



KIỂM THỬ PHẦN MỀM

Lê thị Mỹ Hạnh



NỘI DUNG

- Giới thiệu sơ lược về Testing
- Thiết kế trường hợp kiểm thử
- Các kỹ thuật kiểm thử
- Các chiến lược kiểm thử
- Giới thiệu một vài công cụ kiểm thử

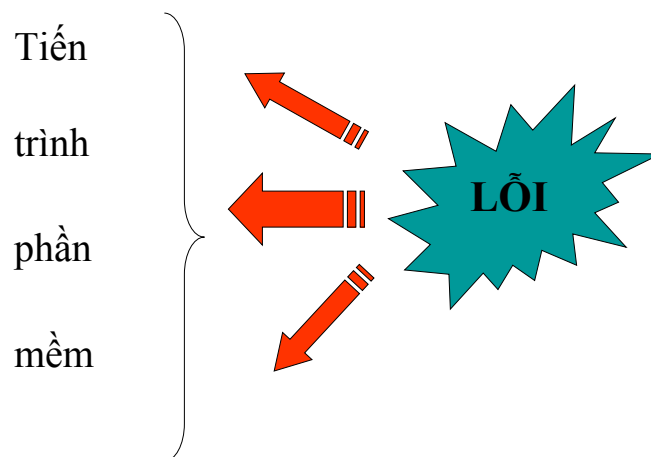
NỘI DUNG

■ Giới thiệu sơ lược về Kiểm thử

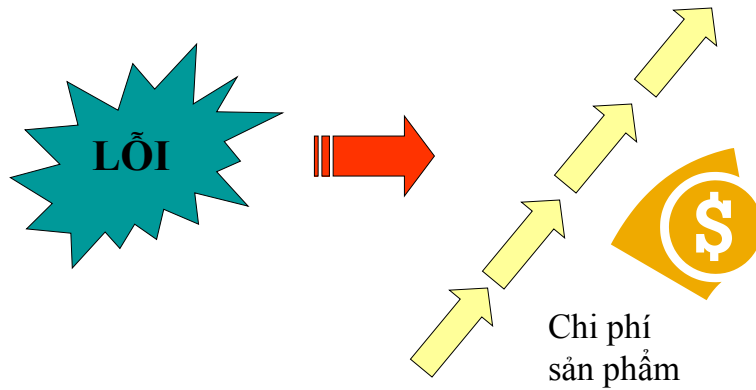
- Tại sao ta lại cần đến kiểm thử?
- Kiểm thử là gì?
- Mục đích của kiểm thử.
- Vị trí của kiểm thử trong tiến trình phần mềm.
- Ai sẽ là người kiểm thử?
- Các nguyên tắc cơ bản khi kiểm thử.
- Thế nào là một kiểm thử tốt?

■ Thiết kế trường hợp kiểm thử

TẠI SAO TA LẠI CẦN ĐẾN KIỂM THỬ



TẠI SAO TA LẠI CẦN ĐẾN KIỂM THỬ?



➡ Lí do tại sao chúng ta lại cần đến kiểm thử!

KIỂM THỬ PHẦN MỀM LÀ GÌ?

- Kiểm thử là một tiến trình thực hiện chương trình với mục đích duy nhất là tìm ra lỗi.
- Là tiến trình (và là nghệ thuật) nhằm phát hiện lỗi bằng việc xem xét lại đặc tả, thiết kế và mã hoá.
- Là mấu chốt của đảm bảo chất lượng phần mềm
- Một trường hợp kiểm thử tốt là trường hợp có xác suất cao để tìm ra lỗi còn chưa lộ ra.
- Một kiểm thử thành công là việc kiểm thử làm lộ ra lỗi còn chưa được xuất hiện.

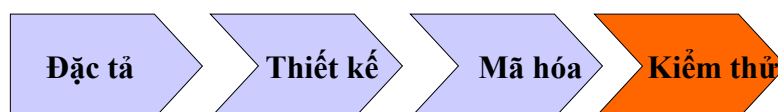
MỤC ĐÍCH CỦA KIỂM THỬ LÀ GÌ?

- Làm lộ ra một cách có hệ thống những lớp lỗi khác nhau (với thời gian và công sức tối thiểu)
- Đảm bảo các thành phần của ứng dụng là ăn khớp với nhau, vận hành như mong đợi và phù hợp với các chuẩn thiết kế.

→ *Kiểm thử chỉ mang tính chất phát hiện ra được rằng tại thời điểm đó chỉ phát hiện ra bấy nhiêu lỗi thôi.*



VỊ TRÍ CỦA KIỂM THỬ TRONG TIẾN TRÌNH PHẦN MỀM



AI SẼ LÀ NGƯỜI KIỂM THỬ?



Nhóm phát triển



Nhóm kiểm thử độc lập

CÁC LOẠI LỖI CÓ THỂ XẢY RA

- Sai: sản phẩm xây dựng khác với đặc tả
 - Thiếu: sản phẩm xây dựng không đáp ứng hết các yêu cầu trong đặc tả
 - Thừa: Yêu cầu có trong sản phẩm nhưng không có trong đặc tả
 - Ngoài ra, một ứng dụng khó sử dụng, chậm chạp cũng được xem là lỗi.
-

6 điểm 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
-

6 điểm lưu ý khi kiểm thử (tiếp)

- (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 trước đó để tránh ảnh hưởng lan truyền sóng
-

CÁC NGUYÊN TẮC CƠ BẢN KHI KIỂM THỬ

- Tất cả các trường hợp kiểm thử phải được hướng theo yêu cầu của khách hàng.
 - Kiểm thử phải được lên kế hoạch trong thời gian dài trước khi bắt đầu kiểm thử.
 - Sẽ là điều không thể nếu như muốn tìm ra hết lỗi.
 - Kiểm thử nên bắt đầu từ cái nhỏ cho đến những cái lớn hơn
 - Để có thể đạt được hiệu quả cao nhất, kiểm thử phải được tiến hành bởi nhóm thứ ba độc lập.
-

THẾ NÀO LÀ MỘT KIỂM THỬ TỐT?

- Một kiểm thử tốt có khả năng cao trong việc tìm kiếm ra lỗi.
 - Một kiểm thử tốt sẽ không bao giờ là dư thừa.
 - Kiểm thử cũng không nên quá đơn giản, cũng không nên quá phức tạp.
-

THIẾT KẾ TRƯỜNG HỢP KIỂM THỬ

- Các phương pháp kiểm thử ra đời cung cấp:
 - Cách tiếp cận đến việc kiểm thử.
 - Cơ chế đảm bảo tính đầy đủ cho kiểm thử
 - Các khả năng cao nhất để phát hiện lỗi trong phần mềm.
 - Có 2 phương pháp kiểm thử sản phẩm phần mềm:
 - Kiểm thử tĩnh
 - Kiểm thử động
-

Kiểm thử tĩnh

- Kiểm thử trên bàn hay Kiểm thử tĩnh: 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)
-

Kiểm thử động

- 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
 - 9 bước của trình tự kiểm thử bằng máy
-

Trình tự kiểm thử bằng máy

- (1) Thiết kế trường hợp kiểm thử trên bàn
 - (2) Trường hợp thử phải có cả kết quả kỳ vọng sẽ thu được
 - (3) Dịch chương trình nguồn và tạo môđun tải để thực hiện
 - (4) Khi trường hợp thử có xử lý tệp vào-ra, phải làm trước trên bàn việc xác định miền của các tệp
-

Trình tự kiểm thử bằng máy (tiếp)

