HUTECH
Đại học Công nghệ Tp.HCM

KHOA CÔNG NGHỆ THÔNG TIN

CÔNG NGHỆ PHẦN MỀM

Bài 7:

TÁC VỤ KIỂM THỬ PHẦN MỀM

Thời gian: 3 tiết

Giảng viên: ThS. Dương Thành Phết

Email: phetcm@gmail.com

Website: http://www.thayphet.net

Tel: 0918158670

facebook.com/DuongThanhPhet

NỘI DUNG

1. Tổng quan
2. Yêu cầu đối với kiểm thử
3. Các kỹ thuật kiểm thử
4. Chiến lược và các giai đoạn
5. Ví dụ minh họa

7.1. TỔNG QUAN

7.1.1. Lịch sử kiểm thử phần mềm

7.1.2. Giới thiệu

7.1.3. Mô hình chữ V

7.1.1. LỊCH SỬ KIỂM THỬ PHẦN MỀM

- ✓ Sự tách biệt giữa việc gỡ lỗi (sửa lỗi, debugging) với kiểm thử (testing) được Glenford J. Myers đưa ra vào năm 1979.
- ✓ Mặc dù sự quan tâm của ông là kiểm thử sự gián đoạn ("một kiểm thử thành công là tìm ra được một lỗi"), nhưng điều đó minh họa mong muốn của cộng đồng công nghệ phần mềm để tách biệt các hoạt động phát triển cơ bản, giống như việc tách phần gỡ lỗi ra riêng khỏi quá trình kiểm thử.
- ✓ Vào năm 1988, Dave Gelperin và William C. Hetzel đã phân loại các giai đoạn và mục tiêu trong kiểm thử phần mềm theo trình tự sau:
- ✓ Trước 1956: Hướng về việc kiểm soát lỗi.
- ✓ 1957-1978: Hướng về chứng minh lỗi.
- ✓ 1979-1982: Hướng về tính phá hủy của lỗi.
- ✓ 1983–1987: Hướng về đánh giá lỗi.
- ✓ 1988–2000: Hướng về việc phòng ngừa lỗi.

7.1.1. LỊCH SỬ KIỂM THỬ PHẦN MỀM

- ✓ Một nghiên cứu được tiến hành bởi NIST trong năm 2002 cho biết rằng các lỗi phần mềm gây tổn thất cho nền kinh tế Mỹ 59,5 tỷ đô mỗi năm
- ✓ Hơn một phần ba chi phí này có thể tránh được nếu việc kiểm thử phần mềm được thực hiện tốt hơn.
- ✓ Theo đó, một kiếm khuyết nếu được tìm ra sớm hơn thì chi phí để sửa chữa nó sẽ rẻ hơn.

7.1.1. LỊCH SỬ KIỂM THỬ PHẦN MỀM

- ✓ Bảng dưới đây cho thấy chi phí sửa chữa các khiếm khuyết tùy thuộc vào giai đoạn nó được tìm ra.
- ✓ Ví dụ, một vấn đề được tìm thấy sau khi đã ra bản phần mềm chính thức rồi sẽ có chi phí gấp 10-100 lần khi giải quyết vấn đề từ lúc tiếp nhận yêu cầu.
- ✓ Với sự ra đời của cách thức triển khai thực tiễn liên tục và các dịch vụ dựa trên đám mây, chi phí tái triển khai và bảo trì có thể làm giảm bớt theo thời gian

Chi phí sửa chữa một khiếm khuyết		*Thời gian phát hiện*				
		Yêu cầu phần mềm	*Kiến trúc phần mềm*	*Xây dựng phần mềm*	*Kiểm thử hệ thống*	*Sau khi phát hành*
Thời gian sử dụng	Yêu cầu phần mềm	1×	3×	5–10×	10×	10–100×
	Kiến trúc phần mềm	–	1×	10×	15×	25–100×
	Xây dựng phầm mềm	–	–	1×	10×	10–25×

7.1.2. GIỚI THIỆU

- ✓ Kiểm thử phần mềm là việc tiến hành thí nghiệm để so sánh kết quả thực tế với lý thuyết nhằm mục đích phát hiện lỗi. *(Xem Phụ lục C – Phần E)*
- ✓ Bộ thử nghiệm (*test cases*) là dữ liệu dùng để kiểm tra hoạt động của chương trình.
- ✓ Bộ kiểm thử tốt là bộ có khả năng phát hiện ra lỗi của chương trình.
- ✓ Khi tiến hành kiểm thử, ta chỉ có *thể chứng minh được sự tồn tại của lỗi* nhưng **không chứng minh được** rằng trong *chương trình không có lỗi*.

7.1.2. GIỚI THIỆU

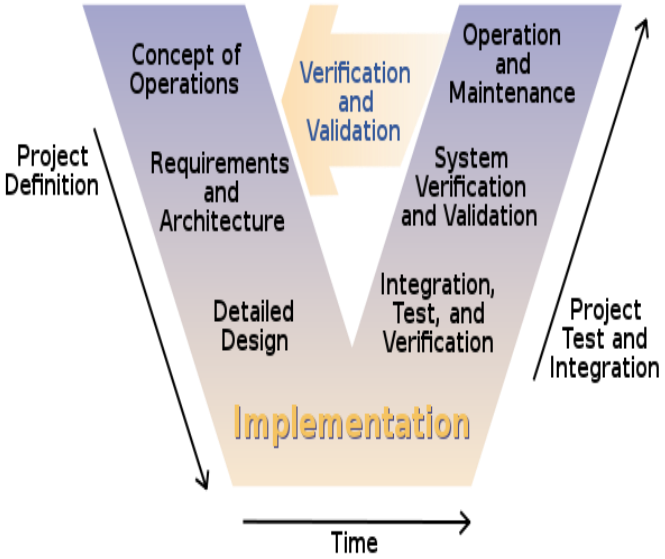
- ✓ Nội dung của bộ thử nghiệm gồm:
  - ✓ Tên mô-đun/chức năng muốn kiểm thử
- ✓ Dữ liệu vào:
  - ✓ Dữ liệu của chương trình: số, xâu ký tự, tập tin,
- ✓ Môi trường thử nghiệm: phần cứng, hệ điều hành,
- ✓ Thứ tự thao tác (kiểm thử giao diện)
- ✓ Kết quả mong muốn
- ✓ Thông thường: số, xâu ký tự, tập tin …
- ✓ Màn hình, thời gian phản hồi

