Bài 3: Các loại kiểm thử

Nội dung bài học

- Static Testing - Kiểm thử tĩnh
- BlackBox Testing - Kiểm thử hộp đen
- WhiteBox Testing - Kiểm thử hộp trắng
- Non-Functional Testing - Kiểm thử phi chức năng

STATIC TESTING – KIỂM THỬ TĨNH

Static Testing - Kiểm thử tĩnh

- Kiểm thử tĩnh là hoạt động kiểm tra bằng cách Review và sẽ không chạy chương trình (hoặc phần mềm) để kiểm thử.
- Kiểm thử tĩnh được thực hiện ở giai đoạn đầu của chu kỳ phát triển phần mềm.
- Kiểm thử tĩnh sẽ kiểm tra tính đúng đắn của code (mã lệnh), thuật toán hay tài liệu.

- Những hoạt động Review trong kiểm thử tĩnh
  - Walkthrough - Hướng dẫn: Một buổi walkthrough là một buổi họp mà ở đó tác giả của các tài liệu đặc tả phần mềm giới thiệu sơ lược về dự án phần mềm sẽ làm để cho những người tham dự có một sự hiểu biết chung về phần mềm đó, đồng thời thu thập những ý kiến phản hồi từ họ.

- Technical Review - Đánh giá kỹ thuật: Một cuộc họp ngang hàng mà ở đó những người tham dự là những người cùng nhóm kỹ thuật với nhau.

Ví dụ: Developer Team, Tester Team. Cuộc họp này được các nhóm tổ chức để quyết định những công nghệ sẽ sử dụng để lập trình hoặc kiểm thử (ví dụ: dùng Java, C#, Ruby,… dùng kỹ thuật kiểm thử hộp đen, hộp trắng, kiểm thử theo Test Cases, kiểm thử tự do,…) và đề xuất những thuật toán sẽ sử dụng, những giải pháp thay thế khi tiến hành phát triển phần mềm.

- Inspection - Kiểm tra: Là cuộc họp mà ở đó các thành viên trong dự án sẽ tham dự cuộc họp, đưa ra những câu hỏi để làm rõ vấn đề, trình bày những lỗi sai hoặc những vấn đề chưa có hướng giải quyết trong tài liệu và đề xuất những phương án hợp lý để cải thiện. Những thành viên tham gia sẽ trình bày, phản biện, tranh luận, kiến nghị,… nhằm mục đích đưa dự án hoạt động đúng hướng, theo đúng mong đợi của khách hàng.

- Những công việc được kiểm tra bởi kiểm thử tĩnh
  - Xem xét đặc tả yêu cầu: yêu cầu nghiệp vụ, yêu cầu chức năng, yêu cầu bảo mật.
  - Xem xét user stories (yêu cầu người dùng), Acceptance criteria (tiêu chí chấp nhận).
  - Xem xét code
  - Xem xét tài liệu thiết kế và kiến trúc...