- (5) Nhập dữ liệu đã thiết kế cho trường hợp kiểm thử
 - (6) Điều chỉnh môi trường thực hiện môđun tải (tạo thủ tục đưa các tệp truy cập tệp vào chương trình)
 - (7) Thực hiện môđun tải và ghi nhận kết quả
 - (8) Xác nhận kết quả với kết quả kỳ vọng
 - (9) Lặp lại thao tác (5)-(8)
-

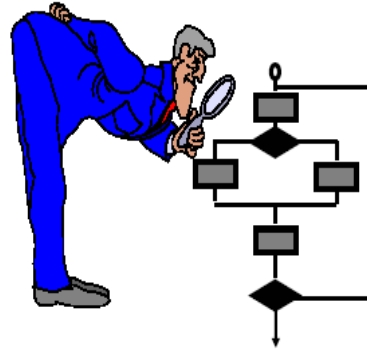
Kỹ thuật thiết kế trường hợp kiểm thử

- Kiểm thử hộp đen (hướng dữ liệu): Kiểm thử các chức năng của sản phẩm có vận hành hoàn toàn không?
 - Kiểm thử hộp trắng (hướng cấu trúc): Kiểm thử tất cả các cấu trúc của chương trình. (ví dụ: kiểm tra các dòng lệnh...)
-

KIỂM THỬ HỘP TRẮNG

■ **Định nghĩa:**

Kiểm thử hộp trắng còn được gọi là kiểm thử hộp kính là 1 quá trình duyệt tất cả các cấu trúc điều khiển của tất cả các sơ đồ trong chương trình nguồn nhằm đưa ra các điều không hợp lý của cấu trúc điều khiển



VÌ SAO PHẢI KIỂM THỬ HỘP TRẮNG

- Các lỗi logic và các giả thiết không đúng tỉ lệ nghịch với xác suất một đường dẫn trong chương trình sẽ được thực thi
- Chúng ta thường tin rằng một đường dẫn logic thì không thể nào được thực hiện khi trong thực tế nó có thể được thực hiện trên cơ sở đều đặn
- Lỗi gõ vào là ngẫu nhiên

MỤC ĐÍCH KIỂM THỬ HỘP TRẮNG

- Biết quá trình hoạt động bên trong của một sản phẩm
 - Đảm bảo các thành phần bên trong vận hành một cách đúng đắn
 - Các thao tác bên trong là chính xác
-

NGUYÊN TẮC KIỂM THỬ HỘP TRẮNG

- Đảm bảo tất cả các đường dẫn độc lập bên trong một môđun đều được thực hiện ít nhất 1 lần
 - Thực hiện tất cả các quyết định logic trên cả nhánh đúng và sai
 - Thực hiện tất cả các vòng lặp tại các đường biên và bên trong đường biên
 - Thực hiện tất cả các cấu trúc dữ liệu bên trong để đảm bảo tính hợp lệ của chúng
-

CÁC TRƯỜNG HỢP KIỂM THỬ HỘP TRẮNG

- 1 Kiểm thử đường dẫn cơ sở
- 2 Kiểm thử điều kiện
- 3 Kiểm thử vòng lặp

KIỂM THỬ ĐƯỜNG DẪN CƠ SỞ

- Cho phép người kiểm thử thực hiện các trường hợp kiểm thử để đảm bảo mọi câu lệnh trong chương trình được thực hiện ít nhất 1 lần
- Trong quá trình kiểm thử người kiểm thử cần thực hiện theo các bước sau:

B1: Vẽ đồ thị lưu trình

B2: Xác định độ phức tạp Cyclomat

Tìm C bằng 1 trong 3 cách sau:

+ $C = \text{Số lượng vòng}$

+ $C = \text{số cạnh} - \text{số đỉnh} + 2$

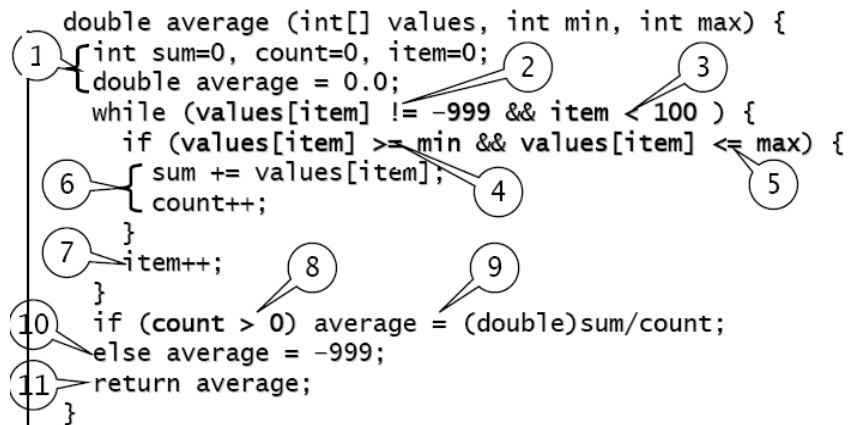
+ $C = \text{Số đỉnh điều kiện} + 1$

B3: Xác định 1 tập các đường dẫn cơ sở độc lập

B4: Thiết kế các trường hợp kiểm thử với mọi đường dẫn độc lập

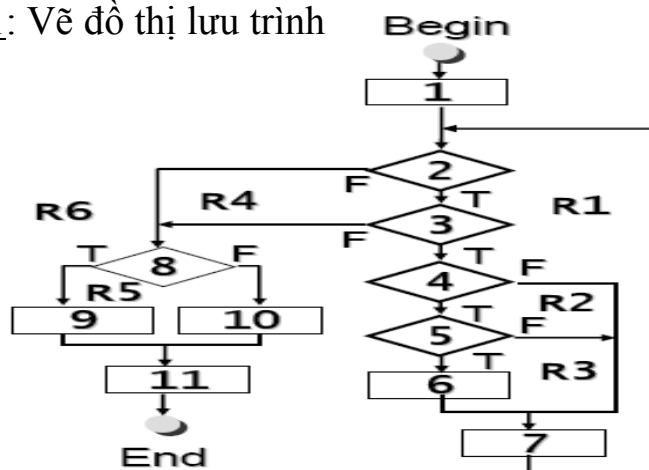
KIỂM THỬ ĐƯỜNG DẪN CƠ SỞ

■ Ví dụ:



KIỂM THỬ ĐƯỜNG DẪN CƠ SỞ

■ B1: Vẽ đồ thị lưu trình



KIỂM THỬ ĐƯỜNG DẪN CƠ SỞ

B2: Xác định độ phức tạp Cyclomat(C)

+ $C = \text{Số lượng vòng} = 6$

+ Hoặc $C = \text{số cạnh} - \text{số đỉnh} + 2 = 15 - 11 + 2 = 6$

+ Hoặc $C = \text{Số đỉnh điều kiện} + 1 = 5 + 1 = 6$

KIỂM THỬ ĐƯỜNG DẪN CƠ SỞ

B3: Xác định các đường dẫn độc lập

■ Dựa vào độ phức tạp đưa ra đường dẫn độc lập

■ Có 6 đường dẫn độc lập:

Path1: $1 \rightarrow 2 \rightarrow 8 \rightarrow 9 \rightarrow 11$

Path2: $1 \rightarrow 2 \rightarrow 8 \rightarrow 10 \rightarrow 11$

Path3: $1 \rightarrow 2 \rightarrow 3 \rightarrow 8 \rightarrow 9 \rightarrow 11$

Path4: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 2 \dots$

Path5: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 2 \dots$

Path6: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 2 \dots$

KIỂM THỬ ĐƯỜNG DẪN CƠ SỞ

- B4: Thiết kế trường hợp kiểm thử cho từng đường dẫn:

Test Case1 cho Path1:

Đầu vào: $a=\{1, 7, -999, 37\}$, $\min=0$, $\max=100$

Đầu ra mong đợi: $\text{avg}=(1+7)/2$

Test Case2 cho Path2:

Đầu vào: $a=\{-999\}$, $\min=0$, $\max=100$

Đầu ra mong đợi: $\text{avg}=-999$

Test Case3 cho Path3:

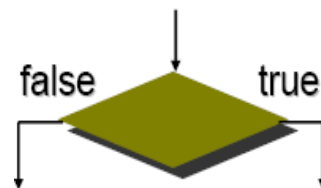
Đầu vào: $a=\{7, 5, \dots, 50\}$ (>100 phần tử), $\min=0$, $\max=100$

Đầu ra mong đợi: avg =giá trị trung bình của 100 phần tử đầu tiên

KIỂM THỬ ĐIỀU KIỆN

- Là phương pháp thiết kế trường hợp kiểm thử để thực hiện tất cả các điều kiện logic

Mục đích: Tìm ra những lỗi trong điều kiện và các lỗi khác



KIỂM THỬ ĐIỀU KIỆN

- Việc kiểm thử điều kiện sẽ tìm ra các lỗi sau:
 - + Lỗi toán tử logic
 - + Lỗi biến logic
 - + Lỗi của dấu ngoặc đơn
 - + Lỗi biểu thức quan hệ
 - + Lỗi của biểu thức số học
- Kiểm thử điều kiện bao gồm:
 - + Kiểm thử nhánh
 - + Kiểm thử miền
 - + Kiểm thử nhánh và toán tử quan hệ (BRO)

KIỂM THỬ NHÁNH/ KIỂM THỬ MIỀN

Kiểm tra nhánh

- Là chiến lược kiểm thử điều kiện đơn giản nhất. Với một điều kiện hợp thành C, các nhánh đúng và sai của C và mọi điều kiện đơn trong C cần thực hiện ít nhất 1 lần

Kiểm tra miền

- Đòi hỏi ba hay bốn phép kiểm thử được dẫn ra cho biểu thức quan hệ
- Với biểu thức quan hệ có dạng
 $E1 < \text{toán tử quan hệ} > E2$
 Thì cần ba phép thử cho giá trị $E1 >, =, < E2$
- Với 1 biểu thức có n biến thì sẽ cần tới tất cả 2^n phép kiểm thử có thể có ($n > 0$)
- Phương pháp này chỉ khả thi khi n nhỏ

KIỂM THỬ NHÁNH VÀ TOÁN TỬ QUAN HỆ (BRO)

- Kỹ thuật này đảm bảo việc phát hiện ra các lỗi toán tử quan hệ và nhánh trong một điều kiện.
- Chiến lược BRO dùng các điều kiện ràng buộc đối với điều kiện C. Một ràng buộc điều kiện trong C với n là điều kiện đơn được xác định là $(D1, D2, \dots, D_n)$.

■ Ví dụ 1: C1: $B1 \& B2$

Với B1, B2 là các biến boolean. C1 có dạng $(D1, D2)$ với mỗi D1, D2 là t hay f. Ta có tập ràng buộc sau:

$\{(t, t), (t, f), (f, t)\}$

Còn trường hợp (f, f) là dư thừa vì chỉ cần 1 trong 2 biến sai là biểu thức đã sai.

KIỂM THỬ NHÁNH VÀ TOÁN TỬ QUAN HỆ (BRO)

■ Ví dụ 2: C2: $B1 \& (E1 = E2)$

Với B1 là biến, còn E1, E2 là các biểu thức số học. Ràng buộc cho điều kiện C2 có dạng $(D1, D2)$ với D1 là t hay f, còn D2 là $<, >, =$. Theo như ví dụ 1, giả sử B2 là $(E1 == E2)$ thì tập ràng buộc sẽ là $\{(t, t), (t, f), (f, t)\}$

“t” cho trường hợp $(E1 == E2)$ là “=”

“f” cho trường hợp $(E1 == E2)$ là “>” hoặc “<”.

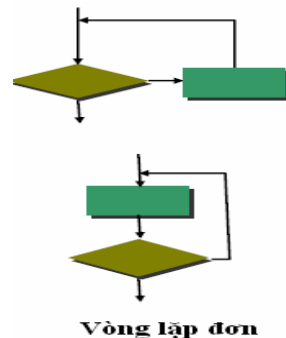
Tập ràng buộc C2 là $\{(t, =), (f, =), (t, <), (t, >)\}$

KIỂM THỬ VÒNG LẶP

- Bao gồm:
 - + Vòng lặp đơn
 - + Vòng lặp lồng nhau
 - + Vòng lặp nối tiếp
 - + Vòng lặp phi cấu trúc

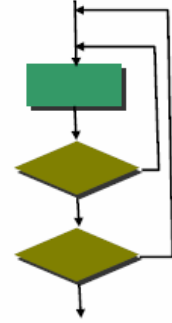
VÒNG LẶP ĐƠN

- Giả sử n là số lớn nhất các lần thực hiện của một vòng lặp đơn, có các trường hợp kiểm tra là:
 - + Nhảy qua toàn bộ vòng lặp
 - + Nhảy một bước qua vòng lặp
 - + Hai bước qua vòng lặp
 - + m bước qua vòng lặp với $m < n$



VÒNG LẶP LÔNG

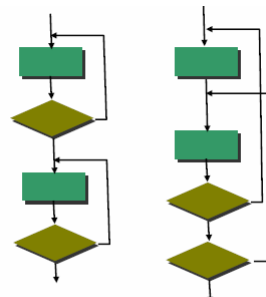
- Bắt đầu từ vòng lặp bên trong nhất
Đặt tất cả các vòng lặp khác
vào giá trị nhỏ nhất.
- Tiến hành các kiểm thử
vòng lặp đơn cho vòng lặp
bên trong nhất.
- Tiến dần ra ngoài, thực hiện
phép kiểm thử cho vòng lặp tiếp.
- Tiếp tục cho đến khi mọi vòng lặp đã được kiểm
thử hết.



Vòng lặp lồng nhau

CÁC VÒNG LẶP KHÁC

- Vòng lặp nối tiếp: Nếu vòng lặp là độc lập thì dùng
cách tiếp cận vòng lặp đơn. Nếu các vòng lặp là
không độc lập thì dùng cách tiếp cận là vòng lặp lồng
nhau.
- Vòng lặp phi cấu trúc:
Viết lại các vòng lặp trên



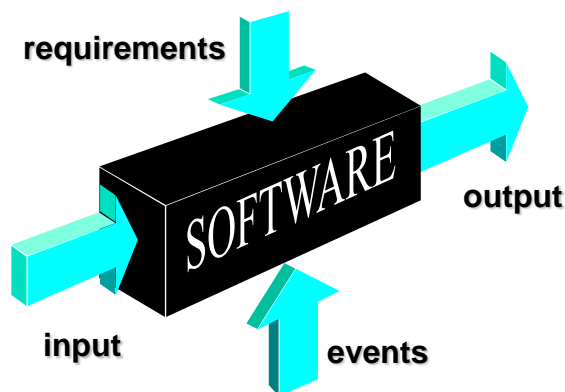
Vòng lặp nối tiếp

Vòng lặp phi
cấu trúc

NỘI DUNG TRÌNH BÀY

- **Giới thiệu về kiểm thử hộp đen**
 1. Kiểm thử hộp đen là gì?
 2. Mục tiêu và trọng điểm của kiểm thử hộp đen?
 - **Các phương pháp kiểm thử hộp đen**
 1. Phân hoạch tương đương (Equivalence Partitioning)
 2. Phân tích giá trị biên (Boundary Value Analysis)
 3. Kỹ thuật đồ thị nhân quả (Cause – Effect Graphics)
 4. Kiểm thử so sánh (Comparison Testing)
 5. Kiểm thử dựa trên nền đồ thị
 6. ...
-

GIỚI THIỆU VỀ KIỂM THỬ HỘP ĐEN



TẠI SAO PHẢI KIỂM THỬ HỘP ĐEN?

