Lý thuyết
Kiểm Tra Phần Mềm

Bài 06: ***Kiểm tra module***

GV: Nguyễn Ngọc Tú

Email: nntu@hoasen.edu.vn

Bộ môn: Kỹ thuật Phần mềm

Nội dung

- Giới thiệu
- Kiểm thử đơn vị
- Thiết kế kịch bản kiểm thử
- Kiểm tra gia tăng
- Phương pháp Top-Down và Bottom-Up

Giới thiệu

- Yếu tố liên quan Thiết kế mẫu thử
  - Cơ chế kiểm thử
  - Kích cỡ mã / chương trình
- Kiểm thử đơn vị
  - quá trình kiểm tra từng chương trình con/ thủ tục/ hàm
  - kiểm tra từng phần nhỏ của chương trình

Kiểm thử đơn vị

- Lý do kiểm thử đơn vị
  - quản lý các phần tử được kết hợp / phần nhỏ của chương trình
  - giảm nhẹ công việc gỡ rối
  - thực hiện song song với quá trình kiểm thử toàn bộ chương trình

Kiểm thử đơn vị

- Mục tiêu
  - so sánh chức năng của module đang thực hiện với chức năng hoặc giao diện của module đã được định nghĩa / đặc tả
  - làm sao thấy hết các "*mâu thuẫn/ trái ngược*" so với đặc tả
- Các góc nhìn
  - Cách thức mà Mẫu kiểm thử được thiết kế
  - Thứ tự các module được kiểm thử và tích hợp

Thiết kế kịch bản kiểm thử

- Cần hai kiểu thông tin khi thiết kế mẫu thử cho kiểm tra đơn vị:
  - Đặc tả về module
    - Định nghĩa các đặc trưng nhất của module bao gồm: các thông số nhập, suất và chức năng của nó
  - Mã nguồn của module

- Cách kiểm thử chính
  - Kiểm thử White-Box
  - Giai đoạn sau nhằm tìm lỗi dạng khác

Thiết kế kịch bản kiểm thử

- Thủ tục thiết kế mẫu thử
  - Phân tích luồng logic với một hay nhiều phương pháp White-Box
  - Bổ sung thêm các mẫu thử này bằng việc áp dụng phương thức Black-Box dựa trên đặc tả

Các phương thức thiết kế mẫu kiểm thử sử dụng đã được định nghĩa trong chương trước

Kiểm tra gia tăng

- Trong việc thực hiện quá trình kiểm thử đơn vị, có hai vấn đề quan tâm chủ chốt:
  - Thiết kế tập mẫu thử hiệu quả
  - Cách thức các module hợp thành chương trình làm hoạt động được
    - Quan trọng vì liên quan tới
      - dạng mẫu thử đơn vị được viết,
      - kiểu công cụ kiểm thử có thể sử dụng,
      - thứ tự module được mã và kiểm thử,
      - chi phí phát sinh mẫu thử,
      - chi phí gỡ lỗi (định vị và sửa lỗi phát hiện).

Kiểm tra gia tăng

- Câu hỏi được đặt ra
  - nên kiểm thử chương trình bằng cách kiểm thử từng module độc lập rồi kết hợp các module thành chương trình
    - Cách tiếp cận này gọi là kiểm thử/tích hợp không gia tăng hoặc còn gọi là big-bang;
  - hay kết hợp thêm các module "mới" với các module đã được kiểm thử trước đó.
    - cách tiếp cận sau được biết như là kiểm thử/tích hợp gia tăng

