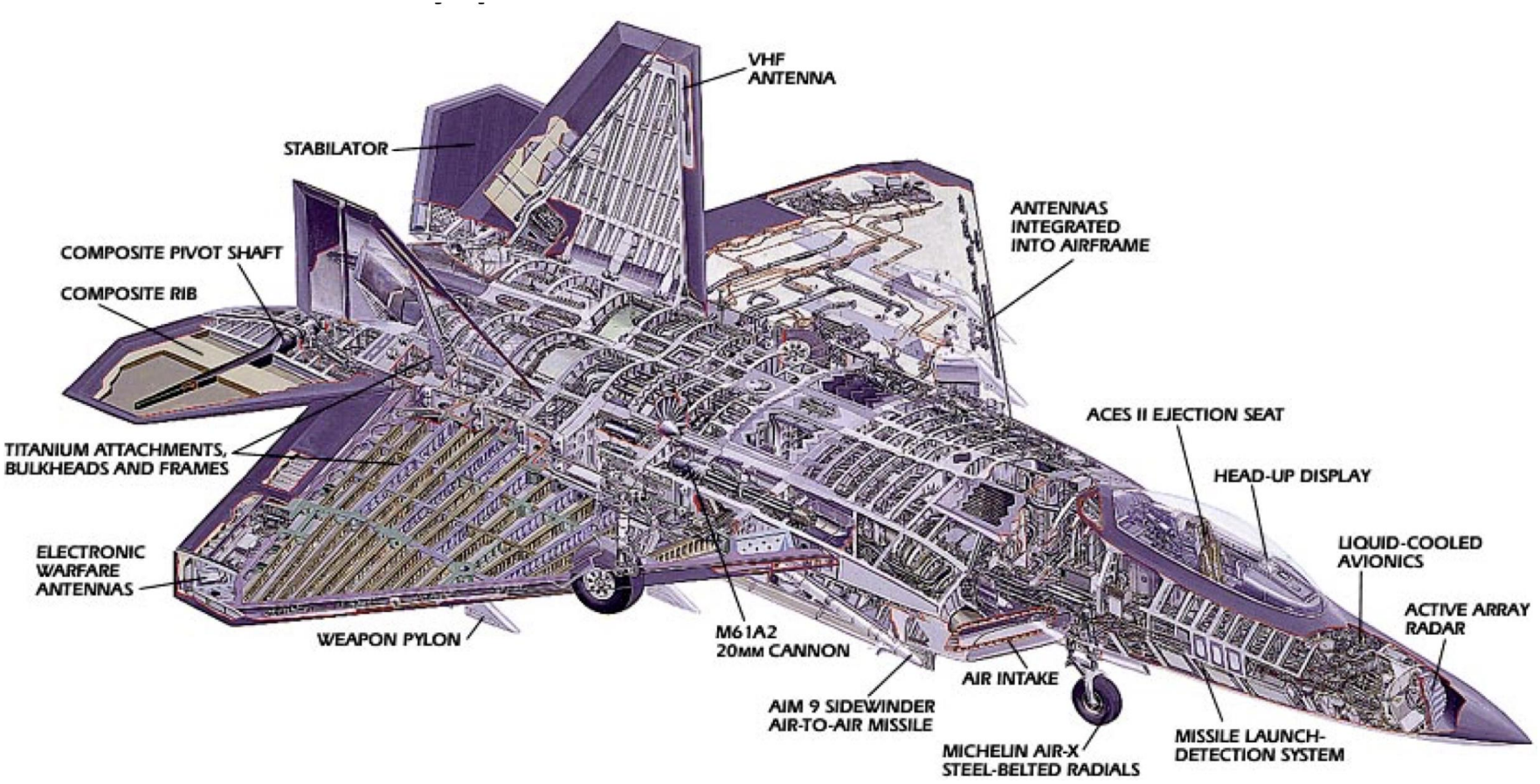
Figure 1.7 Development and testing phases in the V model.