- Kiểm thử hộp đen không phải là phương án của kiểm thử hộp trắng, mà nó là phương pháp bổ sung cho kiểm thử hộp trắng về phương diện tìm ra những lớp lỗi khác mà phương pháp kiểm thử hộp trắng không tìm ra được.
 - Kiểm thử hộp đen được áp dụng trong các giai đoạn sau của kiểm thử. Vì kiểm thử hộp đen không chú ý xét đến cấu trúc điều khiển nên sự chú ý được dồn vào miền thông tin.
-

TRỌNG TÂM CỦA KIỂM THỬ HỘP ĐEN

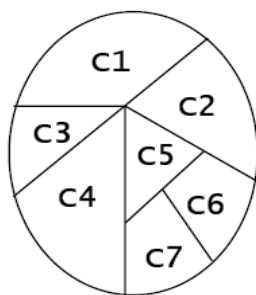
- Kiểm thử hộp đen cố gắng tìm lỗi trong các trường hợp sau:
 - ❑ Hoạt động không đúng hoặc thiếu chức năng
 - ❑ Lỗi truy cập cơ sở dữ liệu ngoài
 - ❑ Lỗi giao diện
 - ❑ Lỗi thực thi chương trình
 - ❑ Lỗi khởi tạo và kết thúc
 - ❑ Các lỗi khác
-

CÁC PHƯƠNG PHÁP KIỂM THỬ HỘP ĐEN

1. Phân hoạch tương đương (Equivalence Partitioning)
2. Phân tích giá trị biên (Boundary Value Analysis)
3. Kỹ thuật đồ thị nhân quả (Cause – Effect Graphics)
4. Kiểm thử so sánh (Comparison Testing)
5. Kiểm thử ngẫu nhiên (Random Testing)
6. ...

PHƯƠNG PHÁP PHÂN HOẠCH TƯƠNG ĐƯƠNG (Equivalence Partitioning)

■ Nội dung:



Miền đầu vào

- Chia các miền đầu vào của chương trình thành các lớp dữ liệu tương đương.
- Thiết kế 1 trường hợp kiểm thử đại diện cho mỗi lớp.
- Lớp tương đương biểu thị cho 1 tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện đầu vào

PHƯƠNG PHÁP PHÂN HOẠCH TƯƠNG ĐƯƠNG
(Equivalence Partitioning)

- Nguyên tắc :
 1. Nếu điều kiện đầu vào là 1 khoảng giá trị thì có 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ được xác định
 2. Nếu điều kiện đầu vào là giá trị cụ thể, 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ được xác định.
 3. Nếu điều kiện đầu vào là 1 tập hợp các giá trị, 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ được xác định.
 4. Nếu điều kiện đầu vào là 1 giá trị Boolean, 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ được xác định.

PHƯƠNG PHÁP PHÂN HOẠCH TƯƠNG ĐƯƠNG
(Equivalence Partitioning)

■ Ví dụ:

STT	Trường hợp của điều kiện đầu vào	Lớp tương đương	
		Hợp lệ	Không hợp lệ
1	Là 1 khoảng các giá trị	1	2
	Ví dụ: [3,10]	$3 < x < 10$	$x < 3$ hoặc $x > 10$
2	Là 1 giá trị cụ thể	1	2
	Ví dụ: 6	$x = 6$	$x < 6$ hoặc $x > 6$
3	Là 1 tập hợp các giá trị	1	1
	Ví dụ: $A = \{1,5,4,8,6\}$	x	
4	Là 1 giá trị Boolean	1	1

PHƯƠNG PHÁP PHÂN HOẠCH TƯƠNG ĐƯƠNG
(Equivalence Partitioning)

- **Ví dụ 1:** Một hệ thống tự động của ngân hàng đòi hỏi dữ liệu đầu vào như sau: ⇒ Dữ liệu đầu vào được phân hoạch như sau:

- User ID: 1 chuỗi lớn hơn hoặc bằng 6 kí tự
- Password: Gồm 6 chữ số, không được bắt đầu với số 0
- Command: tập các lệnh { kiểm tra tài khoản, gửi tiền, thanh toán hoá đơn }

Dữ liệu đầu vào			Lớp	Nhận xét
UserID	Tồn tại	>=6chữ số	1a	Hợp lệ
		<6 chữ số	1b	Không Hợp lệ
	Không tồn tại		2	Không hợp lệ
Password	Tồn tại	100000 → 999999	3a	Hợp lệ
		<100000	3b	Không Hợp lệ
		>999999	3c	Không Hợp lệ
	Không tồn tại		4	Không hợp lệ
Command	Thuộc tập các lệnh		5	Hợp lệ
	Không thuộc tập các lệnh		6	Không hợp lệ

PHƯƠNG PHÁP PHÂN HOẠCH TƯƠNG ĐƯƠNG
(Equivalence Partitioning)

- **Ví dụ 2:** Chương trình nhận vào ba số thực, kiểm tra ba số thực có là độ dài ba cạnh một tam giác. Nếu là độ dài ba cạnh của một tam giác thì kiểm tra xem nó là tam giác nhọn, vuông hay tù.

	Nhọn	Vuông	Tù
Thường	6, 5, 3	3, 4, 5	5, 6, 10
Cân	6, 2, 6	√2, 2, √2	7, 4, 4
Đều	4, 4, 4	Không thể	Không thể
Không là tam giác	-2, 3, 8		

PHƯƠNG PHÁP PHÂN TÍCH GIÁ TRỊ BIÊN (Boundary Values Analysis - BVA)



■ Nội dung:

- BVA là 1 kỹ thuật thiết kế trường hợp kiểm thử
- BVA bổ sung cho phân hoạch tương đương.
 - BVA chọn 1 hoặc nhiều phần tử tại mỗi biên của lớp tương đương
 - BVA suy dẫn các trường hợp kiểm thử từ miền cái ra.

PHƯƠNG PHÁP PHÂN TÍCH GIÁ TRỊ BIÊN (Boundary Values Analysis - BVA)

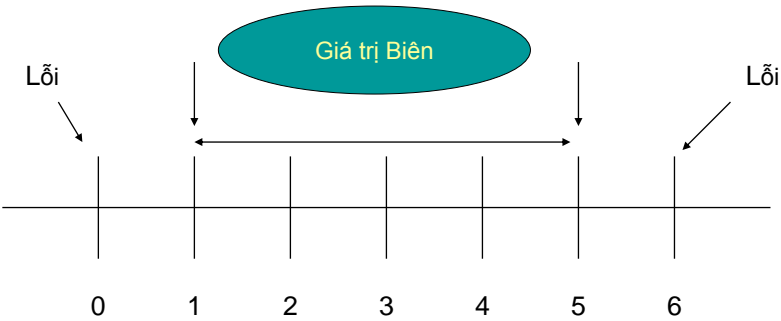
■ Nguyên tắc:

1. Nếu điều kiện đầu vào là 1 khoảng giữa a và b, trường hợp kiểm thử bao gồm cả a và b, giá trị cận trên và cận dưới a và b
 Ví dụ: - Số nguyên D với điều kiện đầu vào [3,10]
 Giá trị kiểm thử: 2,3,4,9,10,11
2. Nếu điều kiện đầu vào là 1 tập hợp các giá trị, trường hợp kiểm thử bao gồm các giá trị cực đại, cực tiểu, giá trị cận trên và cận dưới của giá trị cực đại và cực tiểu cũng được kiểm thử
 Ví dụ: - Dữ liệu E được liệt kê {3,5,100,102}
 Giá trị kiểm thử: 3,102,2,103
3. Áp dụng nguyên tắc 1 và 2 cho điều kiện đầu ra.
4. Nếu cấu trúc dữ liệu bên trong chương trình đã chỉ định rõ các biên, trường hợp kiểm thử nên được thiết kế để kiểm tra tại biên đó.
 Ví dụ: 1 mảng gồm 100 phần tử

PHƯƠNG PHÁP PHÂN TÍCH GIÁ TRỊ BIÊN
(Boundary Values Analysis – BVA)

Ví dụ 1: Một ngân hàng cho vay vốn, lãi suất và thế chấp được quy định như sau:

- thế chấp được phép trong khoảng từ 1 đến 5 căn nhà
- Lãi suất tương ứng hàng tháng từ 1000\$ đến 83,333 \$




PHƯƠNG PHÁP PHÂN TÍCH GIÁ TRỊ BIÊN
(Boundary Values Analysis – BVA)

Lãi Xuất	Thế chấp	Kết quả	Mô tả
1000\$	1	Hợp lệ	Lãi suất Min - Thế chấp Min
83333\$	1	Hợp lệ	Lãi xuất Max - thế chấp Min
1000\$	5	Hợp lệ	Lãi suất Min - Thế chấp Max
83333\$	5	Hợp lệ	Lãi xuất Max - thế chấp Max
1000\$	0	Không hợp lệ	Lãi suất Min - < Thế chấp Min
83333\$	0	Không hợp lệ	Lãi xuất Max - < thế chấp Min
1000\$	6	Không hợp lệ	Lãi suất Min - > Thế chấp Max
83333\$	6	Không hợp lệ	Lãi xuất Max - > thế chấp Max
999\$	1	Không hợp lệ	< Lãi suất Min - Thế chấp Min
83334\$	1	Không hợp lệ	> Lãi xuất Max - > Thế chấp Min
999\$	5	Không hợp lệ	< Lãi suất Min - Thế chấp Max
83334\$	5	Không hợp lệ	> Lãi xuất Max - > Thế chấp Max

PHƯƠNG PHÁP PHÂN HOẠCH TƯƠNG ĐƯƠNG
(Equivalence Partitioning)

■ Ví dụ 2: Bài toán tam giác

	1, 1, 2	Không là tam giác
	0, 0, 0	Chỉ là một điểm
	4, 0, 3	Một cạnh bằng không
	1, 2, 3.00001	Gần là một tam giác
	0.001, 0.001, 0.001	Tam giác rất nhỏ
	99999, 99999, 99999	Tam giác rất lớn
	3.00001, 3, 3	Tam giác gần đều
	2.99999, 4, 3	Tam giác gần cân
	3, 4, 5.00001	Tam giác gần vuông
	3, 4, 5, 6	Nhập vào bốn giá trị
	3	Chỉ nhập một giá trị
		Giá trị rỗng
	-3, 4, 5	Giá trị âm

Bài tập

- Một chương trình đọc vào một tệp văn bản và thực hiện một số xử lý sau: xóa các dấu cách dư thừa, đếm và in ra tổng số từ, chỉ ra sự hiện diện của ít nhất một từ có độ dài lớn hơn 20. Chương trình in ra văn bản, sau khi xử lý, theo từng dòng, mỗi dòng nhiều nhất là 80 ký tự. Các ký tự đọc vào của văn bản là các ký tự trong bảng chữ cái (từ a đến z) và các dấu cách. Một từ được phân cách với từ khác bởi một hoặc nhiều dấu cách.
 - Xây dựng tập dữ liệu thử cho chương trình trên bởi kỹ thuật **kiểm thử giá trị biên**.
 - Xây dựng tập dữ liệu thử cho chương trình trên bởi kỹ thuật **kiểm thử lớp tương đương**.

KỸ THUẬT ĐỒ THỊ NHÂN QUẢ (Cause – Effect Graphics)

■ **Nội dung:**

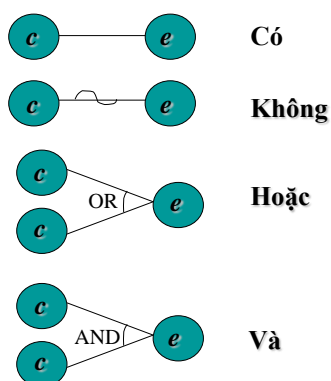
- Là 1 kỹ thuật thiết kế trường hợp kiểm thử cung cấp cách biểu diễn chính xác cho các điều kiện logic và hành động tương ứng.

■ **Nguyên tắc:** Tuân theo 4 bước

1. Liệt kê nguyên nhân và kết quả cho các thành phần phần mềm.
+ Nguyên nhân: biểu diễn cho điều kiện đúng của đầu vào hoặc kết hợp các đầu vào.
+ Kết quả: như 1 biểu thức boolean, biểu diễn cho 1 kết quả hoặc các kết quả được kết hợp.
2. Xây dựng đồ thị nguyên nhân - kết quả
3. Đồ thị được chuyển thành bảng quyết định
4. Các qui tắc của bảng quyết định được chuyển thành trường hợp kiểm thử

KỸ THUẬT ĐỒ THỊ NHÂN QUẢ (Cause – Effect Graphics)

■ Mối quan hệ logic giữa nguyên nhân (c) và kết quả (e)



KĨ THUẬT ĐỒ THỊ NHÂN QUẢ

(Cause – Effect Graphics)

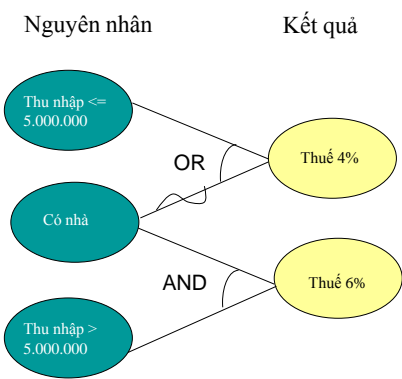
■ Ví dụ: Đề tính thuế

- Nguyên nhân và kết quả như sau:
- + Vô gia cư: thuế 4%
 - + Có nhà: - Nếu thu nhập <= 5.000.000 : thuế 4%
- Nếu thu nhập > 5.000.000 : thuế 6%

KĨ THUẬT ĐỒ THỊ NHÂN QUẢ

(Cause – Effect Graphics)

2. Xây dựng đồ thị nhân quả
4. Lập bảng quyết định



NN	Thu nhập <= 5.000.000	Thu nhập > 5.000.000
Vô gia cư	X	-
Có nhà	X	X
Thuế 4%	Y	-
Thuế 6%	-	Y

CÁC PHƯƠNG PHÁP KIỂM THỬ HỢP ĐƠN KHÁC

- Kiểm thử so sánh
 - Kiểm thử ngẫu nhiên
 - Kiểm thử dựa trên nền đồ thị
 - Đoán lỗi
-

CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM

- Cách tiếp cận chiến lược tới kiểm thử phần mềm.
 - Kiểm thử đơn vị.
 - Kiểm thử tích hợp.
 - Kiểm thử hợp lệ
 - Kiểm thử hệ thống
 - Tiến trình gỡ lỗi
-

