Chương 8

Kiểm thử module (đơn vị)

8.1 Giới thiệu

Kiểm thử module (hay kiểm thử đơn vị) là quá trình kiểm thử từng chương trình con, từng thủ tục nhỏ trong chương trình. Một số động cơ của việc kiểm thử đơn vị :

- Kiểm thử đơn vị là 1 cách quản lý nhiều phần tử cần kiểm thử, bắt đầu tập trung chú ý trên từng phần tử nhỏ của chương trình.
- Kiểm thử đơn vị giúp dễ dàng việc debug chương trình.
- Kiểm thử đơn vị tạo cơ hội tốt nhất cho ta thực hiện kiểm thử đồng thời bởi nhiều người.

Mục đích của kiểm thử đơn vị : so sánh chức năng thực tế của từng module với đặc tả chức năng hay đặc tả interface của module đó. Sự so sánh này có tính chất :

1. Không chỉ ra việc module có thoả mãn đầy đủ đặc tả chức năng ?
2. Mà chỉ ra việc module có làm điều khác biệt gì so với đặc tả của module.

8.2 Thiết kế testcase

Hai tài nguyên thiết yếu sau sẽ cần thiết cho việc thiết kế các testcase :

- Đặc tả chức năng module : nêu rõ các thông số đầu vào, đầu ra và các chức năng cụ thể chi tiết của module.
- Mã nguồn của module.

Tính chất các testcase là dựa chủ yếu vào kỹ thuật kiểm thử hợp trắng :

- Khi kiểm thử phần tử ngày càng lớn hơn thì kỹ thuật kiểm thử hộp trắng ít khả thi hơn.
- Việc kiểm thử sau đó thường hướng đến việc tìm ra các kiểu lỗi (lỗi phân tích, lỗi nắm bắt yêu cầu phần mềm).

Thủ tục thiết kế testcase

Phân tích luận lý của module dựa vào 1 trong các kỹ thuật kiểm thử hộp trắng.

Áp dụng các kỹ thuật kiểm thử hộp đen vào đặc tả của module để bổ sung thêm các testcase khác.

8.3 Kiểm thử không tăng tiến

Để thực hiện qui trình kiểm thử các module, hãy để ý 2 điểm chính :

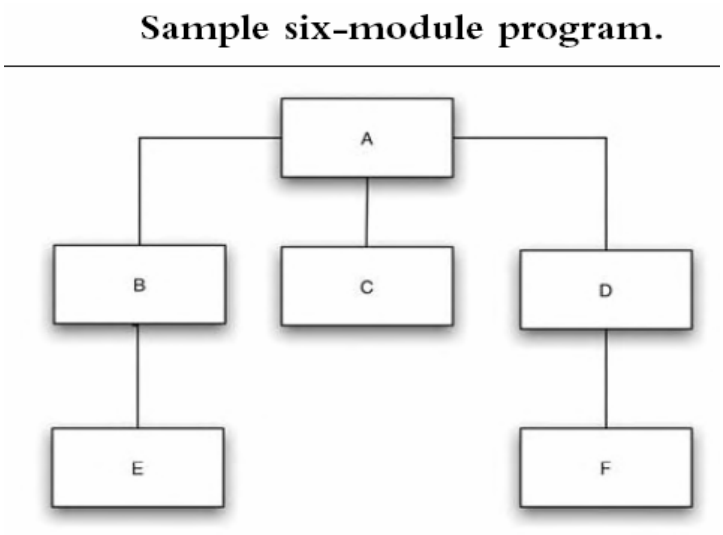
1. Làm sao thiết kế được 1 tập các testcase hiệu quả.
2. Cách thức và thứ tự tích hợp các module lại để tạo ra phần mềm chức năng :
    - Viết testcase cho module nào ?
    - Dùng loại tiện ích nào cho kiểm thử ?
    - Coding và kiểm thử các module theo thứ tự nào ?
    - Chi phí tạo ra các testcase ?
    - Chi phí debug để tìm và sửa lỗi ?

