Kiểm thử và đảm bảo chất lượng phần mềm

Giới thiệu

Nội dung

- Một số khái niệm cơ bản
- Tại sao cần kiểm thử phần mềm?
- Mục tiêu của kiểm thử
- Kiểm chứng và kiểm định phần mềm
- Kiểm thử hộp trắng, kiểm thử hộp đen

Phần mềm

- Ngành công nghiệp phần mềm đã phát triển mạnh mẽ trong vài thập kỷ qua.
- Phần mềm ngày càng:
  - Lớn hơn
  - Cạnh tranh hơn
  - Có nhiều người dùng hơn
- Phần mềm là một phần quan trọng của:
  - airplanes, spaceships, air traffic control systems…
  - watches, ovens, cars, DVD players, mobile phones, remote controls…

Tại sao cần kiểm thử phần mềm?

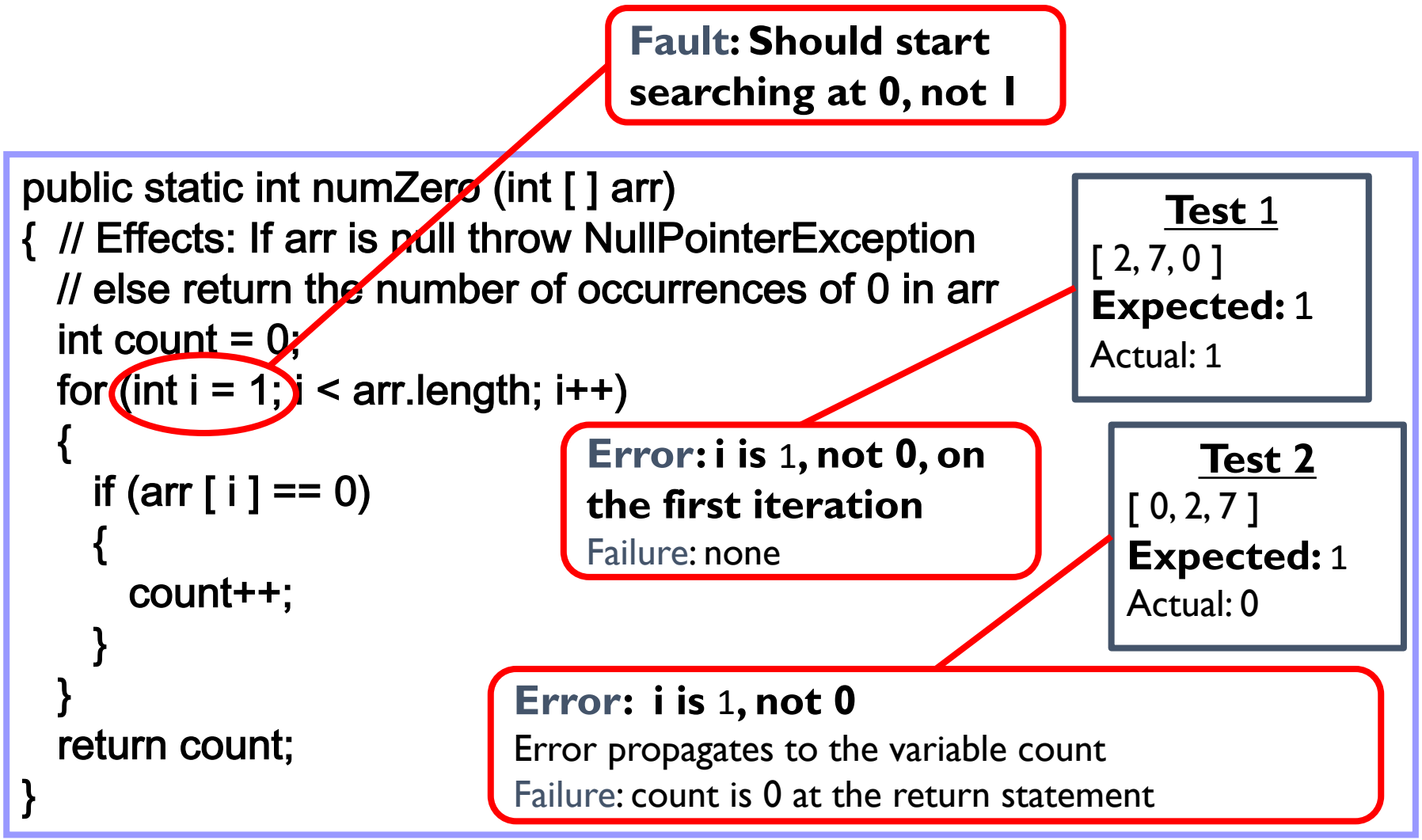
- Phần mềm ngày càng được ứng dụng vào nhiều lĩnh vực.
- Con người ngày càng phụ thuộc chặt chẽ vào các sản phẩm phần mềm.
- Ngành công nghiệp phần mềm đang đối mặt với hai thách thức:
  - Sản xuất phần mềm với giá thành thấp
  - Phần mềm cần dễ dùng, an toàn và tin cậy