CÁCH TIẾP CẬN CHIẾN LƯỢC TỚI KIỂM THỬ PHẦN MỀM

- Kiểm thử là một tập hợp những hoạt động có thể được lập kế hoạch trước và tiến hành một cách có hệ thống.
 - Các đặc trưng chung của một chiến lược kiểm thử (CLKT):
 - Việc kiểm thử bắt đầu tại mức mô đun và làm việc “hướng ra ngoài”.
 - Các kỹ thuật kiểm thử khác nhau được sử dụng tại các thời điểm khác nhau.
 - Việc kiểm thử được tiến hành bởi người phát triển phần mềm và một nhóm kiểm thử độc lập
 - Việc kiểm thử và gỡ lỗi là những hoạt động khác nhau nhưng gỡ lỗi phải phù hợp với bất kì chiến lược kiểm thử nào.
-

CÁCH TIẾP CẬN CHIẾN LƯỢC TỚI KIỂM THỬ PHẦN MỀM (tt)

- Kiểm chứng và hợp lệ hóa
 - Tổ chức việc kiểm thử phần mềm
 - Chiến lược kiểm thử phần mềm
 - Tiêu chuẩn hoàn thành kiểm thử
-

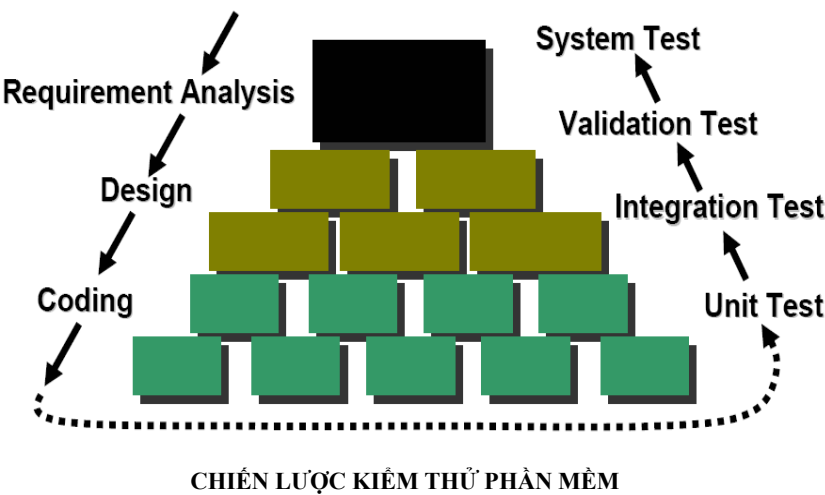
KIỂM CHỨNG VÀ HỢP LỆ HOÁ (V&V)

- Kiểm chứng: "Chúng ta có làm ra sản phẩm đúng không?"
Are you building the right product?
 - Hợp lệ hóa: "Chúng ta có làm ra đúng sản phẩm không?"
Are you building the product right?
-

TỔ CHỨC VIỆC KIỂM THỬ PHẦN MỀM

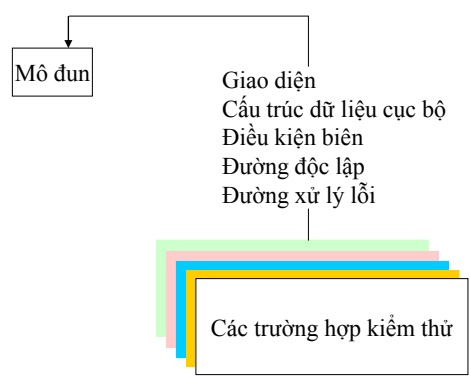
- Người phát triển phần mềm bao giờ cũng có trách nhiệm kiểm thử riêng các đơn vị (mô đun) chương trình. Trong nhiều trường hợp người phát triển cũng tiến hành cả kiểm thử tích hợp.
 - Có thể dùng một nhóm kiểm thử độc lập để làm công việc kiểm thử (ITG)
-

CHIẾN LƯỢC KIỂM THỬ PHẦN MỀM



KIỂM THỬ ĐƠN VỊ

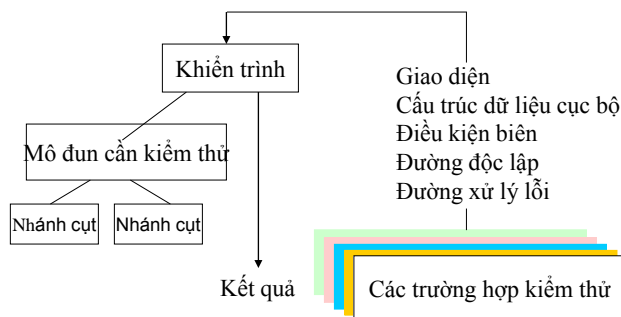
■ Các xem xét kiểm thử đơn vị



KIỂM THỬ ĐƠN VỊ

KIỂM THỬ ĐƠN VỊ (tt)

- Thủ tục kiểm thử đơn vị:
 - Được xét như một bước chuyển sang bước mã hóa



MÔI TRƯỜNG KIỂM THỬ ĐƠN VỊ

KIỂM THỬ TÍCH HỢP

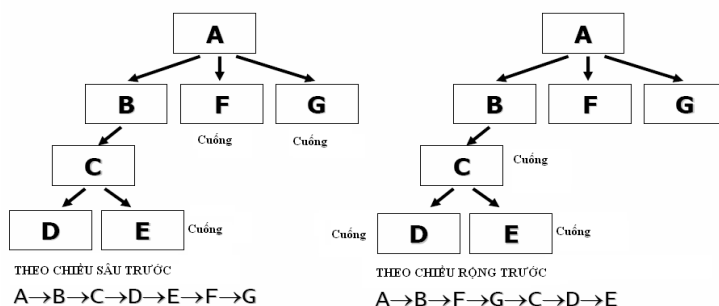
- Kiểm trong tích hợp :
 - So sánh ngữ nghĩa của các giao diện giữa các đơn vị có trao đổi với nhau, tránh sự vênh giao diện (interface mismatch) về ngữ nghĩa ; hoặc giữa hệ thống và bên ngoài (input/output mismatch) ;
 - Vênh giao diện thường đưa đến những lỗi rất khó hiểu. Các chương trình biên dịch chỉ tìm ra sự vênh về cú pháp
 - Phân tích luồng thông tin (information flow analysis): kiểm soát những ràng buộc trong thứ tự gửi nhận thông điệp giữa các bộ phận hay với bên ngoài
- Thử tích hợp là công việc tương đối rất khó
 - Cần các chuyên viên nhiều kinh nghiệm : thường là một nhóm độc lập với nhóm lập trình
 - Cần càng nhiều công cụ hỗ trợ tự động càng tốt
 - Nên bỏ thì giờ viết lấy một số công cụ đặc biệt cho dự án nếu công cụ CASE không đủ.

KIỂM THỬ TÍCH HỢP

- Kiểm thử bigbang
- Tích hợp từ trên xuống
- Tích hợp từ dưới lên
- Bình luận về việc kiểm thử tích hợp
- Tài liệu kiểm thử tích hợp

TÍCH HỢP TỪ TRÊN XUỐNG

- Thử các giao diện trao đổi trước
- Cần viết thêm một chương trình giả (stubs) để thay thế một cách đơn giản và tạm thời mỗi mô đun chưa thực sự được tích hợp ; trong đó chỉ có các giao diện và các bảng biến đổi vào ra đã tính sẵn
- Cần phác thảo thứ tự tích hợp thực sự : Các mô đun được tích hợp bằng cách đi dần xuống qua cấp bậc điều khiển, bắt đầu với mô đun điều khiển chính (ctrình chính). Các mô đun phụ thuộc vào mô đun điều khiển chính sẽ được tổ hợp dần vào trong cấu trúc theo chiều sâu trước hoặc chiều rộng trước.