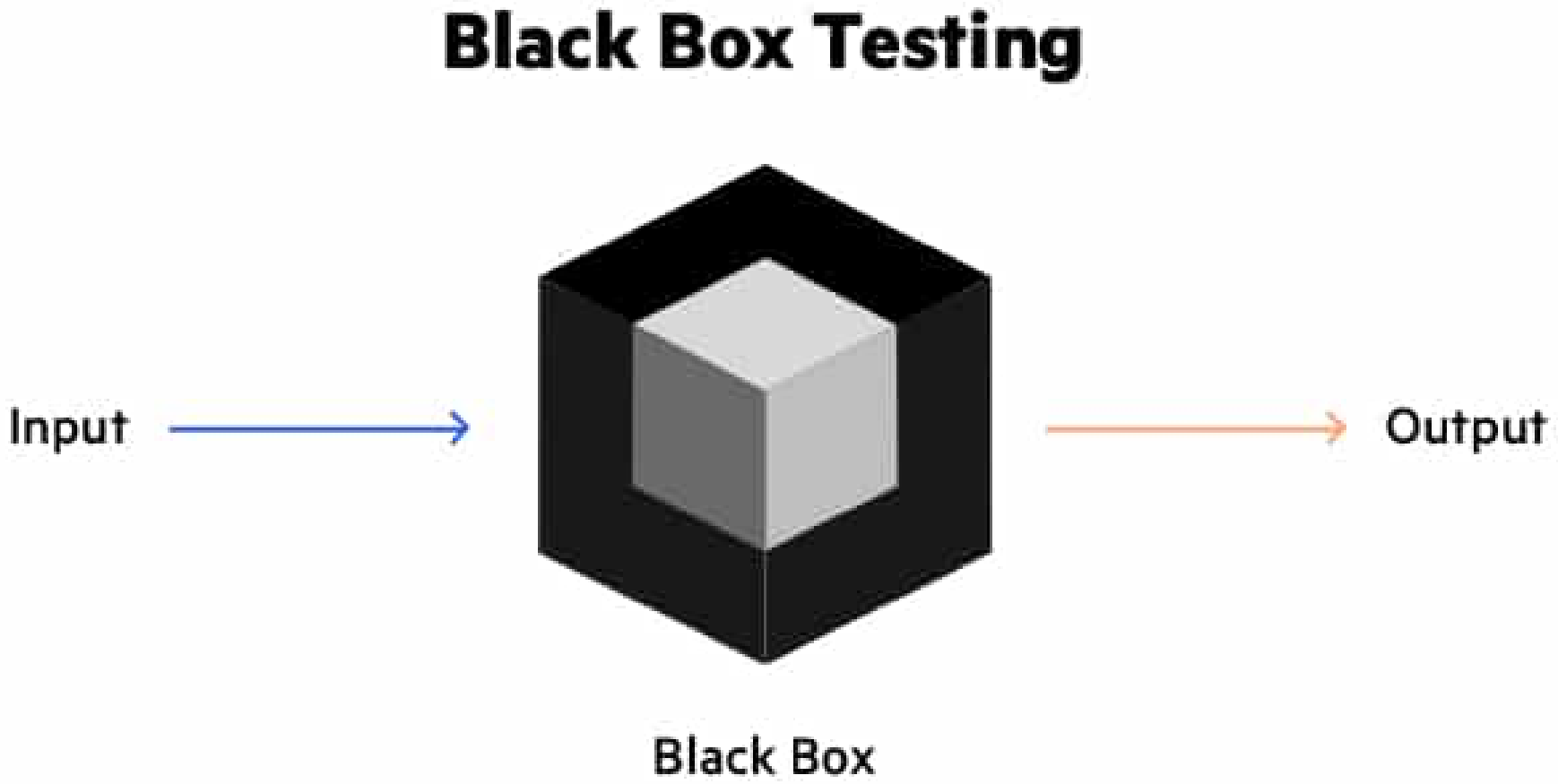
Ưu điểm của Kiểm thử tĩnh

- Bởi vì kiểm thử tĩnh có thể bắt đầu sớm trong quy trình phát triển phần mềm, do đó sẽ có được những phản hồi sớm về vấn đề chất lượng của phần mềm cũng như dự án.
- Phát hiện các lỗi ở giai đoạn đầu, chi phí sửa chữa sẽ thấp.
- Kiểm thử tĩnh góp phần gia tăng nhận thức về các vấn đề chất lượng.

Các lỗi điển hình được tìm thấy bởi Kiểm thử tĩnh

- Lỗi về tài liệu yêu cầu, như sự không nhất quán, mơ hồ, mâu thuẫn, thiếu sót, không chính xác và dư thừa trong tài liệu.
- Lỗi thiết kế, như thuật toán không hiệu quả, cấu trúc cơ sở dữ liệu.
- Lỗi code, như các biến có giá trị không xác định, các biến được khai báo nhưng không bao giờ sử dụng, code không thể truy cập, code bị trùng lặp.
- Độ lệch so với tiêu chuẩn như thiếu tuân thủ theo các tiêu chuẩn code.
- Giao diện không chính xác.

KIỂM THỬ HỘP ĐEN


Black Box Testing
Input
Output
Black Box

BlackBox Testing - Kiểm thử hộp đen

- Kiểm thử hộp đen là một phương pháp kiểm thử phần mềm được thực hiện mà không biết được cấu tạo bên trong của phần mềm, là cách mà các tester kiểm tra xem hệ thống như một chiếc hộp đen, không có cách nào nhìn thấy bên trong của cái hộp.
- Kiểm thử hộp đen thực hiện theo hướng dữ liệu input và output.
- Kiểm thử viên đối với hệ thống là không dùng bất kỳ một kiến thức về cấu trúc lập trình bên trong hệ thống, xem hệ thống là một cấu trúc hoàn chỉnh, không thể can thiệp vào bên trong.