Có 2 phương án để kiểm thử các module :

1. Kiểm thử không tăng tiến hay kiểm thử độc lập (Non-incremental testing) : kiểm thử các module chức năng độc lập nhau, sau đó kết hợp chúng lại để tạo ra chương trình.
2. Kiểm thử tăng tiến (Incremental testing) : kết hợp module cần kiểm thử vào bộ phận đã kiểm thử (lúc đầu là null) để kiểm thử module cần kiểm thử trong ngữ cảnh.

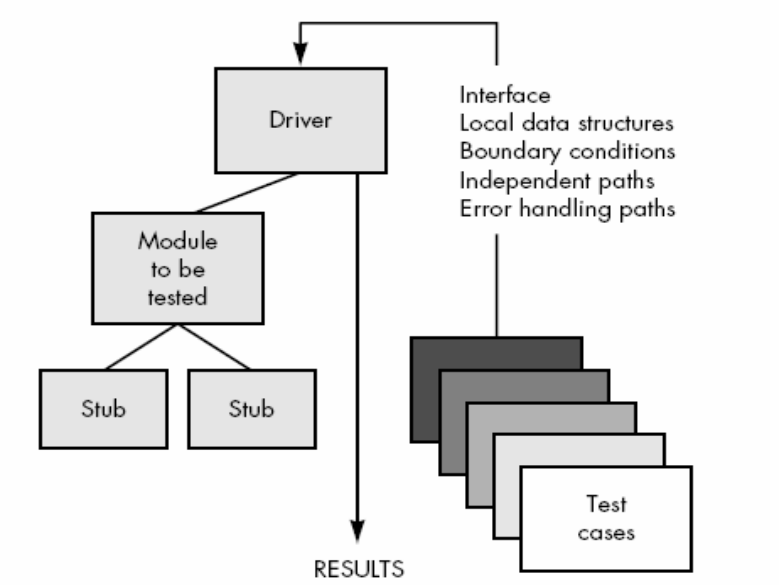
Các bước kiểm thử không tăng tiến :

1. Kiểm thử từng module chức năng 1 cách độc lập.
2. Tích hợp chúng lại thành chương trình.
3. Để kiểm thử 1 module độc lập, ta cần viết 1 Driver và nhiều Stub cho nó.



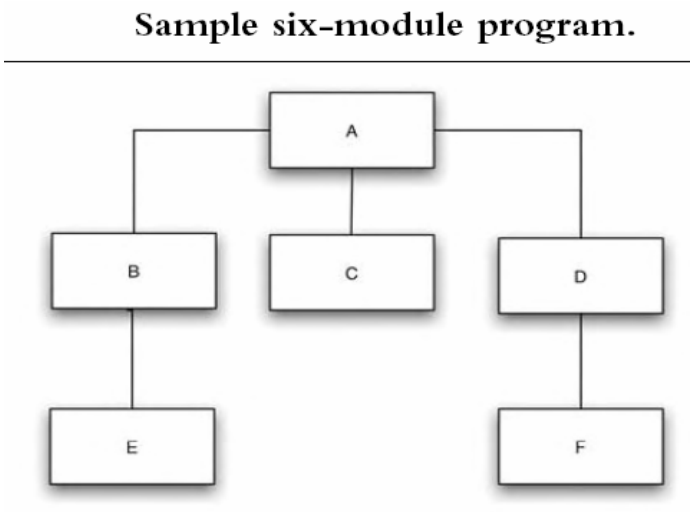
Driver là module có nhiệm vụ kích hoạt các testcase để kiểm thử module đang cần kiểm thử.

Stub là một hiện thực ở mức độ tối thiểu nào đó cho 1 module chức năng được dùng bởi module đang cần kiểm thử.



Các ý tưởng kiểm thử tăng tiến :

- Trước khi kiểm thử module mới, ta tích hợp nó vào tập các module đã kiểm thử rồi.
- Tích hợp các module theo thứ tự nào ? Từ trên xuống (top-down) hay từ dưới lên (bottom-up).
- Có phải kiểm thử tăng tiến tốt hơn kiểm thử không tăng tiến ?