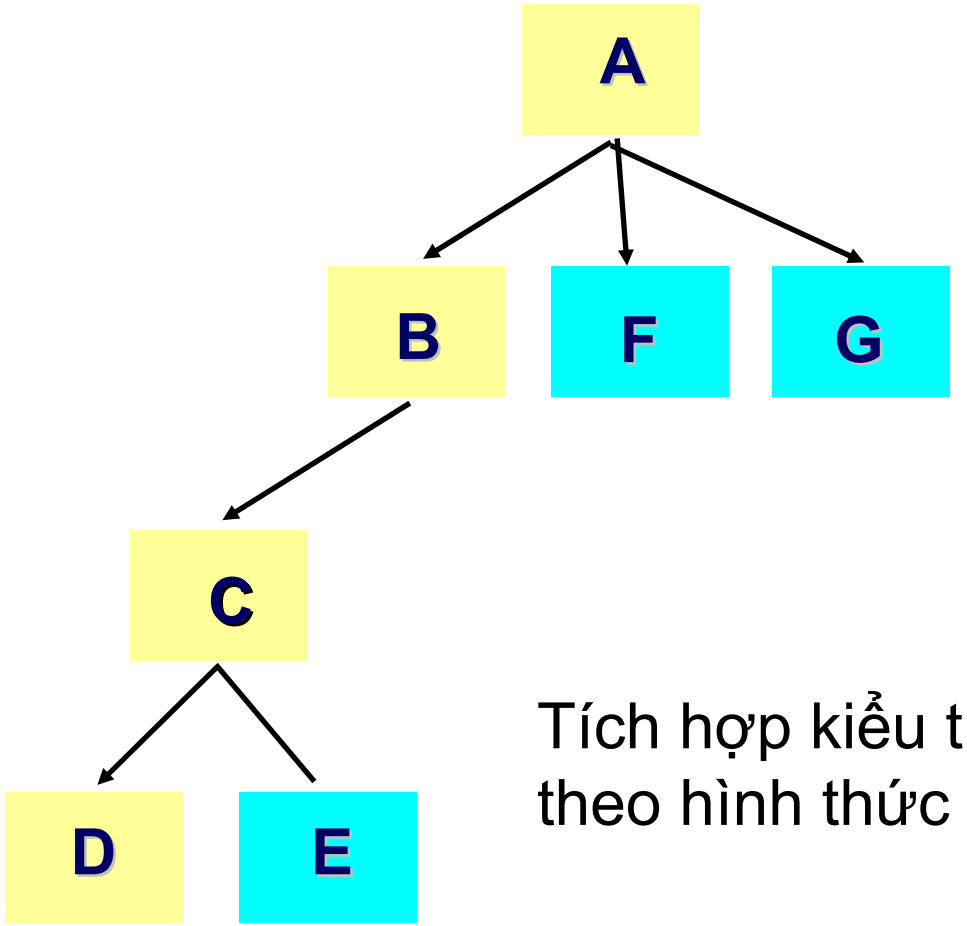
Kiểm tra gia tăng

- Kiểm thử trên xuống (*Top-Down*)
- Kiểm thử/tích hợp dưới lên (*Bottom Up*)

Kiểm thử trên xuống

- Kiểm thử trên xuống tiến hành kiểm thử với các mô đun ở mức cao trước, các mô đun mức thấp được tạm thời phát triển với các chức năng hạn chế.
- Thông thường, để sớm có một phiên bản thực hiện người ta thường tích hợp theo một nhánh cho đến các mô đun cấp thấp nhất.

Kiểm thử trên xuống



Tích hợp kiểu từ trên xuống theo hình thức *depth-first*

Kiểm thử trên xuống

- *Module* chính được dùng như là *driver*, và *stub* được thay thế bởi các *module* con trực tiếp của của *module* chính này.
- Tuỳ thuộc vào cách tích hợp theo chiều sâu (*depth-first*) hoặc chiều ngang(*breath-first*), mỗi *stub* con được thay thế một lần bởi *module* tương ứng đã kiểm nghiệm.
- Tiến hành kiểm nghiệm khi có sự thay thế mới
- Tiến hành kiểm nghiệm hồi quy để phát hiện các lỗi khác trong từng *module*