Đối tượng Kiểm thử hộp đen

- Đối tượng được kiểm thử là 1 thành phần phần mềm (TPPM). TPPM có thể là 1 chức năng, 1 module chức năng, 1 phân hệ chức năng… Nói chung, chiến lược kiểm thử hộp đen thích hợp cho mọi cấp độ kiểm thử từ kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử độ chấp nhận của người dùng.

Các bước Kiểm thử hộp đen

- Bước 1: Phân tích đặc tả về các yêu cầu chức năng mà TPPM cần thực hiện.
- Bước 2: Dùng 1 kỹ thuật định nghĩa các testcase xác định(sẽ giới thiệu sau) để định nghĩa các testcase. Định nghĩa mỗi testcase là xác định 3 thông tin sau :
  - ✔ Giá trị dữ liệu nhập để TPPM xử lý (hoặc hợp lệ hoặc không hợp lệ).
  - ✔ Trạng thái của thành phần phần mềm(TPPM) cần có để thực hiện testcase.
  - ✔ Giá trị dữ liệu xuất mà TPPM phải tạo được.

- Bước 3: Kiểm thử các testcase đã định nghĩa.
- Bước 4: So sánh kết quả thu được với kết quả kỳ vọng trong từng testcase, từ đó lập báo cáo về kết quả kiểm thử.

Biểu mẫu Test case:

ID	Test Case Description	Test Case Procedure	Expected Result	Test Result
1	*<Brief description of this case: what is tested?>* Test username and password = empty	*<Describe steps to perform this case>* 1. Username and Pasword textbox = empty 2. Click "Log in" button	*<Describe results which meet customer's requirement>* Message displays 'Username and Password are required'	Pass
2	<Test case 2>			Fail

❑ Các phương pháp Kiểm thử hộp đen

Vì chiến lược kiểm thử hộp đen thích hợp cho mọi mức độ kiểm thử nên có nhiều kỹ thuật kiểm thử. Sau đây 3 loại phổ biến:

- ➢ Phương pháp phân vùng tương đương - Equivalence Class Partitioning
- ➢ Phân tích các giá trị biên - Boundary value analysis
- ➢ Kỹ thuật dùng các bảng quyết định - Decision Tables

Ưu điểm Kiểm thử hộp đen

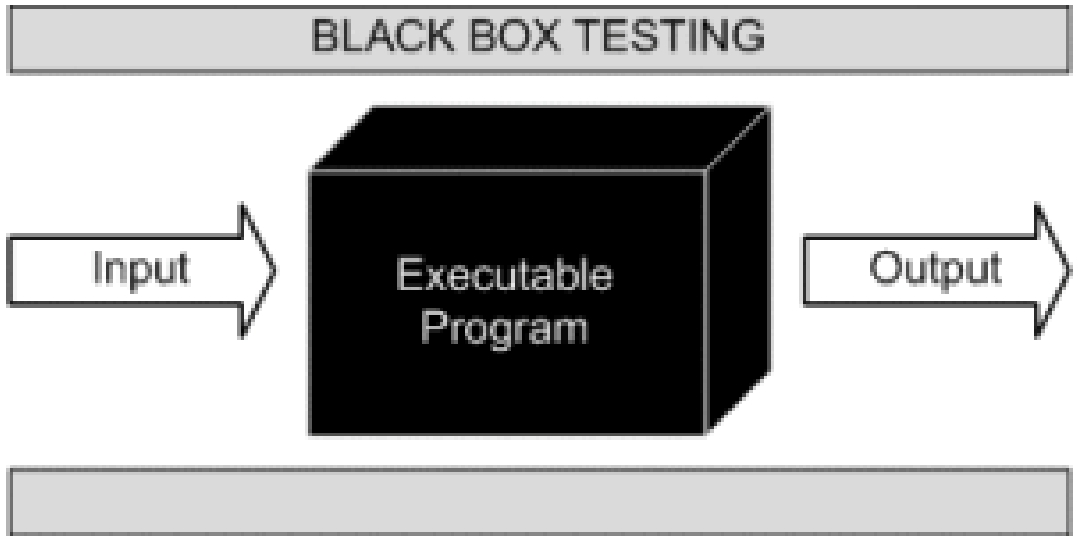
- Các kiểm thử viên được thực hiện từ quan điểm của người dùng.
- Không có “mối ràng buộc” nào với code.
- Tester có thể không phải IT chuyên nghiệp, không cần phải biết ngôn ngữ lập trình hoặc làm thế nào các phần mềm đã được thực hiện.
- Các tester có thể là một cơ quan độc lập cho phép một cái nhìn khách quan và tránh sự phát triển thiên vị.
- Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