Một số ý quan sát :

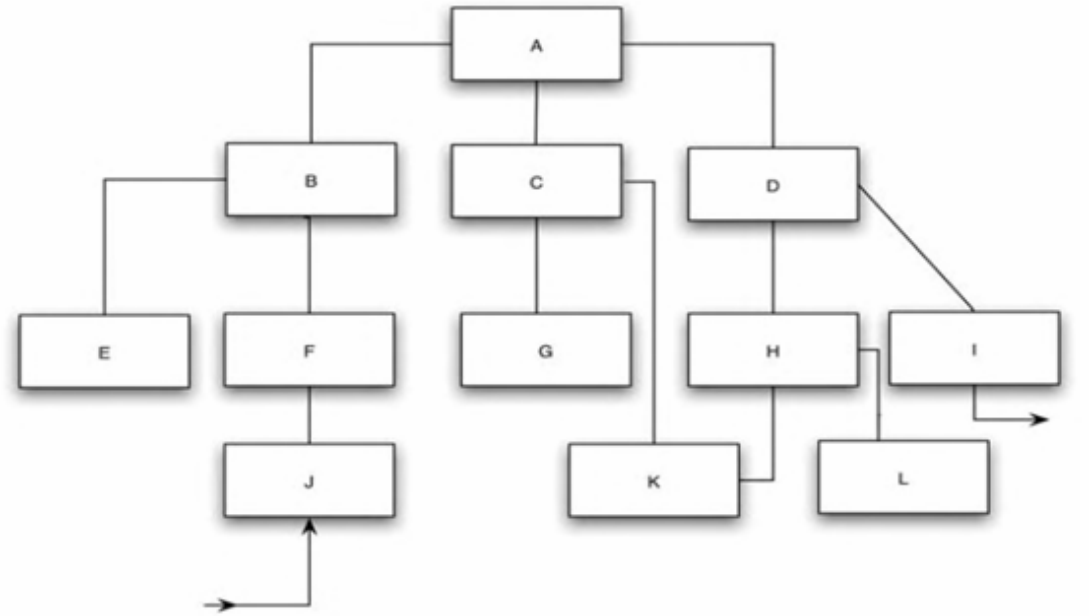
- Kiểm thử tăng tiến cần nhiều công sức hơn kiểm thử không tăng tiến.

- Các lỗi liên quan đến giao tiếp giữa các module sẽ được phát hiện sớn hơn vì việc tích hợp các module xảy ra sớm hơn so với kiểm thử không tăng tiến.
- Kiểm thử tăng tiến cũng sẽ giúp debug dễ dàng hơn.
- Kiểm thử tăng tiến sẽ hiệu quả hơn.
- Kiểm thử không tăng tiến dùng ít thời gian máy hơn (vì chỉ tiến hành trên từng module độc lập).
- Ở đầu chu kỳ kiểm thử module, kiểm thử không tăng tiến tạo cơ hội tốt cho việc kiểm thử đồng thời trên nhiều module khác nhau.

8.4 Kiểm thử từ trên xuống (top-down)

Gồm các bước theo thứ tự :

1. Bắt đầu từ module gốc ở trên cùng của cây cấu trúc chương trình.
2. Sau khi kiểm thử xong module hiện hành, ta chọn module kế tiếp theo ý tưởng :
    - Module kế tiếp phải được dùng trực tiếp bởi module được kiểm thử rồi.
    - Vì có nhiều module cùng thỏa mãn điều kiện trên, nên ta chọn module thực hiện nhiều hoạt động I/O trước.
    - Rồi chọn module "critical", là module dùng thuật giải phức tạp, tiềm ẩn nhiều lỗi và/hoặc lỗi nặng nhất.



Kiểm thử module A trước. Để kiểm thử module A, ta cần phải xây dựng các Stub cho 3 module mà A phụ thuộc trực tiếp là B, C, D.

Tạo các testcase cho module A như thế nào ?