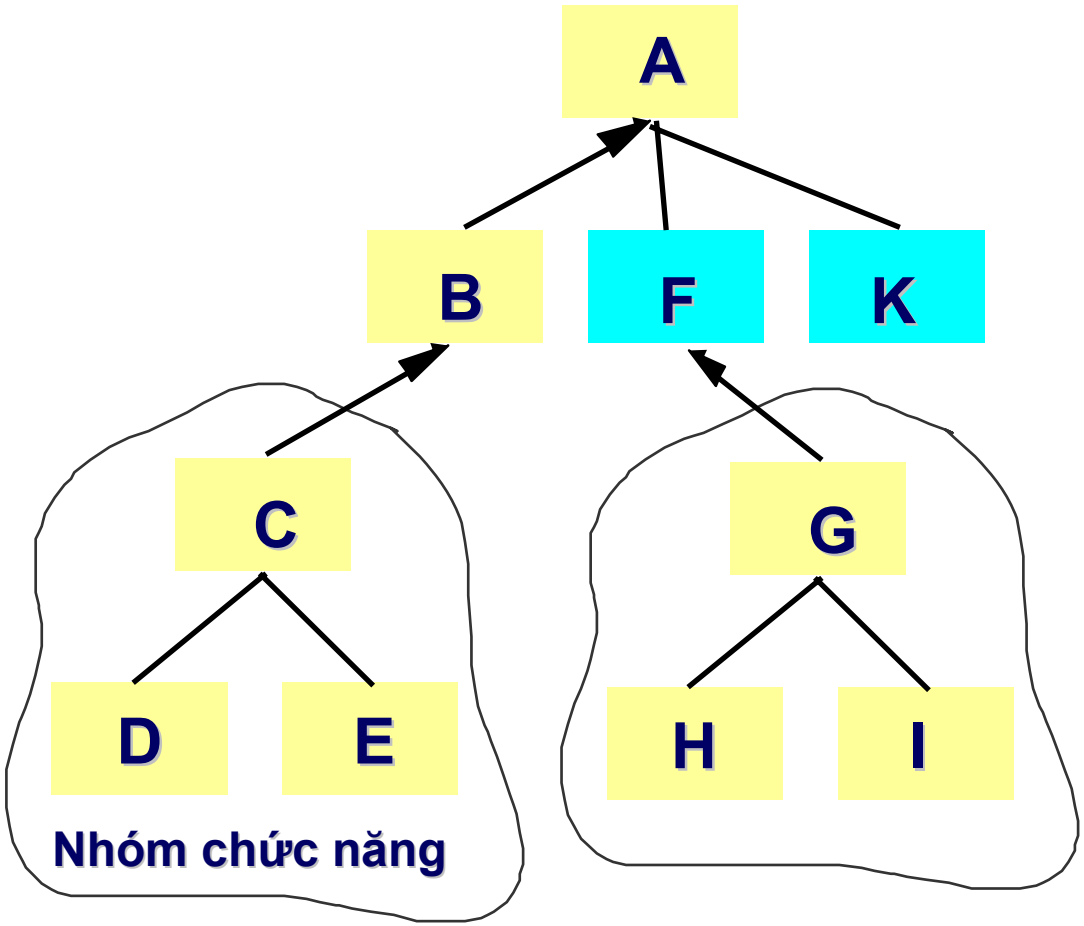
Kiểm thử trên xuống

- Ưu điểm của kiểm thử trên xuống
  - Phát hiện sớm các lỗi thiết kế
  - Có phiên bản hoạt động sớm
- Nhược điểm của kiểm thử trên xuống
  - Khó có thể mô phỏng được các chức năng của mô đun cấp thấp phức tạp
  - Không kiểm thử đầy đủ các chức năng

Kiểm thử dưới lên

- Là quá trình tích hợp và kiểm thử với các mô đun ở mức độ thấp trước.
- Thông thường người ta không thuần túy kiểm thử tất cả các mô đun ở tầng dưới cùng mà nhóm các mô đun này thành các nhóm chức năng, tích hợp và kiểm thử chúng theo từng nhóm.
- Tiến hành tích hợp và kiểm thử một số mô đun cấp trên trước