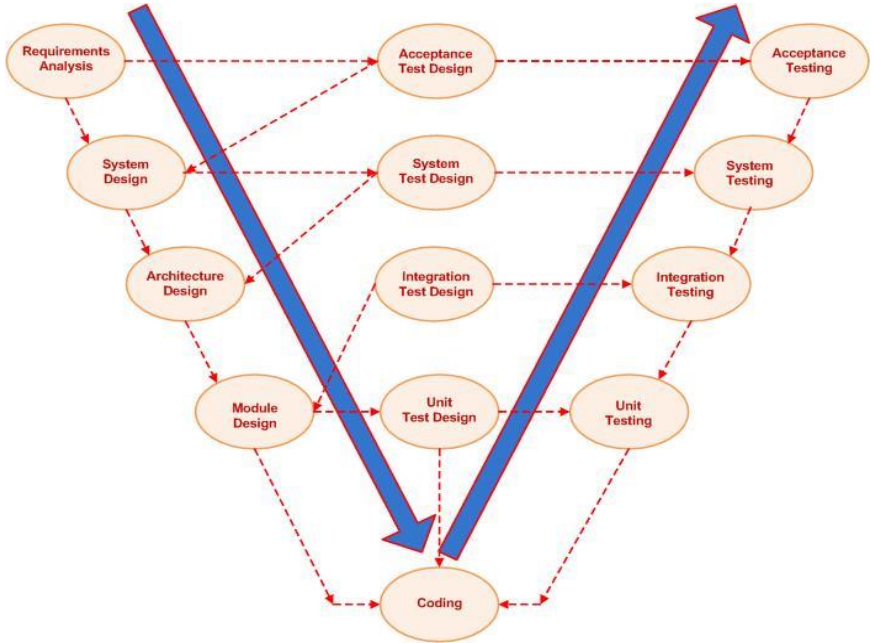
7.1.2. GIỚI THIỆU

- ✓ Kết quả thực tế
- ✓ Không gian thử nghiệm là tập các bộ số thử nghiệm. Không gian này nói chung là rất lớn. Nếu ta có thể vét cạn được không gian thử nghiệm thì chắc chắn qua việc xử lý của phép kiểm tra đơn vị thì sẽ chương trình sẽ không còn lỗi. Tuy nhiên điều này không khả thi trong thực tế. Do đó khi đề cập đến tính đúng đắn của phần mềm ta dùng khái niệm độ tin cậy.
- ✓ Phương pháp kiểm thử là cách chọn bộ số thử nghiệm để tăng cường độ tin cậy của đơn vị cần kiểm tra. Nói cách khác, phương pháp kiểm thử là cách phân hoạch không gian thử nghiệm thành nhiều miền rồi chọn bộ số liệu thử nghiệm đại diện cho miền đó. Như vậy ta cần tránh trường hợp mọi bộ thử nghiệm đều rơi vào một miền kiểm tra.

7.1.3. MÔ HÌNH CHỮ V

- ✓ Mô hình này (h7.1 và h7.2) biểu diễn sự liên hệ giữa kiểm thử và các tác vụ khác, đồng thời đại diện cho quá trình phát triển phần mềm (hay phần cứng) và có thể được xem là một mở rộng của mô hình thác nước.
- ✓ Xem thêm
  http://en.wikipedia.org/wiki/V-Model_(software_development)





7.1.3. MÔ HÌNH CHỮ V

- ✓ Mô hình chữ V thể hiện mối quan hệ giữa mỗi giai đoạn của vòng đời phát triển và giai đoạn liên quan của thử nghiệm.
- ✓ Các trục ngang và dọc đại diện cho thời gian hay hoàn thiện dự án (từ trái qua phải) và mức độ trừu tượng (thô ở tầng trừu tượng cao nhất), tương ứng. Mô hình chữ V hướng đến:
  - Thực hiện *Xác nhận* (Verification) với tác vụ phân tích yêu cầu, thiết kế hệ thống (kiến trúc, mô-đun …) ở mức cao và mức chi tiết
  - ✓ Kiểm tra *Hợp lệ* (*Validation*) với các tác vụ kiểm thử (đơn vị, tương tác, hệ thống, chấp nhận, triển khai)

7.2. YÊU CẦU ĐỐI VỚI KIỂM THỬ

- ✓ Ta cần chú ý các yêu cầu sau đây:
  - Tính lặp lại:
    - Kiểm thử phải lặp lại được (kiểm tra lỗi đã được sửa hay chưa)
    - Dữ liệu/trạng thái phải mô tả được
  - Tính hệ thống: phải đảm bảo đã kiểm tra hết các trường hợp.
  - Được lập tài liệu: phải kiểm soát được tiến trình/kết quả.