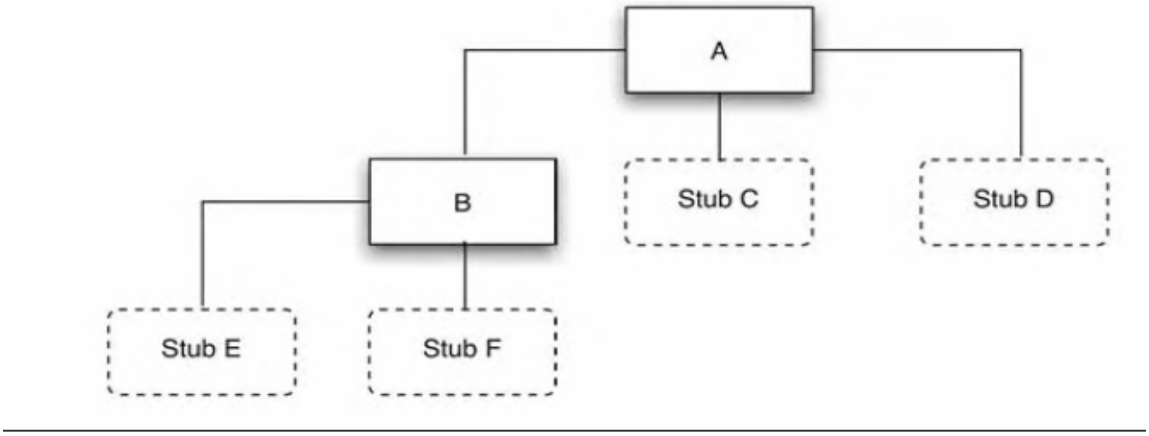
- Các Stubs sẽ có nhiệm vụ cung cấp dữ liệu cho module cần kiểm thử :
  - B—cung cấp thông tin tổng kết về giao tác.
  - C—xác định trạng thái hàng tuần có thỏa quota không?
  - D—tạo báo cáo tổng kết hàng tuần.
- Như vậy 1 testcase cho A là tổng kết giao tác từ B gởi về, Stub D có thể chứa các lệnh để xuất dữ liệu ra máy in để ta có thể xem xét kết quả của mỗi test case.

Nếu module A gọi module B chỉ 1 lần, làm sao cung cấp nhiều dữ liệu test khác nhau cho A ?

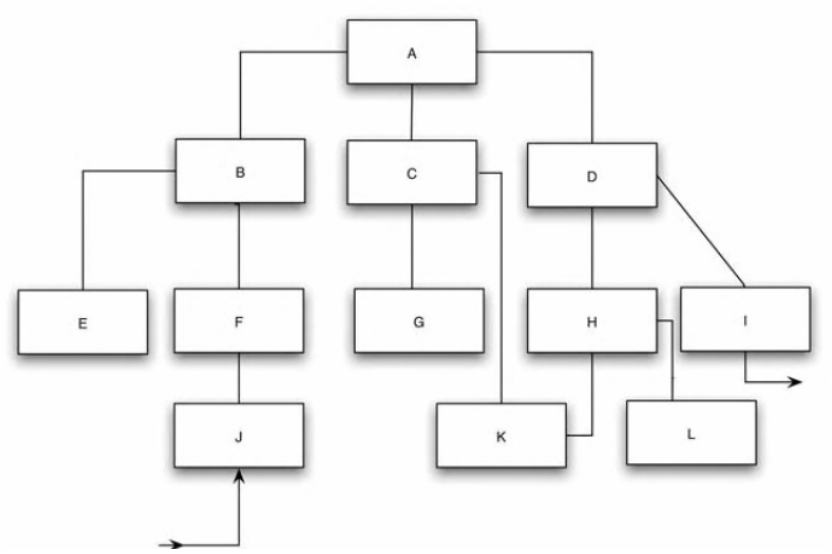
- Viết nhiều version khác nhau cho Stub B, mỗi version cung cấp 1 dữ liệu test xác định cho A.
- Đặt dữ liệu test ở file bên ngoài B, Stub B đọc vào và return cho A.

Sau khi kiểm thử xong module hiện hành, ta chọn 1 trong các Stub và thay thế nó bằng module thật rồi kiểm thử module thật này :

Second step in the top-down test.