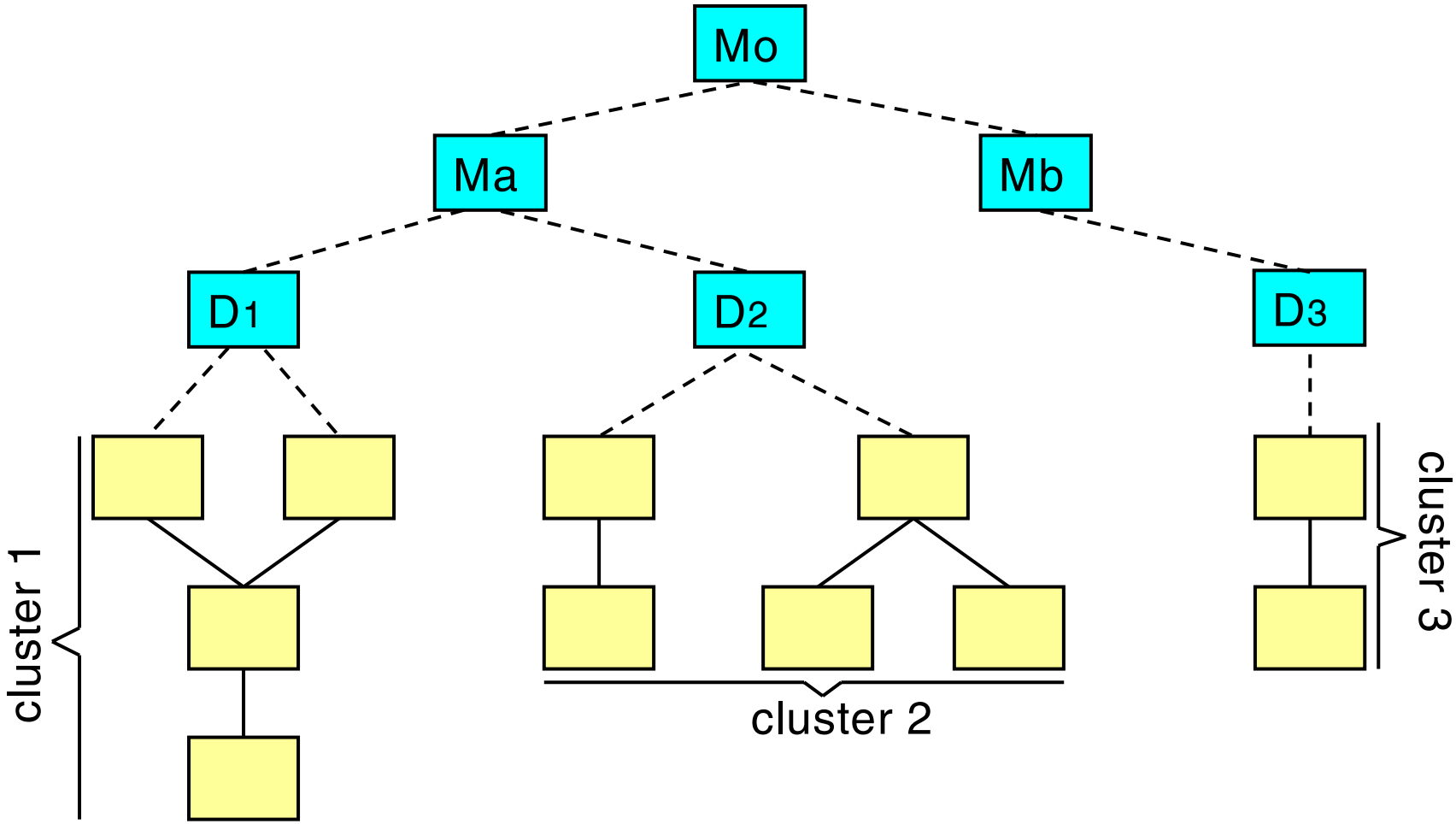
Kiểm thử dưới lên



Kiểm thử dưới lên

- Các *module* mức thấp nhất được kết hợp thành các nhóm thể hiện một chức năng con đặc biệt của phần mềm.
- Một *driver* được tạo ra để thao tác các *test-case*
- Nhóm module được kiểm nghiệm.
- *Driver* được bỏ đi và các nhóm *module* được kết hợp dần lên phía trên trong sơ đồ phân cấp của chương trình.