7.3.1. PHƯƠNG PHÁP HỘP ĐEN (BLACK BOX)

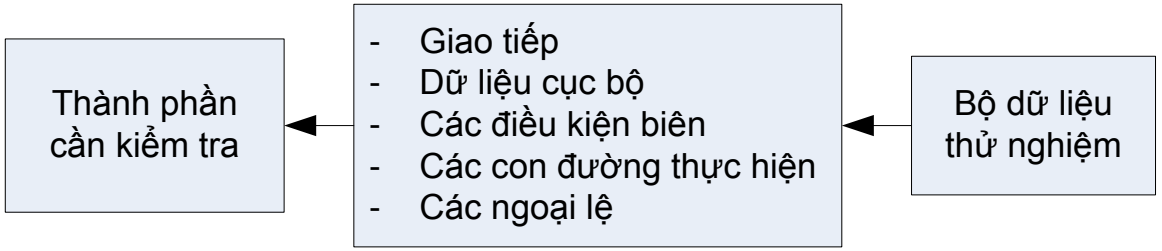
- ✓ Phương pháp kiểm thử dạng kiểm thử chức năng (*functional test*) chỉ dựa trên bản đặc tả các chức năng.
- ✓ Chỉ chú tâm đến *phát hiện các sai sót về chức năng* mà *không quan tâm đến cách hiện thực cụ thể*.
- ✓ Với phương pháp này ta có khả năng phát hiện các sai sót, thiếu sót về mặt chức năng; sai sót về giao diện của mô-đun, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.
- ✓ Do không thể kiểm thử mọi trường hợp trên thực tế, ta sẽ chia nhỏ không gian thử nghiệm dựa vào giá trị nhập xuất của đơn vị cần kiểm tra.
- ✓ Ứng với mỗi vùng dữ liệu, ta sẽ thiết kế những bộ thử nghiệm tương ứng và đặc biệt là các bộ thử nghiệm tại các giá trị biên của vùng dữ liệu.

7.3.1. PHƯƠNG PHÁP HỘP ĐEN (BLACK BOX)

- ✓ Để kiểm chứng chương trình giải phương trình bậc 2 theo phương pháp hộp đen, ta sẽ phân chia không gian thử nghiệm thành 3 vùng là “Vô nghiệm”, “Có nghiệm kép” hay “Có 2 nghiệm riêng biệt”.
- ✓ Sau khi đã kiểm tra thử với các bộ thử nghiệm đã thiết kế, ta cần mở rộng bộ thử nghiệm cho các trường hợp đặc biệt như: biên của số nguyên trong máy tính (32767, -32768), số không, số âm, số thập phân, dữ liệu sai kiểu, dữ liệu ngẫu nhiên.

7.3.2. PHƯƠNG PHÁP HỘP TRẮNG (WHITE BOX)

Phương pháp này hướng đến việc kiểm thử cấu trúc trong đó ta sẽ chia không gian thử nghiệm dựa vào cấu trúc của đơn vị cần kiểm tra



Trong đó:

- ✓ *Kiểm tra giao tiếp của đơn vị* là để đảm bảo dòng thông tin vào ra đơn vị luôn đúng (đúng giá trị, khớp kiểu …)
- ✓ *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. Ví dụ: nhập dữ liệu sai, tên biến sai, kiểu dữ liệu không nhất quán, ràng buộc hay ngoại lệ.

7.3.2. PHƯƠNG PHÁP HỘP TRẮNG (WHITE BOX)

- ✓ *Kiểm tra các điều kiện biên* của các câu lệnh điều kiện và 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. Con đường thực hiện của một đơn vị chương trình là một dãy có thứ tự các câu lệnh bên trong đơn vị đó sẽ được thực hiện khi kích hoạt đơn vị. Ví dụ: Con đường thực hiện của p1 và p2 như sau:

Thủ tục 1	**Thủ tục 2**
Lệnh 1	*Lệnh 1*
Lệnh 2	Nếu *đk* thì *Lệnh 2*
Lệnh 3	Ngược lại thì *Lệnh 3*
Lệnh 4	*Lệnh 4*

Khi đó, ta có các đường thực hiện sau:

Thủ tục 1: Lệnh 1 → Lệnh 2 → Lệnh 3 → Lệnh 4

Thủ tục 2: Lệnh 1 → Lệnh 2 → Lệnh 3 và Lệnh 1 → Lệnh 2 → Lệnh 4

7.3.3. PHƯƠNG PHÁP HỘP XÁM (GREY BOX

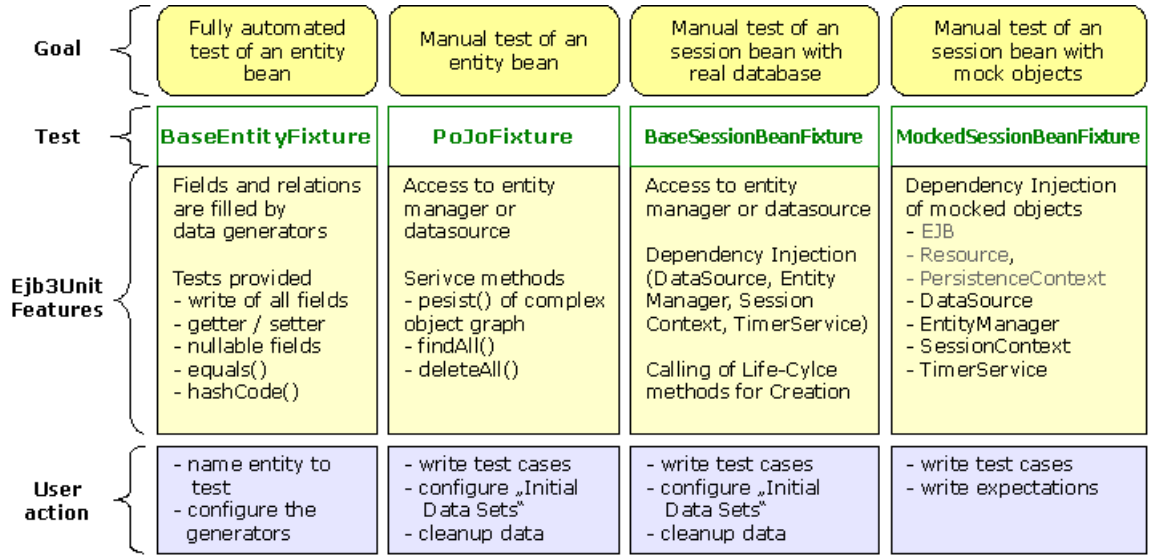
- ✓ Phương pháp này liên quan đến hiểu biết về cấu trúc dữ liệu bên trong và các thuật toán cho mục đích của các bài kiểm thử thiết kế.
- ✓ Khi thực hiện những bài kiểm thử với người dùng hoặc mức độ hộp đen, nhóm kiểm tra không nhất thiết phải truy cập vào mã nguồn của phần mềm.
- ✓ Ta có thể thao tác với dữ liệu đầu vào và định dạng đầu ra không xác định (như hộp xám), bởi vì đầu vào và đầu ra rõ ràng ở bên ngoài của hộp đen mà chúng được hệ thống gọi ra trong quá trình kiểm thử.
- ✓ Sự phân biệt này đặc biệt quan trọng khi tiến hành kiểm thử tích hợp giữa hai mô-đun được viết mã bởi hai nhóm phát triển khác nhau, chỉ có các giao diện được bộc lộ ra để kiểm thử.
- ✓ Tuy nhiên, với các kiểm thử có yêu cầu thay thế một kho lưu trữ dữ liệu bên dưới hệ thống (*back-end*) – như cơ sở dữ liệu hay tập tin đăng nhập – không xác định như hộp xám, người dùng sẽ không thể thay đổi các kho lưu trữ dữ liệu trong khi sản phẩm vẫn đang hoạt động bình thường.
- ✓ Kiểm thử hộp xám cũng có thể bao gồm kỹ thuật đảo ngược để xác định đối tượng, giá trị biên hoặc các thông báo lỗi.