Có nhiều thứ tự kiểm thử khác nhau có thể được chọn như dưới đây, và nếu cần kiểm thử đồng thời, cũng có nhiều thứ tự khác nữa.

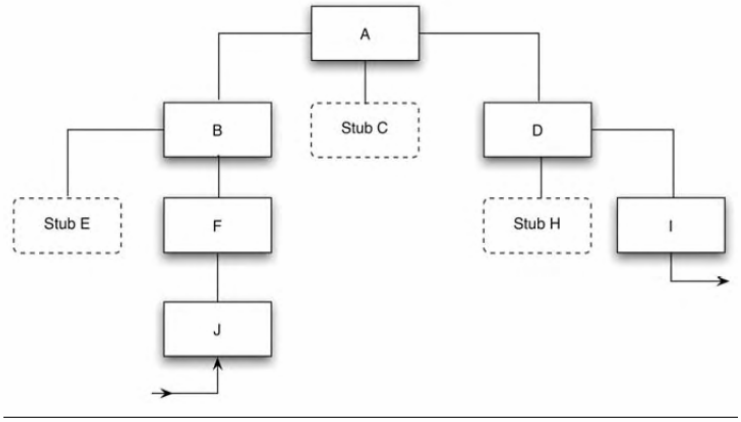


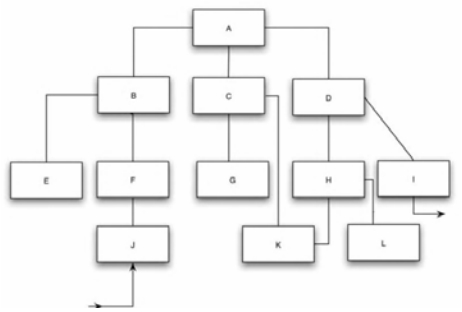
1. A B C D E F G H I J K L
2. A B E F J C G K D H L I
3. A D H I K L C G B F J E
4. A B F J D I E C G K H L
5. ...

Nếu module J và I thực hiện nhập/xuất dữ liệu, còn module G là critical, ta nên chọn thứ tự kiểm thử tăng tiến sau đây :

A B F J D I C G E K H L

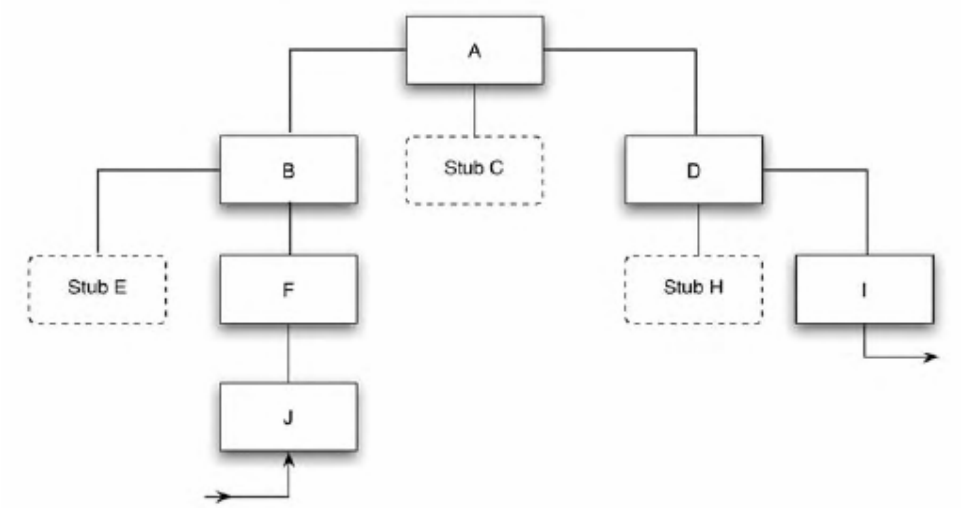
Intermediate state in the top-down test.





Một khi đã kiểm thử đến trạng thái như hình dưới đây :

- Việc miêu tả các testcase và việc thanh tra kết quả sẽ đơn giản.
- Ta đã có 1 version chạy được của chương trình, nó thực hiện được các hoạt động nhập/xuất dữ liệu.
- Ta có cảm giác chương trình hoàn chỉnh đến rất gần rồi.