Nhược điểm Kiểm thử hộp đen

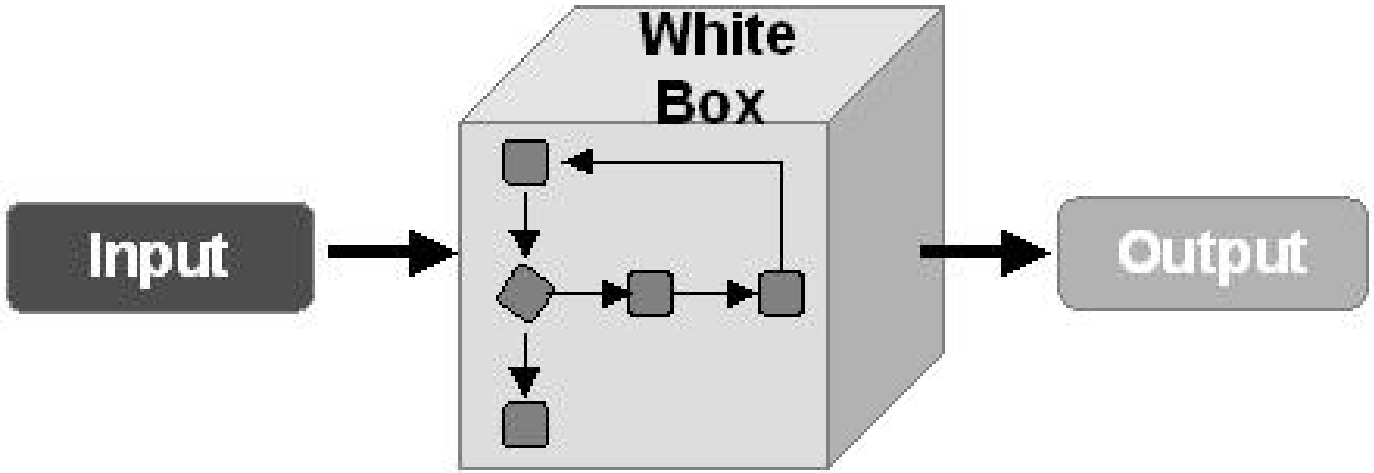
- Dữ liệu đầu vào yêu cầu một khối lượng mẫu (sample) khá lớn.
- Nhiều dự án không có thông số rõ ràng thì việc thiết kế test case rất khó và do đó khó viết kịch bản kiểm thử do cần xác định tất cả các yếu tố đầu vào, và thiếu cả thời gian cho việc tập hợp này.

- Chỉ có một số nhỏ các đầu vào có thể được kiểm tra.
- Kiểm thử black box được xem như "là bước đi trong mê cung tối đen mà không mang đèn pin” bởi vì tester không biết phần mềm đang test đã được xây dựng như thế nào. Có nhiều trường hợp khi một tester viết rất nhiều trường hợp test để kiểm tra một số thứ có thể chỉ được test bằng một trường hợp test và hoặc một vài phần cuối cùng không được test hết.

- Các lỗi điển hình tìm thấy bởi Kiểm thử hộp đen
  - Chức năng không chính xác hoặc thiếu.
  - Lỗi giao diện.
  - Lỗi trong cấu trúc dữ liệu hoặc truy cập cơ sở dữ liệu bên ngoài.
  - Lỗi hành vi của ứng dụng.