7.3.3. PHƯƠNG PHÁP HỘP XÁM (GREY BOX

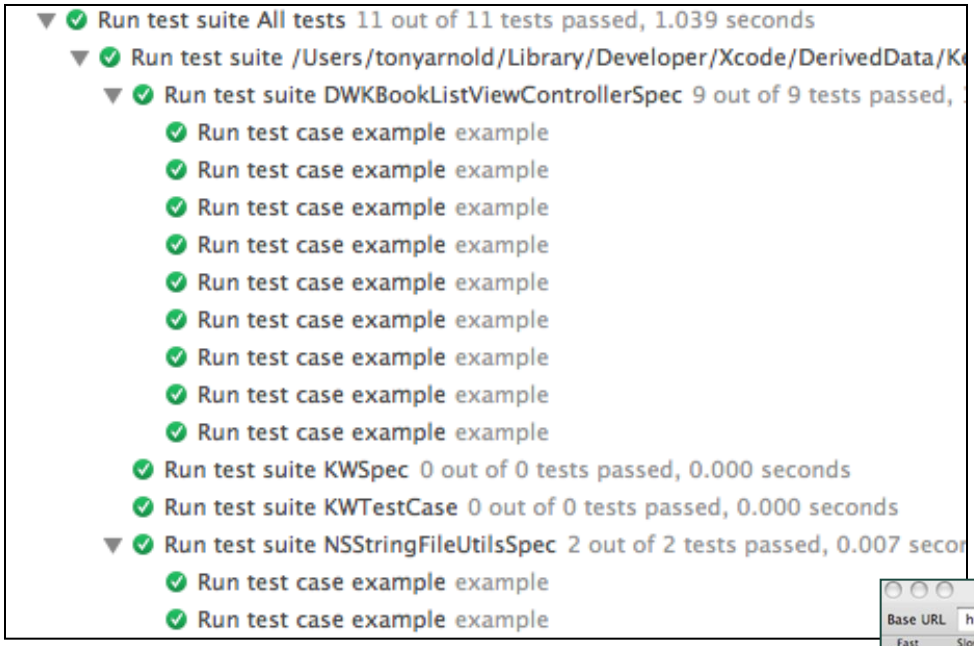
- ✓ Khi biết được những khái niệm cơ bản về cách thức các phần mềm hoạt động như thế nào, nhóm kiểm tra thực hiện kiểm thử phần mềm từ bên trong tốt hơn so với bên ngoài.
- ✓ Thông thường, nhóm kiểm thử theo phương pháp Hộp Xám sẽ được phép thiết lập một môi trường kiểm thử đã bị cô lập với các hoạt động liên quan cơ sở dữ liệu.
- ✓ Các kiểm thử có thể quan sát trạng thái của sản phẩm được kiểm thử (sau khi thực hiện hành động nhất định giống như việc thực hiện các câu lệnh SQL đối với cơ sở dữ liệu) và thực hiện truy vấn để đảm bảo những thay đổi dự kiến đã được phản ánh.
- ✓ Kiểm thử hộp xám thực hiện kịch bản kiểm thử thông minh, dựa trên thông tin hạn chế. Điều này sẽ đặc biệt áp dụng cho các kiểu xử lý dữ liệu, kể cả xử lý ngoại lệ