→ **Kiểm thử có phương pháp là một hoạt động không thể thiếu**

Software faults, errors & failures

- **Software fault:** A static defect in software
- **Software error:** An incorrect interal state that is the manifestation of some fault
- **Software failure:** External, incorrect behavior with respect to the requirements or other description of the expected behavior

Ví dụ


Fault: Should start searching at 0, not 1
public static int numZero (int [ ] arr)
{ // Effects: If arr is null throw NullPointerException
// else return the number of occurrences of 0 in arr
int count = 0;
for (int i = 1; i < arr.length; i++)
{
if (arr [i] == 0)
{
count++;
}
}
return count;
}
Test 1
[2, 7, 0]
Expected: 1
Actual: 1
Error: i is 1, not 0, on the first iteration
Failure: none
Test 2
[0, 2, 7]
Expected: 1
Actual: 0
Error: i is 1, not 0
Error propagates to the variable count
Failure: count is 0 at the return statement

Câu chuyện về phần mềm thất bại

- **NASA's Mars lander:** 9/1999, tàu không gian đã bị đâm vào sao hoả bởi vì lỗi đơn vị tính của 2 mô-dun được làm bởi 2 nhóm phát triển khác nhau.

- **Therac-25 radiation machine**: Không kiểm tra kỹ tính an toàn của phần phần mềm đã làm cho 3 người thiệt mạng.

- **Chip Pentium of Intel**: đưa ra kết quả sai cho một số phép chia

Câu chuyện về phần mềm thất bại

- **Airbus A220:** Sau khi cập nhật phần mềm → động cơ rung quá mức cho phép → ít nhất có 3 chuyến bay phải hạ cánh khẩn cấp

- **Lỗi phanh của Toyota**: rất nhiều vụ tai nạn đã xảy ra

→Lỗi nên được tìm ra trước khi phát hành sản phẩm/ bàn giao sản phẩm cho khách hàng

→Kiểm thử là một cách để đo độ tin cậy của phần mềm

Verification and validation

- **Verification:** Quá trình kiểm tra xem các sản phẩm của một giai đoạn (phases) trong quy trình phát triển phần mềm có đáp ứng các yêu cầu đã được thiết lập trong giai đoạn trước đó hay không.

- **Validation:** Quá trình đánh giá phần mềm khi kết thúc quy trình phát triển phần mềm, để đảm bảo phần mềm đáp ứng đúng mong muốn của khách hàng.

Chất lượng của phần mềm

- **Góc độ người dùng:** Chất lượng là đáp ứng được nhu cầu, mục đích của người dùng
- **Góc độ của nhà sản xuất:** Chất lượng được hiểu là phần mềm tuân theo đặc tả. Mức độ của chất lượng được quyết định bởi mức độ mà nó tuân theo đặc tả.
- **Góc độ sản phẩm:** Chất lượng được xem xét bởi các tính chất, đặc tính của sản phẩm (internal qualities, external qualities)
- **Góc độ giá trị:** Phụ thuộc vào số lượng khách hàng sẵn sàng chi trả cho việc dùng sản phẩm

Kiểm thử chương trình

- Là hoạt động chủ chốt nhằm đánh giá chất lượng
- Có thể chỉ ra lỗi, không thể khẳng định không còn lỗi
- Một ca kiểm thử thành công là ca kiểm thử phát hiện ra lỗi

Mục tiêu của kiểm thử

- Chỉ ra rằng phần mềm hoạt động (it does work)
- Chỉ ra rằng phần mềm không hoạt động (it does not work)
- Giảm rủi ro của phần mềm thất bại (reduce the risk of failure)
- Giảm chi phí kiểm thử (reduce the cost of testing)