KIỂM THỬ HỘP TRẮNG



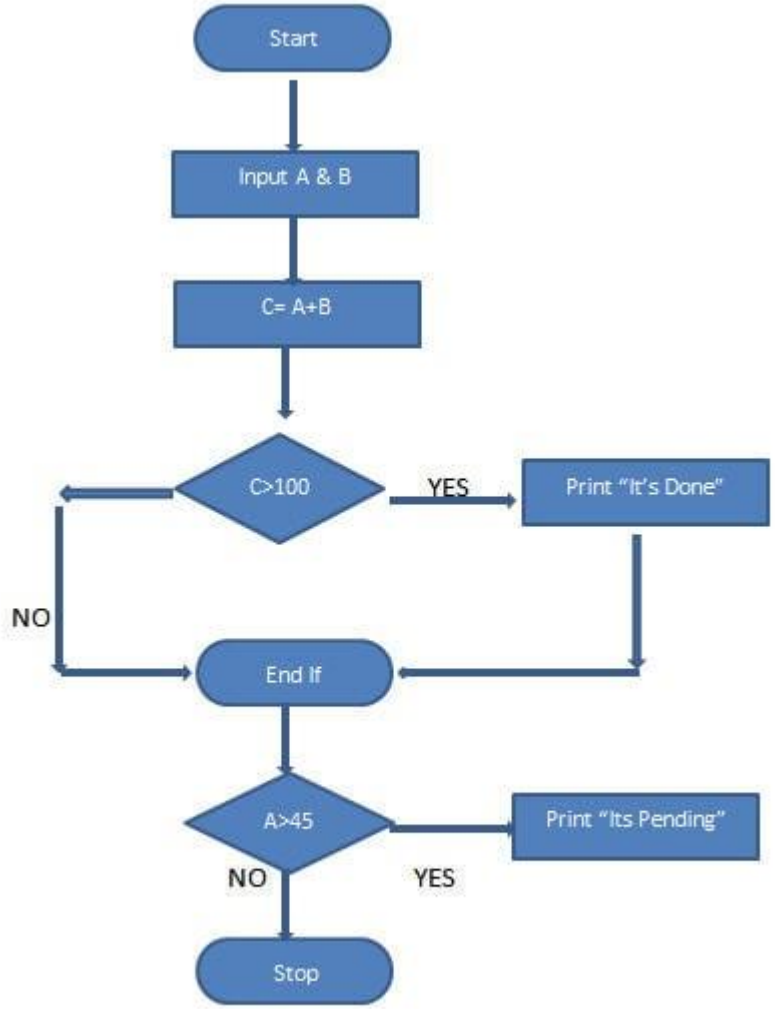
WhiteBox Testing - Kiểm thử hộp trắng

- Kiểm thử Hộp Trắng (còn gọi là Code-Based Testing hoặc Structural Testing) là một phương pháp kiểm thử phần mềm trong đó tester biết về cấu trúc nội bộ / thiết kế.
- Kiểm thử hộp trắng bao gồm phân tích dòng dữ liệu, điều khiển dòng, dòng thông tin, mã thực hành, ngoại lệ và những lỗi trình bày trong hệ thống để kiểm tra những hành động của phần mềm không được định hướng trước.

- Đối tượng được kiểm thử là 1 thành phần phần mềm như là 1 hàm chức năng, 1 module chức năng, 1 phân hệ chức năng…
- Phương pháp Kiểm tra Hộp trắng áp dụng cho các mức độ kiểm tra phần mềm sau đây:
  - ✔ Unit Testing(Kiểm thử đơn vị): Để kiểm tra đường dẫn trong một đơn vị.
  - ✔ Integration Testing(Test tích hợp): Để kiểm tra đường dẫn giữa các đơn vị.
  - ✔ System Testing(Test hệ thống): Để kiểm tra các đường dẫn giữa các hệ thống con.
    - → Tuy nhiên, nó là chủ yếu áp dụng cho các kiểm thử đơn vị .

Đối tượng Kiểm thử hộp trắng

- Đối tượng được kiểm thử là 1 thành phần phần mềm (TPPM).TPPM có thể là 1 hàm chức năng, 1 module chức năng, 1 phân hệ chức năng…



Path Coverage

Các bước Kiểm thử hộp trắng

- Bước 1: Đọc và hiểu mã nguồn(source code).
- Bước 2: Dùng kỹ thuật định nghĩa các testcase xác định(sẽ giới thiệu sau) để định nghĩa các testcase. Định nghĩa mỗi testcase là xác định 3 thông tin sau :
  - Giá trị dữ liệu nhập để TPPM xử lý (hoặc hợp lệ hoặc không hợp lệ).
  - Trạng thái của thành phần phần mềm(TPPM) cần có để thực hiện testcase.
  - Giá trị dữ liệu xuất mà TPPM phải tạo được.

- Bước 3: Sử dụng testcase đã xác định ở bước 2 và thực thi test trong code (không cần thực thi chương trình, vì thực hiện white box testing sẽ sử dụng framework nào đó hỗ trợ (Ví dụ như test kiểu debug).
- Bước 4: So sánh kết quả thu được với kết quả kỳ vọng trong từng testcase, từ đó lập báo cáo về kết quả kiểm thử.