Trang 16, sách Software Testing and Quality assurance

Kiểm thử đơn vị

Unit testing

F-22 Raptor Fighter



F-22 Rapter Fighter

- Được sản xuất bởi Lockheed Martin và Boeing
- Chuyện gì sẽ xảy ra nếu Lockheed lắp ráp 1 chiếc F-22 bởi các thành phần không được kiểm thử?
  - Có thể nó sẽ không hoạt động và cũng có thể bạn không bao giờ có thể làm cho nó hoạt động.

Khái niệm

- Kiểm thử đơn vị: Kiểm thử các đơn vị của chương trình một cách độc lập
  - Đơn vị (program unit): Hàm (functions), thủ tục (procedures), phương thức (methods), hoặc thậm chí class
  - Mỗi đơn vị của chương trình cần được kiểm thử cẩn thận trước khi tích hợp với các đơn vị khác
  - Thường được thực hiện bởi lập trình viên

Vì sao cần kiểm thử đơn vị?

- Dễ dàng sửa các lỗi được tìm thấy trong quá trình kiểm thử
- Dễ dàng (hơn) để thực thi và kiểm thử các đường thực thi khác nhau (distinct execution path) của đơn vị

Lưu ý

- Không phải tất cả các đơn vị đều có thể kiểm thử độc lập

- Một số lỗi chỉ có thể được tìm ra khi một đơn vị được kết hợp với một/một vài đơn vị khác

→ Mặc dù, không thể phát hiện được tất cả các lỗi trong từng đơn vị, việc kiểm thử và đảm bảo rằng từng đơn vị hoạt động “tốt” ở một mức độ nhất định vẫn rất cần thiết.

Lựa chọn dữ liệu kiểm thử

- Kiểm thử hộp đen (black box testing)
- Kiểm thử hộp trắng (white box testing)
- Kiểm thử hộp xám (grey box testing)

Kiểm thử tích hợp

Integration testing

Giới thiệu

- Kiểm thử tích hợp là kiểm thử các hành vi của các đơn vị chương trình khi chúng được tích hợp với nhau

- Mục đích: Đảm bảo các đơn vị chương trình hoạt động "tốt" sau khi được tích hợp với đơn vị khác

Vì sao cần kiểm thử tích hợp?

- Một trong các lỗi phổ biến giữa các đơn vị chương trình đó là: Interface error
  - Interface giữa 2 đơn vị là cơ chế cho phép 1 đơn vị truy cập dịch vụ/giao tiếp với đơn vị còn lại
  - Các đơn vị chương trình thường được phát triển bởi các lập trình viên khác nhau
- Sau khi thực thi unit testing, vẫn rất khó để có thể dự đoán được hành vi của một đơn vị khi kết hợp với các đơn vị khác

Hướng tiếp cận của kiểm thử tích hợp

- Big-bang
- Incremental
- Sandwich

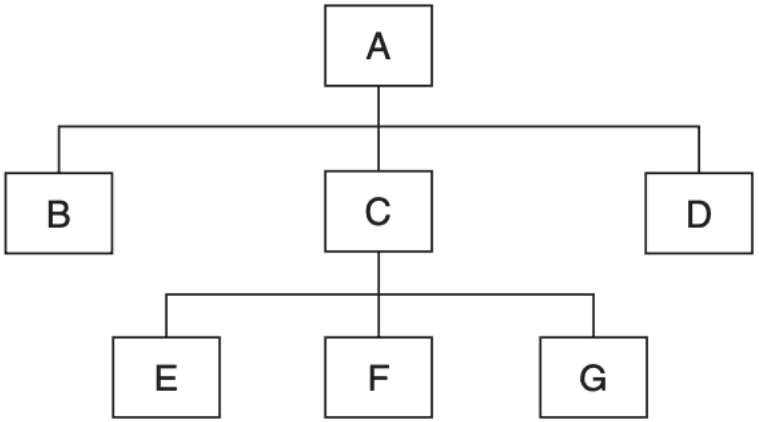
Big bang

- Big bang
  - Đầu tiên, tất cả các đơn vị được kiểm thử độc lập
  - Sau đó, tất cả các đơn vị được tích hợp thành một hệ thống hoàn chỉnh và kiểm thử toàn bộ hệ thống
  - Trong một vài trường hợp, Big-bang được áp dụng để tích hợp và kiểm thử hệ thống nhỏ.
  - Tuy nhiên, hướng tiếp cận này không phù hợp với các hệ thống lớn