Kiểm thử dưới lên



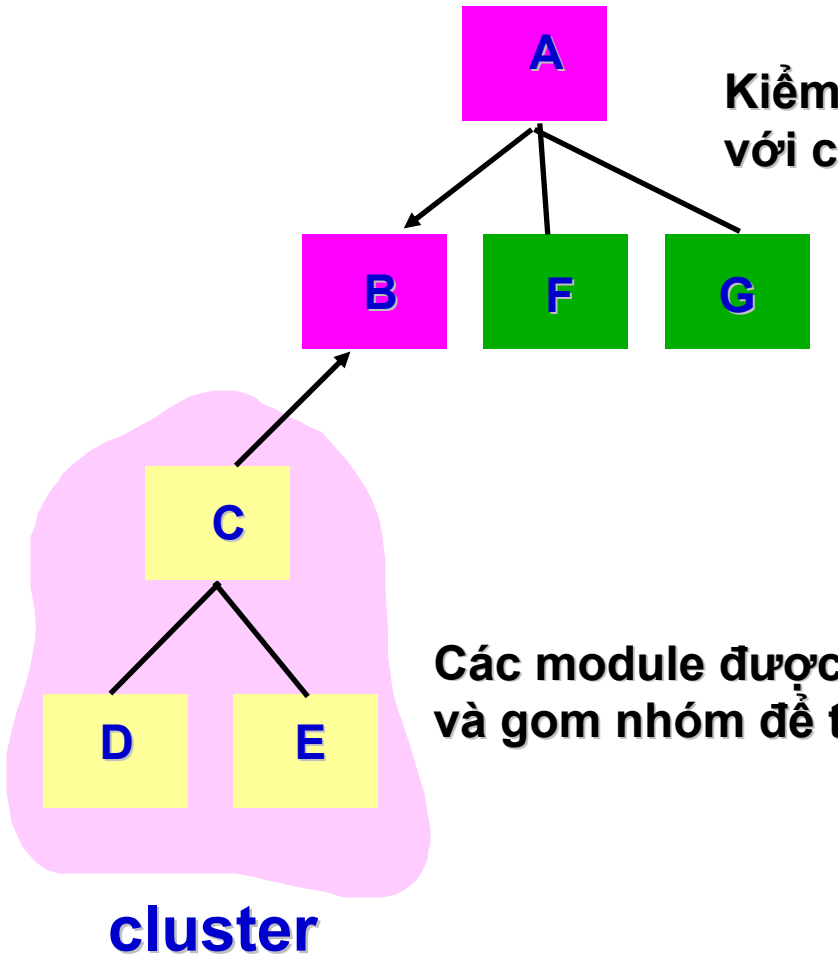
Kiểm thử dưới lên

- Kiểm thử dưới lên có một số ưu điểm:
  - Tránh phải tạo các stub phức tạp hay tạo các kết quả nhân tạo
  - Thuận tiện cho phát triển các mô đun thứ cấp dùng lại được
- Nhược điểm của phương pháp bottom-up:
  - Phát hiện chậm các lỗi thiết kế
  - Chậm có phiên bản thực hiện được của hệ thống

Top-Down và Bottom-Up

- Trên thực tế người ta thường tìm cách phối hợp hai chiến lược này, gọi là *sandwich testing*

Sandwich Testing



Kiểm tra từ trên xuống
với các stubs

Các module được kiểm tra
và gom nhóm để tích hợp

Bài tập - đọc hiểu

Xét hàm chức năng chép bộ nhớ sau

```
void   copy_memory(   char * target,
          char * source,
          int length) {
    char * source_ptr,
          * source_end,
          * target_ptr;
    int step;
    int done = 0;
    if (length < 1)          return;
    if ((target < source) &&
          ((target + length) > source))
          {   // {*}
        source_ptr = source
                    + (length-1);
        source_end = source;
        target_ptr = target + (length-1);
        step = -1;
    }
else {
          source_ptr = source;
          source_end = source + (length-1);
          target_ptr = target;
          step = 1;
      }
      while (1) {
          if (source_ptr == source_end)
done = 1;
          *target_ptr = *source_ptr;
          if (done)    break;
          source_ptr += step;
          target_ptr += step;
      }
  }
```

- Xác định các thông tin sau:
  - Xem xét biến ***done*** được dùng ở đâu và đúng không ?
  - Dòng mã IF ở {*} có thực sự tính toán đúng khi muốn dịch chuyển dữ liệu nội bộ trong một chuỗi ?
  - Thử trường hợp dữ liệu đầu vào là rỗng hoặc NULL ?
  - Xác định rõ ý đồ đoạn mã tử {*} ?
- Kiểm thử các trường hợp sau:
  - Thử chép dữ liệu không bị trùng lặp:
    - *test_buffer = "123456" → copy_memory(test_buffer, test_buffer+3, 3)*
  - Trường hợp dữ liệu bị trùng lặp:
    - *test_buffer = "123456" → copy_memory(test_buffer, test_buffer+2, 4)*

- Xác định các đường – bao phủ đường
- Bao phủ điều kiện

Đọc thêm

- [1]. Chapter 05
- ql072l@sinhvien.hoasen.edu.vn

Q/A