Biểu mẫu Test case:

ID	Test Case Description	Test Case Procedure	Expected Result	Test Result
1	*<Brief description of this case: what is tested?>* Test username and password = empty	*<Describe steps to perform this case>* 1. Username and Pasword textbox = empty 2. Click "Log in" button	*<Describe results which meet customer's requirement>* Message displays ' Username and Password are required'	Pass
2	<Test case 2>			Fail

Ưu điểm Kiểm thử hộp trắng

- Test có thể bắt đầu ở giai đoạn sớm hơn, không cần phải chờ đợi cho GUI để có thể test.
- Test kỹ càng hơn, có thể bao phủ hầu hết các đường dẫn.
- Thích hợp trong việc tìm kiếm lỗi và các vấn đề trong mã lệnh.
- Cho phép tìm kiếm các lỗi ẩn bên trong.
- Các lập trình viên có thể tự kiểm tra.
- Giúp tối ưu việc mã hoá.
- Do yêu cầu kiến thức cấu trúc bên trong của phần mềm, nên việc kiểm soát lỗi tối đa nhất.

Nhược điểm Kiểm thử hộp trắng

- Vì các bài kiểm tra rất phức tạp, đòi hỏi phải có các nguồn lực có tay nghề cao, với kiến thức sâu rộng về lập trình thực hiện.
- Không thể kiểm thử được hết các đường dẫn.
- Kết quả có độ chính xác không cao do con người tự tính toán nên có thể dẫn đến nhầm lẫn.
- Chương trình sai do thiếu logic.
- Dễ bị miss các case đặc biệt nếu vòng lặp thuật toán phức tạp
- Không thể đảm bảo rằng chương trình đã tuân theo đặc tả và cover hết các trường hợp.

Các lỗi điển hình được tìm thấy bởi Kiểm thử hộp trắng

- Lỗi cú pháp ví dụ như thiếu dấu kết thúc một câu lệnh, một số ngôn ngữ từ khoá phân biệt chữ hoa, chữ thường thì lại gõ chữ hoa, v.v. gọi nôm na là lỗi chính tả.

```
class SV
{
   int ma sv; //Lỗi viết sai tên biến, khai báo biến không được có khoảng trắng
   void nhap() {
     for (i=0; i<=2; i++) // lỗi chưa khai báo biến i
        cin>>i;
   }
} //lỗi thiếu dấu ;

void main //lỗi thiếu dấu () sau hàm main()
{
}
```

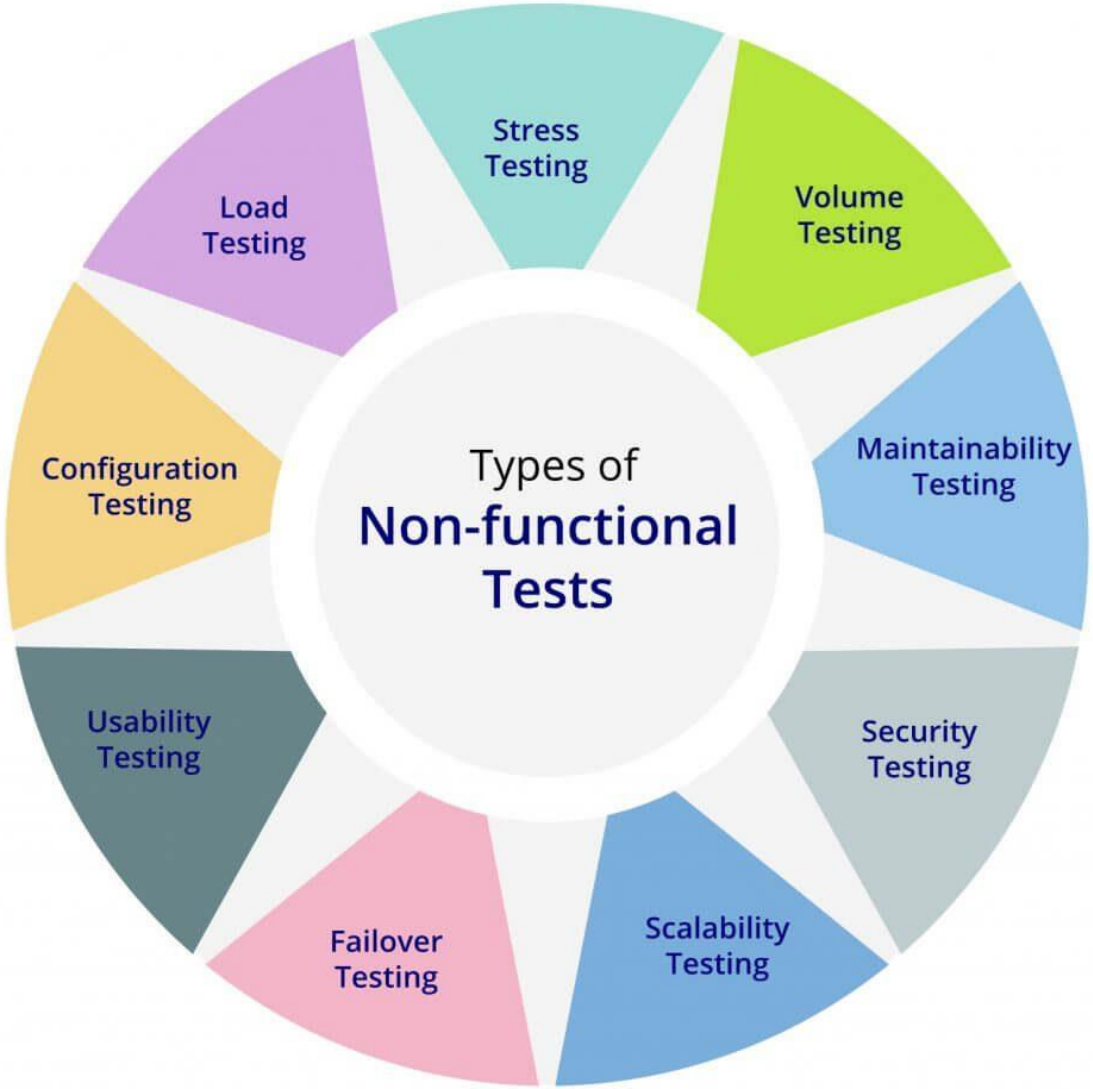
- Lỗi thực thi thường xảy ra do người lập trình viết code ẩu, không lường hết các trường hợp xảy ra, khiến chương trình đang chạy thì bị lỗi treo màn hình, thoát khỏi chương trình hoặc thoát luôn chương trình, v.v. Lỗi này có thể dễ dàng phát hiện bằng cách Debug.

```
float s=0;
for (int i=0; i<=10; i++)
s=s+1/(i-3);
```