Incremental

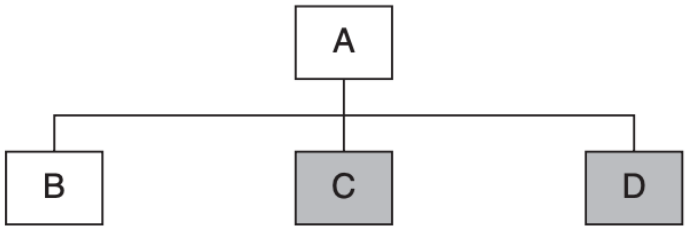
- Incremental
  - Kiểm thử tích hợp được thực hiện theo các chu kỳ tăng dần
  - Chu kỳ sau bổ sung tích hợp các đơn vị mới vào kiểm thử cùng các đơn vị đã được kiểm thử tích hợp ở chu kỳ trước
- Bao gồm 2 hướng chính:
  - Top-down
    - Sử dụng stubs
  - Bottom-up
    - Sử dụng driver

Incremental: Top-down

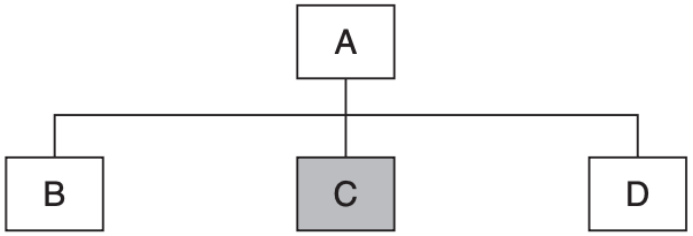


Ví dụ: Hệ thống 3 lớp (levels) với 7 modules

Incremental: Top-down

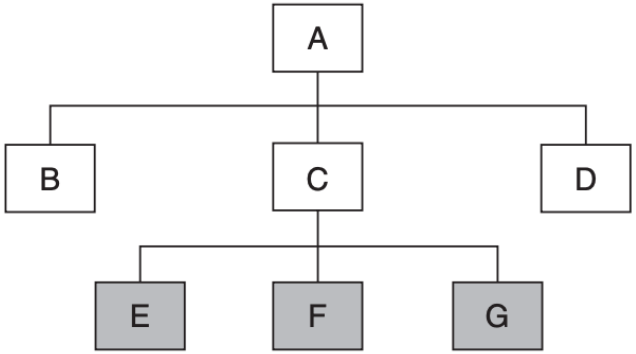


Đầu tiên: A và B được tích hợp, sử dụng stubs C và D.

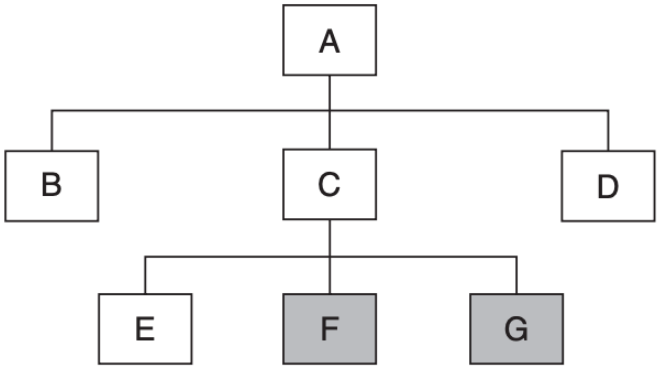


Tiếp theo, stub D được thay thế bằng module D thực.
Có hai loại kiểm thử cần thực thi, kiểm thử giao diện giữa A và D; và kiểm thử hồi quy để đảm bảo sự xuất hiện của D không gây ra bất kỳ bất thường nào cho modules A và B