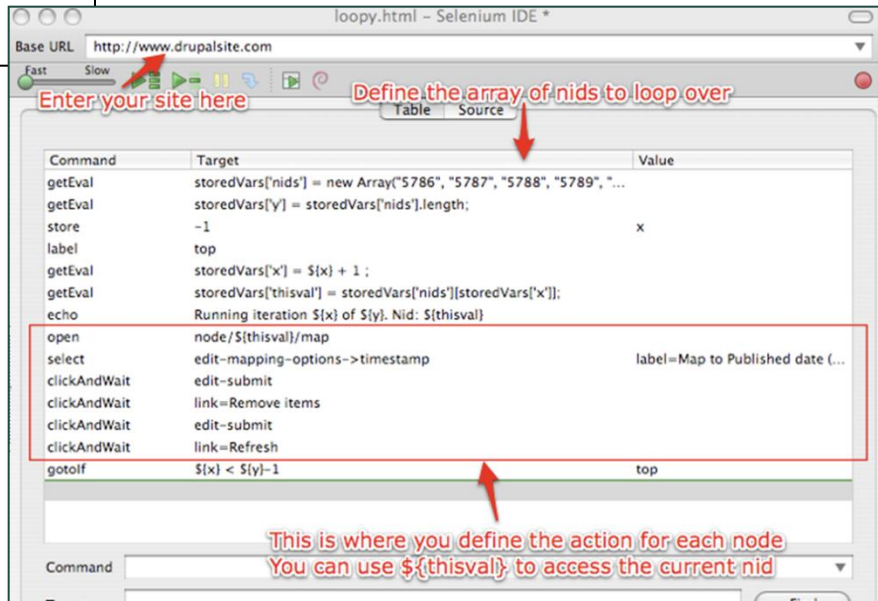
7.4. CHIẾN LƯỢC & CÁC GIAI ĐOẠN

- ✓ Với những dự án phần mềm lớn, những người tham gia được chia thành:
  - Nhóm thứ nhất: Những người chịu trách nhiệm kiểm tra các đơn vị của chương trình để chắc chắn chúng thực hiện đúng theo thiết kế.
  - Nhóm thứ hai: Các chuyên gia tin học độc lập và không thuộc nhóm thứ nhất, có nhiệm vụ phát hiện các lỗi do nhóm thứ nhất chủ quan còn để lại.
- ✓ Ví dụ minh họa về kiểm thử phần mềm



7.4. CHIẾN LƯỢC & CÁC GIAI ĐOẠN





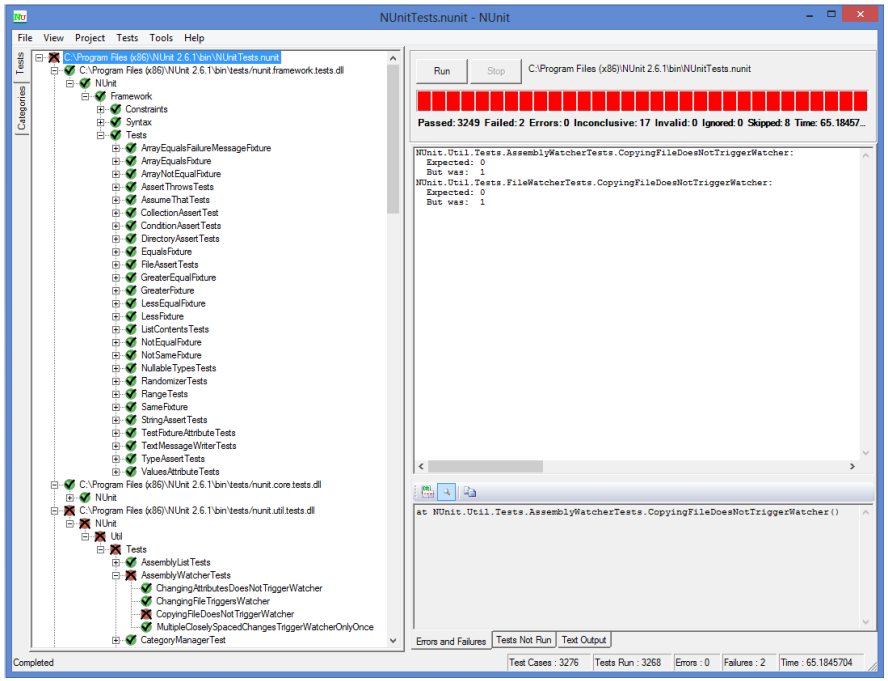
7.4.1. KIỂM THỬ ĐƠN VỊ (UNIT TEST)

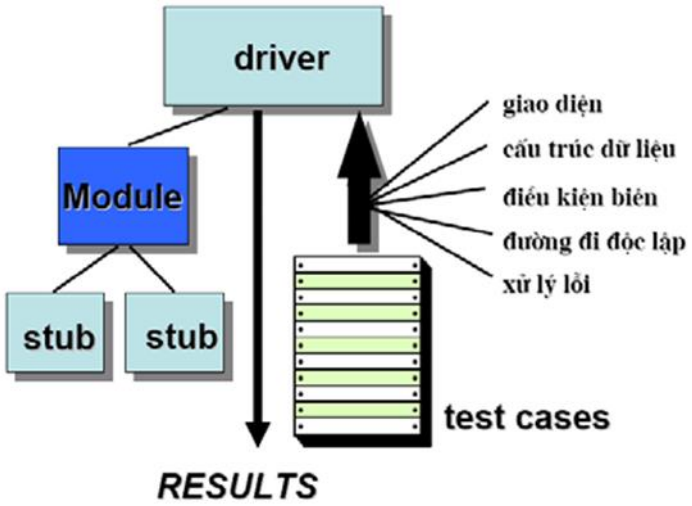
Việc kiểm thử được tiến hành qua các giai đoạn:

- ✓ Giai đoạn này *sử dụng kỹ thuật hộp trắng* và dựa vào *hồ sơ thiết kế* để xây dựng các bộ thử nghiệm *sao cho khả năng phát hiện lỗi là lớn nhất*.
- ✓ Vì đơn vị (*unit*) được kiểm tra thường là một đoạn chương trình hay một thủ tục nhưng không phải là 1 chương trình đầy đủ, hơn nữa đơn vị đó có thể được gọi thực thi bởi những đơn vị khác (hoặc gọi đến những đơn vị khác để thực thi), nên dù chương trình đã được hoàn tất đầy đủ các đơn vị, ta cũng *không nên giả thuyết sự tồn tại* hoặc *tính đúng đắn của các đơn vị khác* mà phải xây dựng các mô-đun giả lập đơn vị gọi (đặt tên là *driver*) và đơn vị bị gọi (đặt tên là *stub*):

7.4.1. KIỂM THỬ ĐƠN VỊ (UNIT TEST)

- ✓ *Driver* đóng vai trò như một chương trình chính nhập các bộ số thử nghiệm và gởi chúng đến đơn vị cần kiểm tra đồng thời nhận kết quả trả về của đơn vị cần kiểm tra.
- ✓ *Stub* là chương trình giả lập thay thế các đơn vị được gọi bởi đơn vị cần kiểm tra.
- ✓ Stub thực hiện các thao tác xử lý dữ liệu đơn giản như in ấn, kiểm tra dữ liệu nhập và trả kết quả ra.