- Beizer chia mục tiêu kiểm thử thành 5 mức (Introduction to Software Testing, p34)

Phân tích tĩnh và phân tích động

- Chất lượng phần mềm được cải tiến thông qua chu trình: Kiểm thử – tìm lỗi – sửa lỗi

- Quy trình này gồm 2 công việc chính:

+ Phân tích tĩnh

- Khảo sát đặc tả, các tài liệu thiết kế, mã nguồn
- Cũng có thể dùng các phương pháp hình thức

+ Phân tích động

- Thực thi chương trình để phát hiện thất bại

Ca kiểm thử

- Một cặp **<test input, expected output>**
- Ví dụ: hàm add(int a, int b), thực hiện cộng hai tham số đầu vào và trả về tổng của chúng

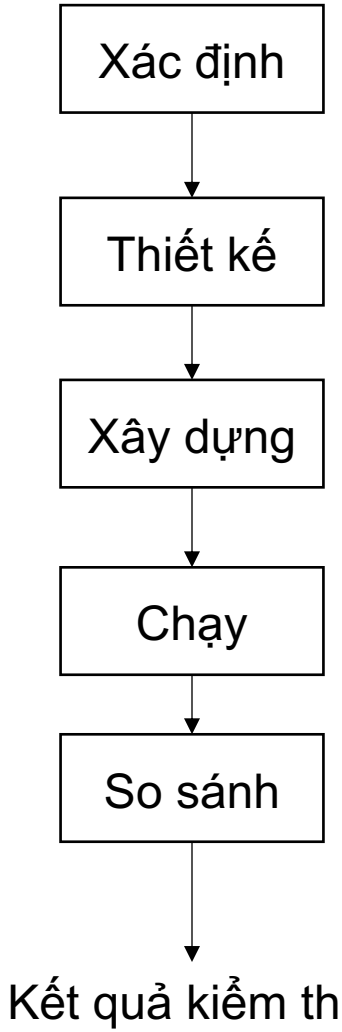