TIẾN TRÌNH TÍCH HỢP TỪ TRÊN XUỐNG

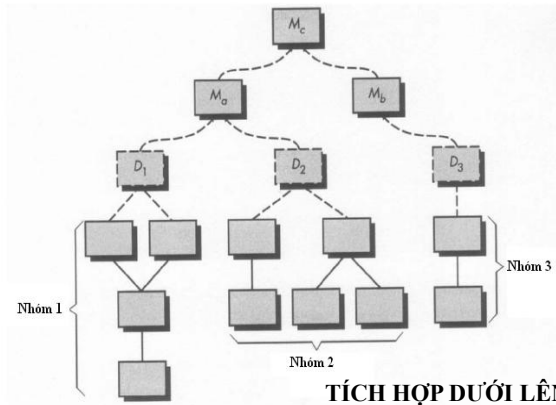
- Mô đun điều khiển chính được dùng như 1 khiển trình kiểm thử và các mô đun phụ thuộc được thể bởi các cuống
- Tùy theo cách tiếp cận tích hợp được chọn lựa, các cuống phụ thuộc được thay thế bằng các mô đun thực tại
- Việc kiểm thử được tiến hành khi từng mô đun được tích hợp vào
- Khi hoàn thành tập các phép kiểm thử, cuống sẽ được thay thế bằng các mô đun thực
- Kiểm thử hồi quy có thể được tiến hành để đảm bảo rằng những lỗi mới không bị đưa thêm vào

TÍCH HỢP TỪ DƯỚI LÊN

- Chiến lược tích hợp dưới lên:
 - Cần phác thảo cây tích hợp
 - Các mô đun mức thấp được tổ hợp vào các chùm (kiểu kiến trúc) thực hiện cho một chức năng con phần mềm riêng biệt
 - Khiến trình được viết ra để phối hợp việc vào và ra trường hợp kiểm thử
 - Kiểm thử chùm
 - Loại bỏ khiến trình và chùm được tổ hợp, chuyển lên trong cấu trúc chương trình.

TÍCH HỢP TỪ DƯỚI LÊN (tt)

- ❑ Mỗi lần thử cần viết thêm một chương trình khung tích hợp (test&integration driver) để thay thế một cách đơn giản và tạm thời những môđun vắng mặt



Test-workbench (cơ sở thử) là một dụng cụ CASE cho phép viết nhanh những test&integration driver, và phát sinh nhanh dữ liệu thử

BÌNH LUẬN VỀ VIỆC TÍCH HỢP KIỂM THỬ

- Kiểm thử từ trên xuống
 - ❑ Cho phép sớm phát hiện các lỗi về kiến trúc
 - ❑ Sớm có các phiên bản hoạt động
 - ❑ Các bộ dữ liệu thử có thể được tái sử dụng cho các bước tiếp theo
 - ❑ Tuy nhiên chiến lược này đòi hỏi phải xây dựng nhiều stubs, khó có thể mô phỏng được các chức năng của các modun cấp thấp phức tạp.
- Kiểm thử từ dưới lên
 - ❑ Ít xây dựng các stub phức tạp,
 - ❑ Thuận tiện cho việc phát triển các modun thứ cấp dùng lại được
 - ❑ Nhưng phát hiện lỗi thiết kế trễ hơn
- Trên thực tế người ta tìm cách phối hợp hai chiến lược kiểm thử này, gọi là **sandwich testing**.

KIỂM THỬ HỒI QUI

- Là tiến hành lại các phép thử đã thành công mỗi khi tích hợp thêm môdun hoặc khi cập nhật mã nguồn chương trình.
 - Khi tích hợp thêm môdun mới hoặc khi tiến hành nâng cấp chương trình thì sẽ tạo ra một số tổ hợp trạng thái mới dẫn đến:
 - ❑ Xuất hiện lỗi ở môdun trước đây chưa gây lỗi.
 - ❑ Khắc phục một lỗi mới có thể sẽ làm ảnh hưởng tới một lỗi đã sửa.
 - ❑ Sinh ra lỗi mới mà trước đây chưa từng có.
-

TÀI LIỆU KIỂM THỬ TÍCH HỢP

DÀN BÀI ĐẶC TẢ KIỂM THỬ

- I. Phạm vi kiểm thử
 - II. Kế hoạch kiểm thử
 - A. Các giai đoạn và khối kiểm thử
 - B. Lịch biểu
 - C. Tổng phí phần mềm
 - D. Môi trường và tài nguyên
 - III. Thủ tục kiểm thử n (mô tả việc kiểm thử cho khối n)
 - A. Thứ tự tích hợp
 - 1. Mục đích
 - 2. Mô đun cần kiểm thử
 - B. Kiểm thử đơn vị cho các mô đun trong khối
 - 1. Mô tả kiểm thử cho mô đun m
 - 2. Mô tả tổng phí phần mềm
 - 3. Kết quả dự kiến
 - C. Môi trường kiểm thử
 - 1. Công cụ hay kỹ thuật đặc biệt
 - 2. Mô tả tổng phí phần mềm
 - D. Dữ liệu trường hợp kiểm thử
 - E. Kết quả dự kiến cho khối n
 - IV. Kết quả kiểm thử thực tế
 - V. Tham khảo
-

KIỂM THỬ HỢP LỆ

- ❑ Kiểm thử hợp lệ là để đảm bảo rằng chức năng phần mềm theo đúng đặc tả yêu cầu
 - ❑ Sự đặc tả yêu cầu phần mềm tốt chứa đựng trong phần gọi là “**Tiêu chuẩn kiểm tra hợp lệ**” và dựa vào đó để tiến hành các kiểm thử hợp lệ
 - ➔ Tốt nhất là được thực hiện bởi nhóm kiểm thử độc lập nhóm phát triển và có thể bao gồm cả khách hàng.
-

KIỂM THỬ HỢP LỆ

- Lập kế hoạch và trường hợp kiểm thử thiết kế để đảm bảo rằng:
 - ❑ Tất cả yêu cầu chức năng được thoả mãn
 - ❑ Tất cả đặc trưng hiệu năng được thoả mãn
 - ❑ Những yêu cầu khác (như khả năng tương thích, khắc phục lỗi, bảo hành bảo trì) được đáp ứng.
 - Sau khi tiến hành mỗi trường hợp kiểm thử tồn tại:
 - ❑ Yêu cầu chức năng và hiệu năng phù hợp với đặc tả và được chấp nhận
 - ❑ Độ lệch so với đặc tả được phát hiện và danh sách các khiếm khuyết tạo ra.
-

KIỂM THỬ HỢP LỆ

- Nếu phần mềm được phát triển như sản phẩm được sử dụng bởi nhiều khách hàng thì không thực tế cho việc thực hiện các kiểm thử chấp nhận chính thức cho từng khách hàng.
 - Phần lớn người xây dựng phần mềm sử dụng 1 trong 2 kiểu:
 - Kiểm thử hợp lệ Alpha
 - Kiểm thử hợp lệ Beta
-

CÁC KIỂU KIỂM THỬ HỢP LỆ

- Kiểm thử hợp lệ Alpha
 - Được khách hàng kiểm soát tại nơi người phát triển phần mềm tạo ra nó, làm việc trong môi trường có kiểm soát. Phần mềm được khách hàng sử dụng và người phát triển ghi lại các lỗi và những vấn đề về cách sử dụng.
 - Kiểm thử hợp lệ Beta
 - Được tiến hành tại nơi khách hàng. Khác với kiểm thử Alpha là người phát triển không hiện diện
 - KH ghi tất cả những vấn đề mà gặp phải (Thực tế hoặc tương tượng) và báo cáo vấn đề đó cho người phát triển trong những khoảng thời gian sớm nhất
 - Người phát triển sửa đổi chuẩn bị phát hành sản phẩm phần mềm cho toàn bộ khách hàng
-