7.4.2. KIỂM THỬ CHỨC NĂNG (FUNCTIONAL TEST)

- ✓ Như trình bày trong phần "Phương pháp Hộp đen", giai đoạn này thực hiện việc kiểm tra khả năng thực thi của các chức năng có đáp ứng được đầy đủ những yêu cầu (đã nêu trong bản đặc tả) của các chức năng đó hay không.
- ✓ Vì thế, ta tập trung phát hiện các sai sót về chức năng mà không quan tâm đến cách hiện thực cụ thể.
- ✓ Với phương pháp này ta có khả năng phát hiện các sai sót, thiếu sót về mặt chức năng; sai sót về giao diện của mô-đun, kiểm tra tính hiệu quả; phát hiện lỗi khởi tạo, lỗi kết thúc.

7.4.3. KIỂM THỬ TÍCH HỢP (INTEGRATION TEST)

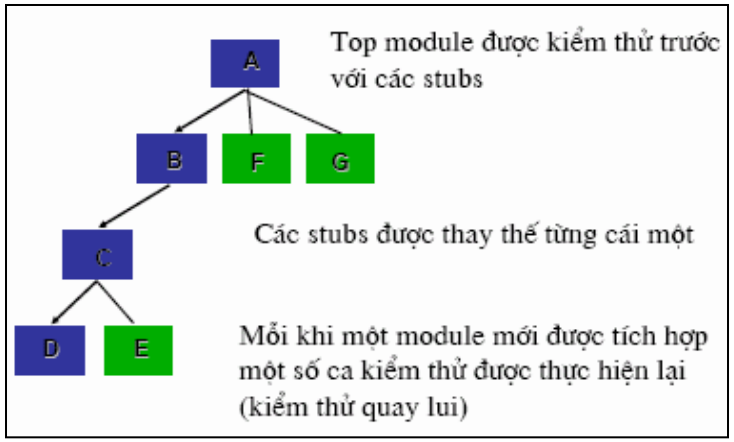
- ✓ Giai đoạn này được tiến hành sau khi đã hoàn tất công việc kiểm thử chức năng cho từng mô-đun riêng lẻ bằng cách tích hợp các mô-đun này lại với nhau.
- ✓ Mục đích của giai đoạn này là kiểm tra giao diện của các đơn vị và sự kết hợp hoạt động giữa chúng, kiểm tra tính đúng đắn so với đặc tả, kiểm tra tính hiệu quả, với phương pháp thực hiện chủ yếu sử dụng kiểm tra chức năng dựa trên chiến lược từ trên xuống, hoặc từ dưới lên.

7.4.3. KIỂM THỬ TÍCH HỢP (INTEGRATION TEST)

Hướng xử lý Trên Xuống (Top-Down)

Thuật giải của hướng tiếp cận này gồm những bước sau:

- ✓ Sử dụng mô-đun chính như 1 driver và các stub được thay cho tất cả các mô-đun là con trực tiếp của mô-đun chính,
- ✓ Lần lượt thay thế các stub bởi các mô-đun thực sự,
- ✓ Tiến hành kiểm tra tính đúng đắn,
- ✓ Một tập hợp bộ thử nghiệm được hoàn tất khi hết stub,
- ✓ Kiểm tra lùi có thể được tiến hành để đảm bảo không phát sinh lỗi mới



7.4.3. KIỂM THỬ TÍCH HỢP (INTEGRATION TEST)

Ưu điểm:

- ✓ Kiểm thử trên xuống kết hợp với phát triển trên xuống sẽ giúp phát hiện sớm các lỗi thiết kế và làm giảm giá thành sửa đổi.
- ✓ Nhanh chóng có phiên bản thực hiện với các chức năng chính.
- ✓ Có thể thẩm định tính dùng được của sản phẩm sớm.

Nhược điểm:

- ✓ Nhiều mô-đun cấp thấp rất khó mô phỏng
- ✓ Thao tác với cấu trúc dữ liệu phức tạp, kết quả trả về phức tạp…

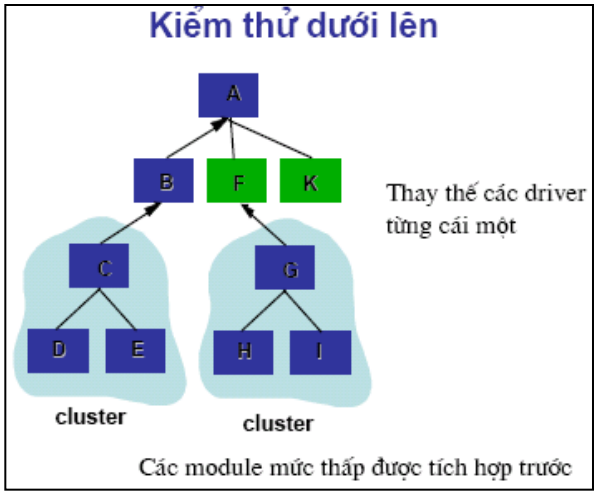
7.4.3. KIỂM THỬ TÍCH HỢP (INTEGRATION TEST)

Hướng xử lý Dưới Lên (Bottom-Up)

- ✓ Ta kiểm tra mô-đun lá (cấp thấp nhất) trước, do đó ta không cần phải viết stub.