T1 = <(0, 1), 1>

T2 = <(1, 2), 3>

- **Test report**

ID	Test input	Expected output	System output	Test result
T1				
T2				
T3				

Các hoạt động kiểm thử



Điều kiện kiểm thử ("Cái gì"): Một phần tử hoặc sự kiện cần kiểm tra

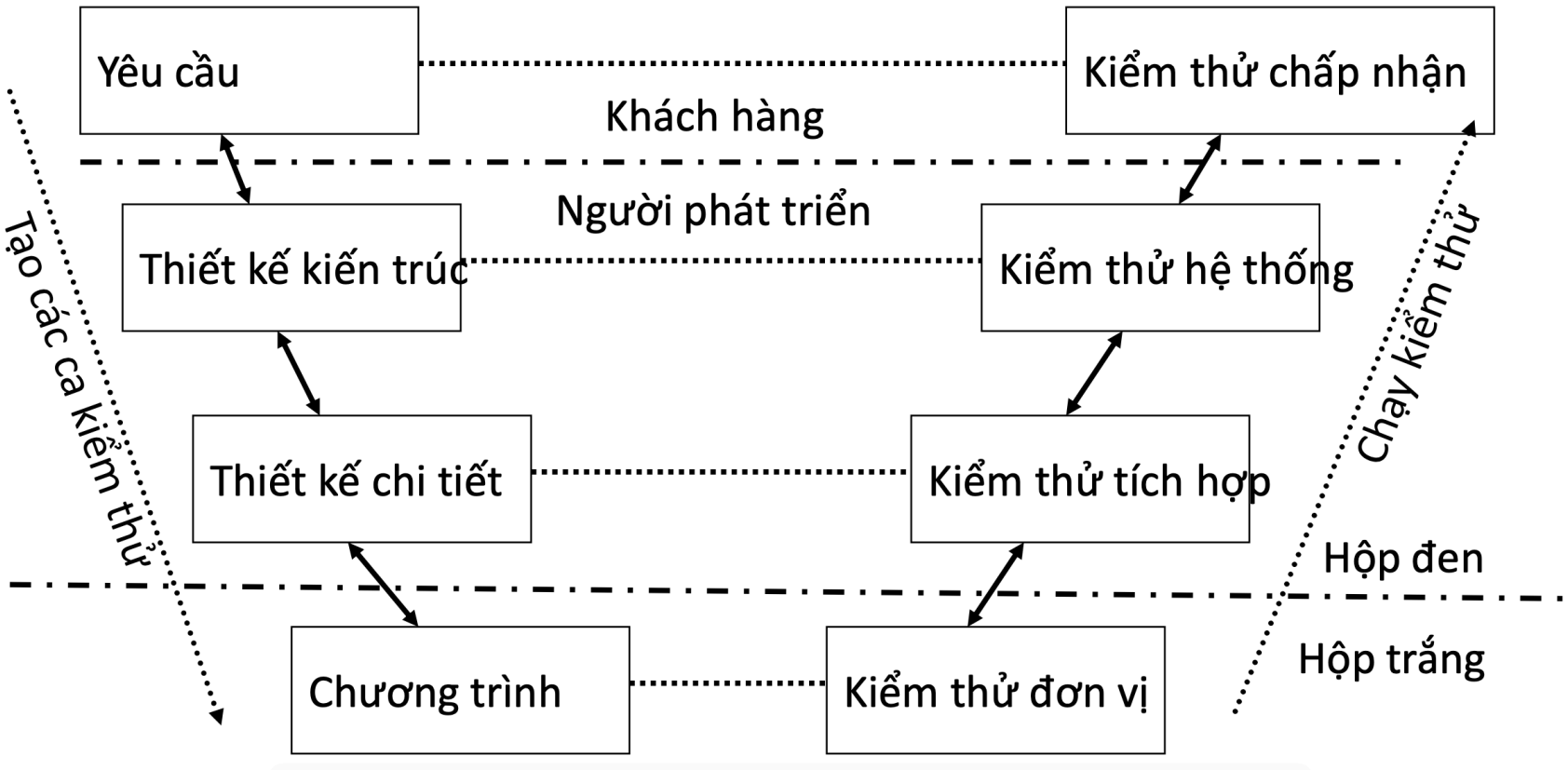
Làm thế nào để có thể kiểm tra được: thực hiện

Xây dựng ca kiểm thử

Chạy hệ thống

So sánh kết quả của hệ thống và kết quả mong đợi

Kiểm thử trong mô hình chữ V

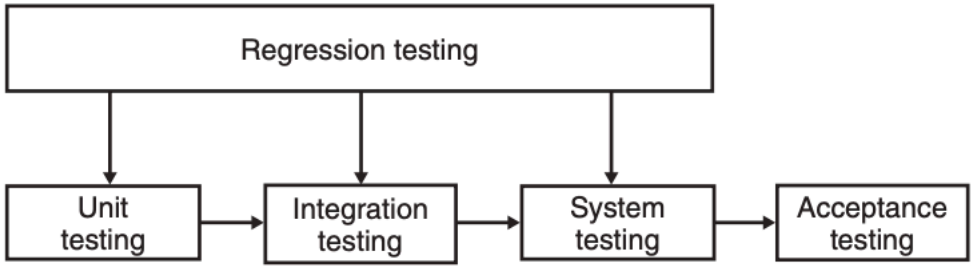


Kiểm thử hồi quy - regression testing

- Khi hệ thống có một chỉnh sửa (sửa lỗi, thêm/bớt chức năng,…) toàn bộ bộ kiểm thử cần phải chạy lại