- Lỗi logic do hiểu đặc tả yêu cầu sai, do tư duy sai, thuật toán sai dẫn đến sai kết quả sai. Ví dụ sinh viên không biết viết thuật toán tìm ước số chung lớn nhất, không biết viết công thức nghiệm giải phương trình bậc 2, thực hiện sai giải thuật.

KIỂM THỬ PHI CHỨC NĂNG

NON-FUNCTIONAL TESTING



- Một số câu hỏi trước khi kiểm thử phi chức năng
  - Ứng dụng làm việc trong điều kiện bình thường như thế nào?
  - Ứng dụng hành xử như thế nào khi quá nhiều người dùng đăng nhập đồng thời?
  - Ứng dụng có thể chịu được tải lớn không?
  - Ứng dụng bảo mật tới mức nào?
  - Ứng dụng có thể phục hồi từ bất kì sự cố nào hay không?

Non-Functional Testing - Kiểm thử phi chức năng

- Kiểm thử phi chức năng (non-functional) được định nghĩa là một loại kiểm thử phần mềm để kiểm tra các khía cạnh phi chức năng (hiệu suất, khả năng sử dụng, độ tin cậy, v.v.) của ứng dụng phần mềm. Nó được thiết kế để kiểm tra sự sẵn sàng của một hệ thống theo các tham số không thuộc về chức năng và không bao giờ được giải quyết bằng kiểm thử chức năng.

- Một số loại của Kiểm thử phi chức năng
  - Kiểm thử tải trọng(Load Testing).
  - Kiểm thử về áp lực(Stress Testing).
  - Kiểm thử khối lượng(Volume Testing).
  - Kiểm thử bảo mật(Security Testing).

- Kiểm thử tải trọng(Load Testing): Là kiểm tra hệ thống bằng cách tăng tải liên tục và đều đặn cho hệ thống cho đến khi đạt đến giới hạn ngưỡng.

  Ví dụ: Một trang web của Airline đã có hiện tượng chậm và xuất hiện trạng thái loading khi có hơn 10000 người dùng trong đợt ưu đãi mùa tết.

- Kiểm thử sức chịu đựng(Stress Testing): Là Kiểm tra tập trung vào các trạng thái tới hạn, các “điểm chết”, các tình huống bất thường...