Thuật giải của hướng này là:

- ✓ Các mô-đun cấp thấp được nhóm thành từng nhóm (thực hiện cùng chức năng),
- ✓ Viết driver điều khiển tham số nhập xuất,
- ✓ Bỏ driver và gắn chùm vào mô-đun cao hơn.



7.4.3. KIỂM THỬ TÍCH HỢP (INTEGRATION TEST)

Ưu điểm:

- ✓ Tránh xây dựng các mô-đun tạm thời phức tạp, tránh sinh các kết quả nhân tạo, thuận tiện cho phát triển các mô-đun để dùng lại.

Nhược điểm:

- ✓ Chậm phát hiện lỗi kiến trúc, chậm có phiên bản thực hiện

7.4.4. KIỂM THỬ HỆ THỐNG (SYSTEM TEST)

- ✓ Đến giai đoạn này, công việc kiểm thử được tiến hành với nhìn nhận phần mềm như là một yếu tố trong một hệ thống thông tin phức tạp hoàn chỉnh.
- ✓ Công việc kiểm thử nhằm kiểm tra khả năng phục hồi sau lỗi, độ an toàn, hiệu năng và giới hạn của phần mềm

7.4.5. KIỂM THỬ CHẤP NHẬN (ACCEPTANCE TEST)

- ✓ Kiểm thử chấp nhận được tiến hành bởi khách hàng, còn được gọi là *alpha testing*.
- ✓ Mục đích là nhằm thẩm định lại xem phần mềm có những sai sót, thiếu sót so với yêu cầu người sử dụng không.
- ✓ Trong giai đoạn này *dữ liệu dùng để kiểm thử do người sử dụng cung cấp*.

7.4.6. KIỂM THỬ BETA

- ✓ Đây là giai đoạn mở rộng của alpha testing.
- ✓ Khi đó, việc kiểm thử được thực hiện bởi *một số lượng lớn* người sử dụng.
- ✓ Công việc kiểm thử được tiến hành một cách ngẫu nhiên mà không có sự hướng dẫn của các nhà phát triển.
- ✓ Các lỗi nếu được phát hiện sẽ được thông báo lại cho nhà phát triển

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

Một số công việc khác trong quá trình kiểm thử:

- ✓ *Báo cáo kết quả kiểm thử*: Sau khi hoàn tất kiểm thử, nhóm kiểm tra tạo ra các số liệu và báo cáo cuối cùng về nỗ lực kiểm thử của họ và cho biết phần mềm có sẵn sàng để phát hành hay không.
- ✓ *Phân tích kết quả kiểm thử hoặc phân tích thiếu sót*: được thực hiện bởi nhóm phát triển kết hợp với khách hàng để đưa ra quyết định xem những thiếu sót gì cần phải được chuyển giao, cố định và từ bỏ (tức là tìm ra được phần mềm hoạt động chính xác) hoặc giải quyết sau.
- ✓ *Kiểm tra lại khiếm khuyết*: Khi một khiếm khuyết đã được xử lý bởi đội ngũ phát triển, nó phải được kiểm tra lại bởi nhóm kiểm thử.

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử hồi quy*: ta thường xây dựng chương trình kiểm thử nhỏ là tập hợp các bài kiểm tra cho mỗi tích hợp mới, sửa chữa hoặc cố định phần mềm, để đảm bảo rằng những cung cấp mới nhất đã không phá hủy bất cứ điều gì và toàn bộ phần mềm vẫn còn hoạt động một cách chính xác.
- ✓ *Kiểm thử đóng gói*: mỗi phép thử thỏa mãn các chỉ tiêu truy xuất và thu được những kết quả quan trong như: bài học kinh nghiệm, kết quả, các bản ghi, tài liệu liên quan được lưu trữ và sử dụng như một tài liệu tham khảo cho các dự án trong tương lai.

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử tự động hóa*: Nhiều nhóm lập trình càng ngày càng dựa vào phương pháp kiểm thử tự động, đặc biệt là các nhóm sử dụng mô hình TDD (*Test Drive Development*). Có rất nhiều khung phần mềm được viết bên trong và mã trong mỗi phiên bản được chạy kiểm thử tự động mọi lúc khi tích hợp liên tục phần mềm. Tuy kiểm thử tự động không thể sao chép tất cả mọi thứ như con người có thể làm nhưng nó có thể rất hữu ích cho việc kiểm thử hồi quy. Tuy nhiên, nó cũng đòi hỏi phải có những kịch bản phát triển tốt để tiến hành kiểm thử.

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

Kiểm thử chức năng và phi chức năng:

- ✓ *Kiểm thử chức năng* nhằm trả lời câu hỏi "người dùng có hay không làm được với tính năng cụ thể này" (xem Phương pháp Hộp đen)
- ✓ *Kiểm thử phi chức năng* đề cập đến các khía cạnh của phần mềm có thể không liên quan đến một chức năng cụ thể hoặc hành động người dùng, chẳng hạn như khả năng mở rộng và hiệu suất khác, hành vi dưới những hạn chế hoặc bảo mật nhất định. Việc kiểm thử sẽ xác định điểm cuộn mà tại đó khả năng mở rộng và thực hiện của các điểm cực trị hoạt động không ổn định. Những yêu cầu phi chức năng thường là những phản ánh về chất lượng của sản phẩm, đặc biệt là trong bối cảnh các quan điểm phù hợp của người sử dụng

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử sự phá hủy*: cố gắng làm hỏng phần mềm hoặc một hệ thống con.
- ✓ Nó xác minh rằng các phần mềm có chức năng đúng ngay cả khi nó nhận được đầu vào không hợp lệ hoặc không mong muốn, do đó tạo ra sự vững mạnh của xác nhận đầu vào và thói quen quản lý các lỗi.
- ✓ Chèn lỗi phần mềm ở dạng mờ nhạt là một ví dụ về kiểm thử thất bại.
- ✓ Các công cụ kiểm thử phi chức năng thương mại được liên kết từ các trang chèn lỗi phần mềm mà ở đó có sẵn vô số các mã nguồn mở và các công cụ miễn phí để thực hiện kiểm thử sự phá hủy phần mềm.