  - Đảm bảo các tính năng đang hoạt động tốt không bị ảnh hưởng bởi chỉnh sửa

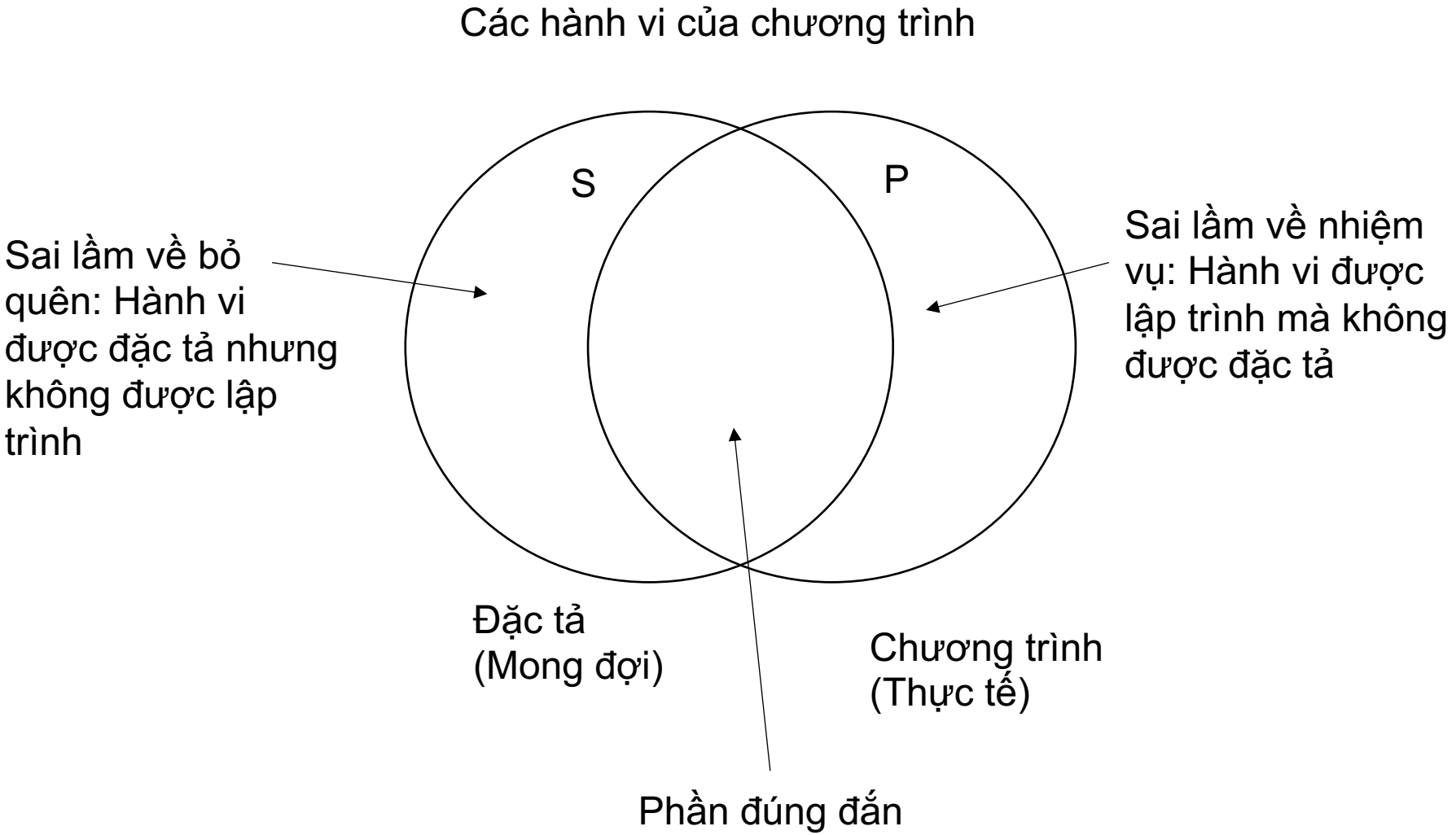
- Không phải là một mức kiểm thử riêng biệt, mà là một phần của kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống



Khi nào nên dừng kiểm thử

- Hết thời gian, hết ngân sách
- Đạt mức độ bao phủ mong muốn
- Đạt tần suất hỏng hóc mong muốn

Bài toán kiểm thử qua biểu đồ ven



Kiểm thử hộp đen

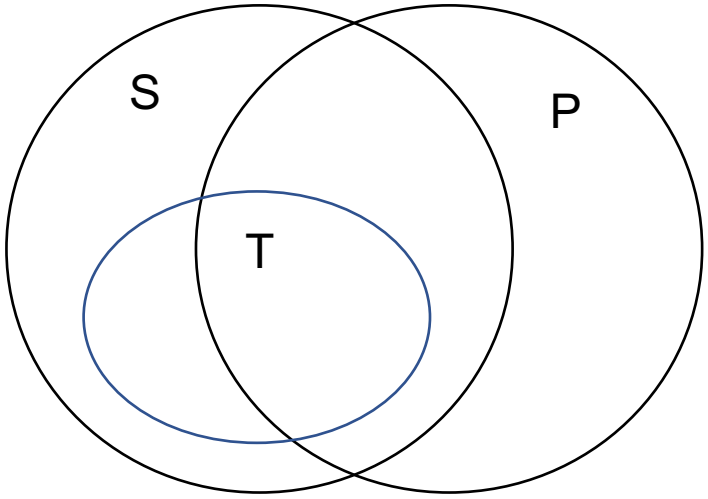
- Kiểm thử hàm, kiểm thử chức năng
- Chương trình được coi như một hộp đen, chỉ quan tâm đến đầu vào và đầu ra của chương trình
- Test input được lựa chọn từ đặc tả của chương trình và các tính chất của miền đầu vào và đầu ra
- Testers chỉ quan tâm đến các chắc năng và tính năng được mô tả trong đặc tả