Ưu điểm của phương pháp top-down

- Nếu các lỗi xảy ra có khuynh hướng nằm trong các module mức cao thì phương pháp top-down sẽ giúp phát hiện sớm chúng.
- Một khi các hoạt động nhập/xuất dữ liệu đã được thêm vào hệ thống thì việc miêu tả các test case sẽ dễ dàng hơn.
- Chương trình khung sườn sớm hình thành để demo và tiếp thêm sức mạnh tinh thần cho những người phát triển phần mềm.

Khuyết điểm của phương pháp top-down

- Phải viết các Stub để kiểm thử module dùng chúng, và viết Stub thường phức tạp hơn nhiều so với suy nghĩ của chúng ta.
- Trước khi các hoạt động nhập/xuất được tích hợp vào hệ thống, việc miêu tả các testcase trong các Stub có thể gặp khó khăn.
- Việc tạo ra điều kiện kiểm thử sẽ rất khó và nhiều lúc là không khả thi : Do có khoảng cách khá xa giữa module cần test và module nhập dữ liệu cung cấp cho module cần test nên rất khó để cung cấp dữ liệu gì để có thể kiểm thử 1 tình huống xác định của module cần kiểm thử.

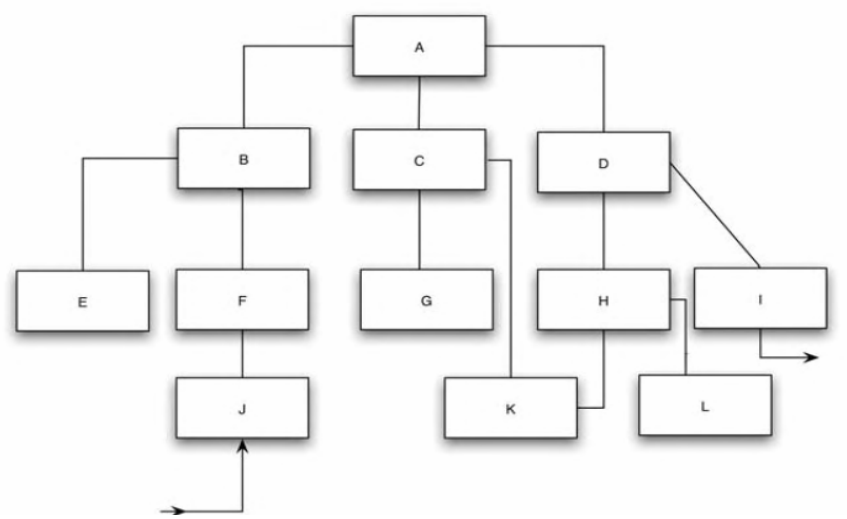
- Quan sát kết quả kiểm thử sẽ gặp khó khăn : Tương tự, xem xét sự tương quan dữ liệu xuất của 1 module và dữ liệu nhập tạo dữ liệu xuất này (nhưng ở trong module có khoảng cách khá xa) sẽ rất khó khăn.
- Nó làm ta nghĩ rằng việc thiết kế và kiểm thử có cùng thứ tự thực hiện : ta sẽ cảm nhận rằng hoạt động kiểm thử và hoạt động thiết kế gối đầu nhau : thiết kế tới đâu thì kiểm thử tới đó. Điều này thật nguy hiểm vì nếu ta tiến hành thiết kế và kiểm thử gối đầu nhau như vậy thì khi kiểm thử tới các module phía dưới mà đòi hỏi hiệu chỉnh bản thiết kế của module phía trên thì sẽ gây lãng phí rất lớn (vì phải huỷ bỏ kết quả đã có và thiết kế lại từ đầumodule phía trên).
- Nó trì hoãn việc kiểm thử 1 số module.
- Nó làm ta dễ quên hiện thực module chức năng vì đã có Stub thay thế.
- Khó lòng kiểm thử đầy đủ module cần kiểm thử trước khi tiến hành kiểm thử module khác. Điều này là do 2 lý do chính sau đây :
  - Các module Stub khó lòng tạo được tất cả dữ liệu thật mà module thực sự tương ứng sẽ tạo ra.
  - Các module cấp trên của cấu trúc chương trình thường chứa các đoạn code tạo, thiết lập trạng thái đầu của các tài nguyên mà sẽ được dùng trong các module phía dưới, nhưng hiện nay module phía dưới chưa được kiểm thử nên không thể kiểm thử các đoạn code thiết lập tài nguyên này được.