Incremental: Top-down

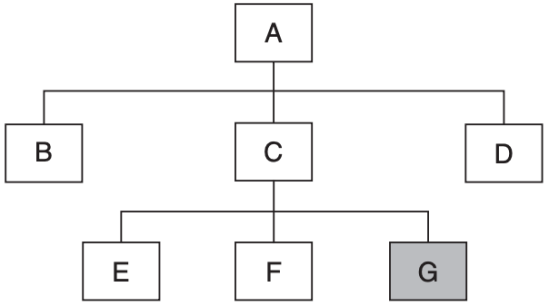


Stub C được thay thế bằng module C thực, các stubs E, F, G được bổ sung để có thể kiểm thử được module C.

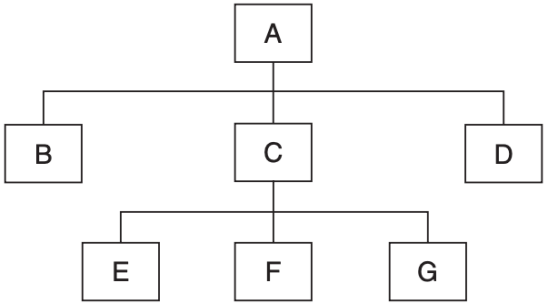


Stub E được thay thế bằng module E thực

Incremental: Top-down

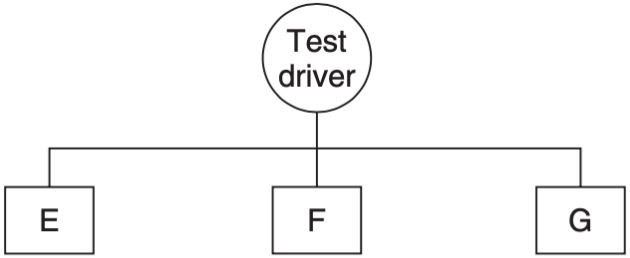


Stub F được thay thế bằng module F thực

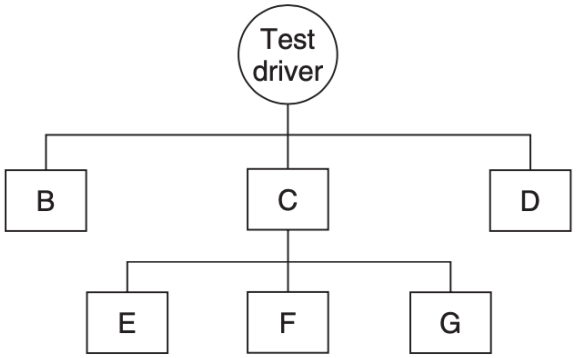


Stub G được thay thế bằng module G thực

Incremental: Bottom-up

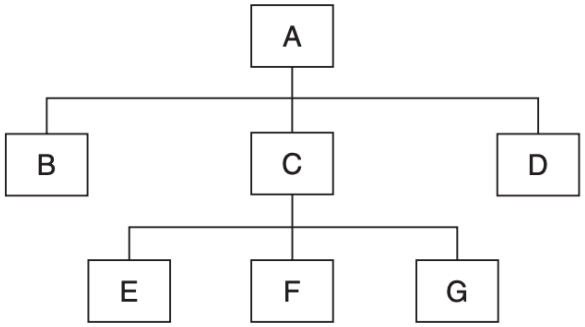


Tầng thấp nhất bao gồm các modules E, F, G.
Cần thiết kế 1 test driver để tích hợp 3 modules này (mô phỏng đơn giản hoạt động của module C).



Test driver được thay thế bằng module C thực.
Một test driver khác được thiết kế để tích hợp B, C, và D.