  Ví dụ: Một trang web của Airline đã bị down khi có 100000 lượt truy cập vào website.

- Kiểm thử khối lượng(Volume Testing): là một thử nghiệm hiệu suất phi chức năng, nơi mà phần mềm phải chịu một lượng lớn dữ liệu.

  Ví dụ: Thử nghiệm tình trạng của trang web âm nhạc khi có hàng triệu người dùng tải bài hát xuống.

- Kiểm thử bảo mật(Security Testing): là việc tìm kiếm tất cả các lỗ hổng có thể và điểm yếu của hệ thống mà có thể dẫn đến mất thông tin trong tay nhân viên hoặc người ngoài của tổ chức. Security Testing rất quan trọng trong ngành công nghiệp CNTT để bảo vệ dữ liệu của tất cả các phương tiện.

  Ví dụ: Người dùng đăng nhập và đăng xuất ra ngoài tuy nhiên cookie và session time vẫn còn điều này có nghĩa là Hacker có thể sử dụng cookie này để thực hiện đăng nhập vào website và thao tác được các tác vụ thay đổi.

Một số thông số trong kiểm thử phi chức năng

- Security (Bảo mật): Tham số xác định cách hệ thống được bảo vệ an toàn trước các cuộc tấn công có chủ ý và đột ngột từ các nguồn bên trong và bên ngoài.
- Reliability (Độ tin cậy): Mức độ mà bất kỳ hệ thống phần mềm nào liên tục thực hiện các chức năng được chỉ định mà không gặp sự cố.
- Survivability (Khả năng sống sót): Tham số kiểm tra rằng hệ thống phần mềm tiếp tục hoạt động và tự phục hồi trong trường hợp lỗi hệ thống.
- Availability (Tính sẵn có): Tham số xác định mức độ mà người dùng có thể phụ thuộc vào hệ thống trong quá trình hoạt động.

- Usability (Khả năng sử dụng): Người dùng có thể dễ dàng học hỏi, vận hành, chuẩn bị đầu vào và đầu ra thông qua tương tác với một hệ thống.
- Scalability (Khả năng mở rộng): Thuật ngữ này đề cập đến mức độ mà bất kỳ ứng dụng phần mềm nào cũng có thể mở rộng khả năng xử lý của nó để đáp ứng nhu cầu gia tăng.
- Interoperability (Khả năng tương tác): Tham số phi chức năng này kiểm tra giao diện hệ thống phần mềm tương tác với các hệ thống phần mềm khác.
- Efficiency (Tính hiệu quả): Tham số này xác định bất kỳ hệ thống phần mềm nào cũng có thể xử lý dung lượng, số lượng và thời gian đáp ứng.

- Flexibility (Tính linh hoạt): Thuật ngữ này đề cập đến sự dễ dàng mà ứng dụng có thể hoạt động trong các cấu hình phần cứng và phần mềm khác nhau. Giống như RAM tối thiểu, yêu cầu CPU.
- Portability (Tính di động): Tính linh hoạt của phần mềm để chạy trên các môi trường phần cứng hoặc phần mềm khác nhau.
- Reusability (Tái sử dụng): Nó đề cập đến một phần của hệ thống phần mềm có thể được chuyển đổi để sử dụng trong một ứng dụng khác.

Ưu điểm Kiểm thử phi chức năng

- Xác định cách hệ thống được bảo vệ an toàn trước các cuộc tấn công có chủ ý và đột ngột từ các nguồn bên trong và bên ngoài.
- Đảm bảo hệ thống phần mềm tiếp tục hoạt động và tự phục hồi trong trường hợp lỗi hệ thống.
- Đảm bảo ứng dụng phần mềm nào cũng có thể mở rộng khả năng xử lý của nó để đáp ứng nhu cầu gia tăng.
- Đảm bảo hệ thống phần mềm nào cũng có thể xử lý dung lượng, số lượng và thời gian đáp ứng.

Nhược điểm Kiểm thử phi chức năng

- Kết quả sau khi kiểm thử phi chức năng thường không đưa ra số chính xác mà chỉ là tương đối.
- Vì các bài kiểm tra rất phức tạp, đòi hỏi phải có các nguồn lực có tay nghề cao, với kiến thức sâu rộng về mạng, bảo mật, lập trình.
- Vì phương pháp thử nghiệm này liên quan chặt chẽ với ứng dụng đang được test, nên các công cụ để phục vụ cho mọi loại triển khai / nền tảng có thể không sẵn có.