8.5 Kiểm thử từ dưới lên (bottom-up)

Gồm các bước theo thứ tự :

1. Bắt đầu từ 1 hay nhiều module lá : module mà không gọi module nào khác.

2. Sau khi kiểm thử xong module hiện hành, ta chọn module kế tiếp theo ý tưởng :
    - Module kế tiếp phải dùng trực tiếp 1 hay nhiều module được kiểm thử rồi.
    - Vì có nhiều module cùng thỏa mãn điều kiện trên, nên ta chọn module thực hiện nhiều hoạt động I/O trước.
    - Rồi chọn module "critical", là module dùng thuật giải phức tạp, tiềm ẩn nhiều lỗi và/hoặc lỗi nặng nhất.

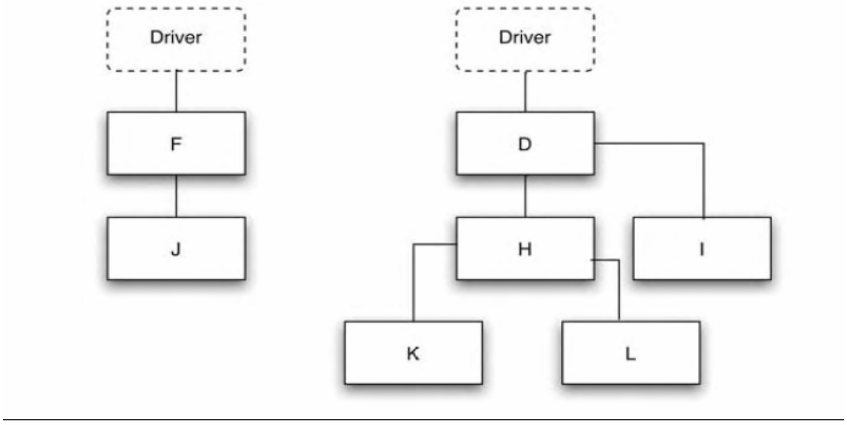


Các module E, J, G, K, L và I được kiểm thử trước.

Để kiểm thử 1 module, ta phải viết driver cho nó. Không cần phải viết nhiều driver khác nhau cho cùng 1 module. Trong đại đa số trường hợp, viết driver dễ hơn nhiều so với viết Stub.

Nếu D và F là 2 module critical nhất, thì ta nên kiểm thử theo trình tự của hình vẽ dưới đây :

Intermediate state in the bottom-up test.



Ưu & khuyết điểm của phương pháp bottom-up

- Ưu :
  - Nếu các lỗi xảy ra có khuynh hướng nằm trong các module mức thấp thì phương pháp bottom-up sẽ giúp phát hiện sớm chúng.
  - Việc tạo các điều kiện kiểm thử sẽ dễ dàng hơn.
  - Việc quan sát các kết quả kiểm thử cũng dễ dàng hơn.
- Khuyết :
  - Phải viết các module driver, mặc dù việc viết các module này khá dễ dàng.
  - Chương trình khung sườn chưa tồn tại 1 thời gian dài cho đến khi module cuối cùng được tích hợp vào hệ thống.

8.6 Kết chương

Chương này đã trình bày các vấn đề cơ bản về hoạt động kiểm thử đơn vị, hay còn gọi là kiểm thử module.

Chúng ta cũng đã trình bày các kỹ thuật kiểm thử đơn vị thường dùng như kỹ thuật kiểm thử không tăng tiến, kỹ thuật kiểm thử tăng tiến từ trên xuống, kỹ thuật kiểm thử tăng tiến từ dưới lên cùng các ưu/khuyết điểm của từng kỹ thuật.