KIỂM THỬ HỆ THỐNG

- Phần mềm là 1 phần của hệ thống máy tính lớn hơn.
- Phần mềm được kết hợp với thành phần hệ thống khác (như phần cứng, thông tin). Một loạt kiểm thử hệ thống sẽ được tiến hành
- Kiểm thử khả năng hoạt động của hệ thống, kiểm tra các vấn đề về hiệu năng của sản phẩm, khả năng phục hồi khi gặp sự cố,...
- Các kiểu kiểm thử hệ thống:

□ Kiểm thử phục hồi	Recovery
□ Kiểm thử bảo mật	Security
□ Kiểm thử ứng suất	Stress
□ Kiểm thử hiệu suất	Performance

CÁC KIỂU KIỂM THỬ HỆ THỐNG

- Kiểm thử phục hồi : là kiểm thử hệ thống bắt buộc phần mềm phải hỏng theo nhiều cách và kiểm chứng rằng việc phục hồi được thực hiện đúng.
- Kiểm thử bảo mật:
 - Các hệ thống có các thông tin “nhạy” sẽ có tiềm năng bị tấn công và đánh cắp
 - Trong suốt giai đoạn kiểm thử bảo mật, người kiểm thử đóng vai trò là cá nhân cố gắng xâm nhập vào hệ thống: ăn cắp mật khẩu, làm phát sinh các lỗi hệ thống.

CÁC KIỂU KIỂM THỬ HỆ THỐNG

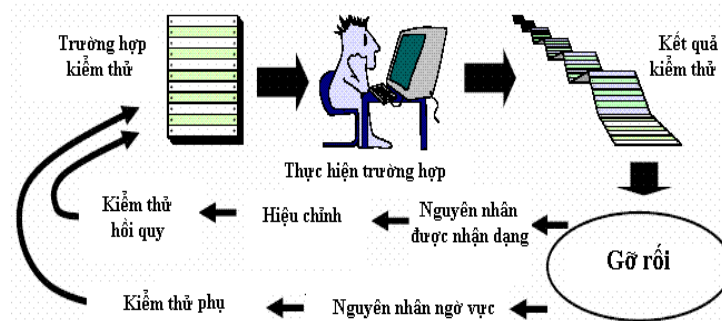
- Kiểm thử ứng suất:
 - Được thiết kế để làm cho phần mềm đương đầu với các tình huống bất thường như ngắt, tràn bộ nhớ, tài nguyên,...
 - Mục đích để tìm hiểu giới hạn chịu tải của hệ thống, đặc trưng của hệ thống khi đạt và vượt giới hạn chịu tải (khi bị sụp đổ)
 - Kiểm thử thực thi:
 - Được thiết kế để quá trình thực thi của phần mềm trong suốt quá trình chạy trong hệ thống tích hợp (trong môi trường hoàn chỉnh đủ các điều kiện để thực thi sản phẩm).
 - Kiểm thử thực thi được xuất hiện trong tất cả các bước của tiến trình kiểm thử
-

GỠ LỖI

- Tiến trình gỡ lỗi
 - “Nghệ thuật” gỡ lỗi
 - Cách tiếp cận gỡ lỗi
-

TIẾN TRÌNH GỠ LỖI

- Gỡ lỗi là tiến trình độc lập với kiểm thử
- Gỡ lỗi là tiến trình xuất hiện khi kiểm thử thành công tức là lỗi được phát hiện thì gỡ lỗi sẽ tiến hành loại bỏ lỗi



“NGHỆ THUẬT” GỠ LỖI

- Không may, một số bằng chứng là sự tinh thông gỡ lỗi thuộc về bẩm sinh của con người. Một số người làm việc này rất giỏi trong khi một số khác thì lại không.
- Mặc dù rất khó để “học” gỡ lỗi, đề nghị ra một số cách tiếp cận gỡ lỗi :
 - ❑ Bắt ép mạnh bạo (Brute Force)
 - ❑ Lăn ngược (Backtracking)
 - ❑ Loại bỏ nguyên nhân (Cause Elimination)

CÁCH TIẾP CẬN GỠ LỖI

■ **Bắt ép mạnh bạo:**

- ❑ Là phương pháp giải quyết vấn đề bằng cách lặp đi lặp lại một thủ tục đơn giản nhiều lần
 - ❑ Là phương pháp thông dụng nhất và kém hiệu quả nhất để cô lập nguyên nhân của lỗi phần mềm. Áp dụng khi các phương pháp khác thất bại
 - ❑ Dùng triết lí “cứ để máy tính tìm ra lỗi”
-

CÁCH TIẾP CẬN GỠ LỖI

■ **Lần ngược:**

- ❑ Là phương pháp khá thông dụng được dùng thành công trong những chương trình nhỏ.
 - ❑ Bắt đầu từ dấu hiệu, lần ngược theo chương trình gốc cho đến khi tìm ra nguyên nhân.
 - ❑ Trong chương trình lớn, số con đường lần ngược tiềm năng có thể trở nên không quản lý nổi
-

CÁCH TIẾP CẬN GỠ LỖI

■ Loại bỏ nguyên nhân:

- ❑ Quy nạp: Thu thập dữ liệu liên quan tới việc xuất hiện lỗi, và đặt trong “giả thiết nguyên nhân”, dữ liệu này dùng để chứng minh hay bác bỏ giả thiết đó
 - ❑ Diễn dịch: Đưa ra danh sách các nguyên nhân. Tiến hành phép kiểm thử để loại trừ dần nguyên nhân, cho đến khi nguyên nhân thực sự được cô lập.
-

GIỚI THIỆU CÔNG CỤ Track+3.0

- Việc ngừng dự án bởi lỗi có thể gây ra những thiệt hại khôn lường cho doanh nghiệp.
 - Track+3.0 ra đời nhằm giúp các doanh nghiệp quản lý dự án tốt hơn.
 - Mã nguồn mở, free hoàn toàn.
 - Hỗ trợ các database thông dụng như Oracle, MySQL, SQLServer.
 - Có khả năng tích hợp rất cao đối với môi trường có sẵn của người dùng.
-

GIỚI THIỆU CÔNG CỤ Track+3(tt)

- Với Track+ bạn có các công cụ sau:
 - ❑ Theo dõi lỗi (Bug Tracking)
 - ❑ Quản lý chất lượng
 - ❑ Quản lý rủi ro (Risk lists)
 - ❑ Quản lý nhân viên thực hiện dự án (Contractor managerment)
-

GIỚI THIỆU CÔNG CỤ Track+3.0 (tt)

- Cấu hình yêu cầu cần có của Track+3.0:
 - ❑ Windows XP
 - ❑ My SQL
 - ❑ Resin (hoặc Tomcat)
 - ❑ JDK 1.4.2



CÁC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Giải thích
Software Testing	Kiểm thử phần mềm
Development Test	Kiểm tra phát triển
Unit Test	Kiểm tra đơn vị
Subsystem integration test	Kiểm tra tích hợp
System test	Kiểm tra hệ thống
Quality assurance (QA)	Đảm bảo chất lượng
Acceptance test	Kiểm tra chất lượng
Information System (IS)	Quản lý hệ thống thông tin

TÀI LIỆU THAM KHẢO

- Công nghệ phần mềm Lê Đức Trung
 - Kỹ nghệ phần mềm Tập 3 Ngô Trung Việt
 - Software Engineering Roger S. Pressman
 - Forum VietRose
-