Incremental: Bottom-up



Test driver được thay thế bằng module A thực

Sandwich

- Sandwich
  - Các đơn vị được tích hợp theo cách kết hợp cả 2 hướng tiếp cận top-down và bottom-up
  - Cấu trúc hệ thống được chia thành 3 lớp:
    - Lớp bottom bao gồm các đơn vị thường được gọi (invoked) → tích hợp và kiểm thử theo hướng tiếp cận bottom-up
    - Lớp top bao gồm các đơn vị lớn/chính → tích hợp và kiểm thử theo hướng top-down
    - Lớp middle bao gồm các đơn vị còn lại (trong một vài trường hợp, không có lớp middle) → tích hợp và kiểm thử theo hướng big-bang sau khi lớp top và lớp bottom được tích hợp và kiểm thử

Kiểm thử hệ thống

System testing

Kiểm thử hệ thống

As a rule, software systems do not work well until they have been used, and have failed repeatedly, in real applications.

— *Dave Parnas*

Kiểm thử hệ thống

- Mục tiêu của kiểm thử hệ thống là kiểm tra sự phù hợp của (toàn bộ) hệ thống đã phát triển so với yêu cầu của khách hàng.

- Hệ thống được kiểm thử trong môi trường thực tế, với sự kết hợp đầy đủ của phần cứng và phần mềm.

Kiểm thử hệ thống

- **Basic tests**: Kiểm tra hệ thống có thể cài đặt, cầu hình, và đưa về trạng thái hoạt động.
- **Functionality tests**: Kiểm tra toàn bộ các chức năng của hệ thống theo yêu cầu trong đặc tả
- **Robustness tests**: Kiểm tra khả năng "phục hồi" của hệ thống với các đầu vào lỗi và các tình huống "hỏng hóc" (failure suitations) khác nhau.
- **Interoperability tests**: Kiểm tra khả năng tương tác của hệ thống với sản phẩm của bên thứ ba.

Kiểm thử hệ thống

- **Performance tests**: Đo hiệu suất của hệ thống ví dụ thông lượng (throughput), thời gian phản hồi … trong các điều kiện khác nhau
- **Scalability tests**: Kiểm tra giới hạn mở rộng của hệ thống về mặt số người dùng, tài nguyên, quy mô địa lý, …
- **Stress tests**: Xác định các hạn chế của hệ thống và các hành vi của hệ thống khi có lỗi xảy ra
- **Load and stability tests**: Kiểm tra để đảm bảo rằng hệ thống hoạt động ổn định trong thời gian dài dưới điều kiện full load.

Kiểm thử hệ thống

- **Reliability tests**: Kiểm tra độ tin cậy của hệ thống, hoạt động trong thời gian dài mà không phát sinh lỗi
- **Documentation tests**: Đảm bảo rằng tài liệu hướng dẫn sử dụng hệ thống được viết đầy đủ, dễ hiểu, và dễ sử dụng
- **Regulatory tests**: Kiểm tra để đảm bảo rằng hệ thống đáp ứng các quy định của chính phủ nơi mà nó sẽ được triển khai

Kiểm thử chấp nhận

Acceptance testing

Kiểm thử chấp nhập

- Khách hàng thực hiện kiểm thử chấp nhận hệ thống dựa vào mong muốn của họ về sản phẩm

- Hai loại kiểm thử chấp nhận:
  - User acceptance testing (UAT)
  - Business acceptance testing (BAT)

Kiểm thử chấp nhận

- Mục tiêu của UAT: Đảm bảo hệ thống đáp ứng đúng mong muốn của khách hàng như đã thoả thuận trong hợp đồng
- UAT thường được thực hiện bởi một đơn vị thứ ba
- BAT được thực hiện bởi đơn vị phát triển như một bước diễn tập (rehearsal) để đảm bảo hệ thống sẽ pass UAT

Một số các tiêu chí của kiểm thử chấp nhận

- Functional correctness and completeness
- Accuracy
- Data Integrity
- Data conversion
- Backup and recovery
- Competitive edge
- Usability
- Performance
- Start-up time
- ….

Bài tập

Tìm hiểu ưu điểm và nhược điểm của kiểm thử tích hợp theo phương pháp top-down và bottom-up.