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử hiệu suất phần mềm*: thường được chạy để xác định một hệ thống hay hệ thống con thực hiện như thế nào về độ nhạy và tính ổn định theo một khối lượng công việc cụ thể. Nó cũng có thể dùng để điều tra, đánh giá, xác nhận hoặc xác minh các thuộc tính chất lượng khác của hệ thống, chẳng hạn như khả năng mở rộng, độ tin cậy và sử dụng tài nguyên. Dạng kiểm thử này bao gồm một số dạng con như:

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử lượng tải* chủ yếu liên quan đến việc kiểm thử hệ thống có thể tiếp tục hoạt động dưới một lượng tải cụ thể, cho dù đó là một lượng lớn dữ liệu hoặc một số lượng lớn người sử dụng. Điều này thường được gọi là khả năng mở rộng phần mềm. Các hoạt động kiểm thử lượng tải có liên quan khi thực hiện như một hoạt động phi chức năng thường được gọi là kiểm thử sức chịu đựng.
- ✓ *Kiểm tra khối lượng* là một cách để kiểm tra các chức năng của phần mềm ngay cả khi một số thành phần (ví dụ như một tập tin hoặc cơ sở dữ liệu) tăng triệt để kích thước. Kiểm thử độ căng là một cách để kiểm tra độ bền đột xuất hoặc ít gặp theo khối lượng công việc.

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử tính ổn định* (thường được tham chiến lượng tải và kiểm thử độ bền) giúp kiểm tra xem phần mềm có thể hoạt động tốt liên tục trong hoặc trên một chu kỳ chấp nhận được.
- ✓ Có rất ít quy ước về các mục tiêu cụ thể của kiểm thử hiệu suất như là: Các thuật ngữ lượng tải, kiểm thử hiệu suất, kiểm thử tính mở rộng và kiểm thử khối lượng, thường được sử dụng thay thế cho nhau.
- ✓ Hệ thống phần mềm thời gian thực có những ràng buộc chính xác thời gian.
- ✓ Để kiểm thử những ràng buộc thời gian được đáp ứng thì người ta dùng phương pháp *kiểm thử thời gian thực*.

7.4.7. MỘT SỐ CÔNG VIỆC KHÁC

- ✓ *Kiểm thử tính khả dụng*: là rất cần thiết để kiểm tra xem giao diện có tiện dụng và dễ hiểu với người dùng không, nó liên quan trực chủ yếu đến năng lực sử dụng của phần mềm.
- ✓ *Kiểm thử khả năng tiếp cận*: chú ý việc tuân thủ các tiêu chuẩn sau:
- ✓ Đạo luật bất khả thi của Mỹ năm 1990
- ✓ Mục 508 sửa đổi của đạo luật Phục hồi năm 1973.
- ✓ Sáng kiến tiếp cận Web (WAI) của W3C.
- ✓ *Kiểm thử bảo mật*: rất cần thiết trong việc xử lý dữ liệu mật và ngăn chặn các hacker xâm nhập hệ thống.

7.5. VÍ DỤ MINH HOẠ

Ví dụ: Xét phần mềm QL Thư viện trong giai đoạn kiểm thử.

Giai đoạn 6: Kiểm chứng phần mềm hướng đối tượng

Kiểm tra tính đúng đắn của các lớp đối tượng

Chuẩn bị dữ liệu thử nghiệm: Nhập dữ liệu thử nghiệm cho các bảng THU_VIEN, SACH, DOC_GIA, MUON_SACH

Kiểm tra:

- ✓ Kiểm tra từng lớp đối tượng:
- ✓ Kiểm tra lớp THU_VIEN (Tra cứu độc giả, Tra cứu sách)
- ✓ Kiểm tra lớp DOC_GIA (Lập thẻ, cho mượn sách)
- ✓ Kiểm trả lớp SACH (Nhận sách, Trả sách)

7.5. VÍ DỤ MINH HOẠ

- ✓ Kiểm tra phối hợp các lớp đối tượng
- ✓ Kiểm tra phối hợp giữa lớp THU_VIEN và lớp DOC_GIA (Lập thẻ và sau đó tra cứu độc giả)
- ✓ Kiểm tra phối hợp giữa lớp THU_VIEN và lớp SACH (Nhận sách và sau đó tra cứu sách)
- ✓ Kiểm tra phối hợp giữa lớp DOC_GIA và lớp SACH (Lập thẻ, Nhận sách, Cho mượn sách và Tra sách)
- ✓ Kiểm tra phối hợp giữa 3 lớp THU_VIEN, DOC_GIA và lớp SACH
- ✓ Xác nhận của khách hàng: Khách hàng sử dụng phần mềm để thực hiện các công việc của mình và so sánh kết quả khi sử dụng phần mềm với kết quả khi thực hiện trong thế giới thực

TÓM TẮT

Giới thiệu các kiến thức về:

- ✓ Mô hình chữ V
- ✓ Các yêu cầu đối với kiểm thử
- ✓ Các kỹ thuật kiểm thử
- ✓ Phương pháp Hộp đen (Kiểm thử chức năng)
- ✓ Phương pháp Hộp trắng (Kiểm thử cấu trúc)
- ✓ Chiến lược kiểm thử & các giai đoạn
- ✓ Kiểm thử Đơn vị (Unit Test)
- ✓ Kiểm thử Chức năng (Functional Test)
- ✓ Kiểm thử Tích hợp (Integration Test)
- ✓ Kiểm thử Hệ thống (System Test)
- ✓ Kiểm thử Chấp nhận (Acceptance Test)
- ✓ Kiểm thử beta

BÀI TẬP

1. Phụ lục A trang 170
2. Phụ lục B trang 179