Kiểm thử hộp trắng

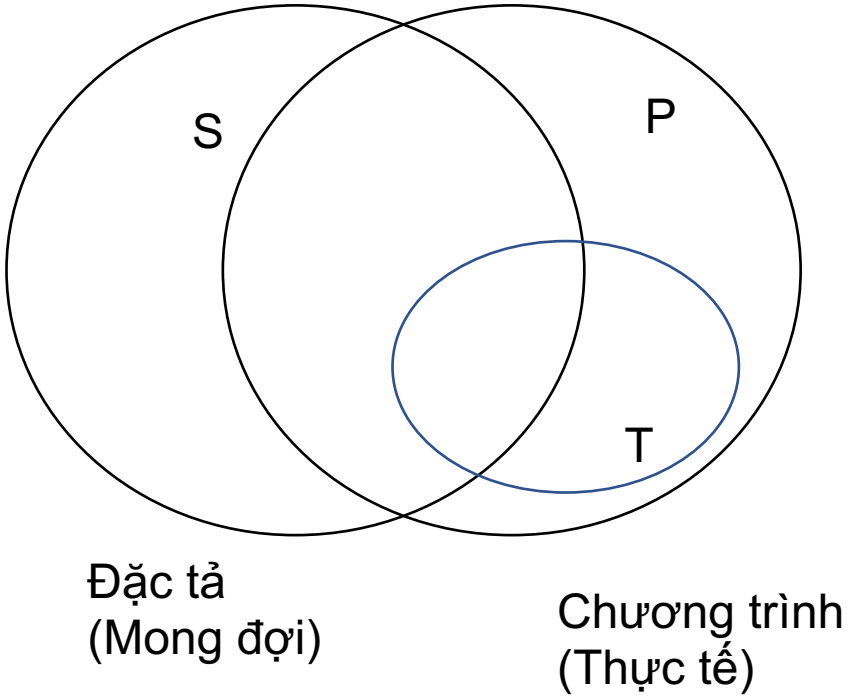
- Kiểm thử cấu trúc, kiểm thử logic
- Kiểm tra mã nguồn dựa trên luồng dữ liệu và luồng điều khiển

Kiểm thử hộp đen và hộp trắng

Kiểm thử hộp đen



Kiểm thử hộp trắng



Công cụ kiểm thử

- Kiểm thử là một công việc tốn kém (cần nhiều lao động) nhất của quá trình phát triển phần mềm
  - Test cases thường được tạo ra một cách thủ công
  - Kết quả kiểm thử được phân tích thủ công

- Có nhiều công cụ hỗ trợ: static code analyzer, test data generator, network analyzer…

Note: **Không kiểm thử thì chi phí phát triển phần mềm còn đắt đỏ, tốn kém hơn nữa**

Nhiều công cụ hỗ trợ các loại kiểm thử

- Kiểm thử đơn vị: Achoo, Junit, Pex/Moles, PyUnit
- Tự động kiểm thử: TestComplete
- Kiểm thử hiệu năng và tải: Jmeter
- Kiểm thử giao diện đồ hoạ: Abbot, Guitar
- Kiểm thử tổ hợp: AETG, FireEye
- Kiểm thử dựa trên mô hình: Spec Explorer
- Phân tích bao phủ: Corbertura
- Quản lý lỗi (defects): Bugzilla

Bài tập

1. Sự khác nhau giữa fault và failure là gì?
2. Bài 5 trang 40, sách Introduction to Software Testing
3. Mô tả một bài toán với đầu vào, đầu ra và yêu cầu chức rõ ràng. Bài toán này sẽ được sử dụng để làm bài tập trong toàn khoá học. Tham khảo các ví dụ trong chương 2 sách giáo trình.