|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **KHOA CÔNG NGHỆ THÔNG TIN**  **---------------------------------------**  **Logo%20HaUI%20ban%20chuan**  BÁO CÁO BTL THUỘC HỌC PHẦN:  ĐẢM BẢO CHẤT LƯỢNG PHẦN MỀM  **ĐỀ TÀI: XÁC ĐỊNH CÁC CA KIỂM THỬ VÀ DỮ LIỆU KIỂM THỬ SỬ DỤNG KĨ THUẬT KIỂM THỬ HỘP ĐEN TRONG MODULE MÃ HOÁ VĂN BẢN**   |  |  | | --- | --- | | **GVHD:** | ***Ts. Vũ Đình Minh*** | | **Sinh viên:** |  | |  | ***Trần Lê Thịnh 2017606165*** | |  | ***Phạm Văn Thư 2017605033*** | |  | ***Vũ Đức Vượng 2017604374*** | | **Nhóm:** | ***1*** | | **Lớp:** | ***20201IT6008002*** | | **Khoá:** | ***12*** |   ***Hà Nội, năm 2020*** |

Mục lục

[Chương 1. Tổng quan về đảm bảo chất lượng phần mềm 5](#_Toc60245745)

[1.1 Phần mềm và chất lượng phần mềm 5](#_Toc60245746)

[1.1.1. Phần mềm 5](#_Toc60245747)

[1.1.2. Chất lượng phần mềm 6](#_Toc60245748)

[1.1.3. Đảm bảo chất lượng phần mềm 8](#_Toc60245749)

[1.2 Thách thức của chất lượng phần mềm 9](#_Toc60245750)

[*1.3* Các yếu tố ảnh hưởng lên chất lượng phần mềm 14](#_Toc60245751)

[1.3.1. Các yếu tố chất lượng phần mềm theo mô hình cây của McCall 14](#_Toc60245754)

[1.4 Các yếu tố chất lượng phần mềm theo ISO 9126, 2001: 15](#_Toc60245755)

[*1.5* Kiến trúc của hệ thống đảm bảo chất lượng phần mềm 18](#_Toc60245756)

[Chương 2. Các kĩ thuật kiểm thử và các ca kiểm thử 21](#_Toc60245757)

[2.1 Các chiến lược kiểm thử 21](#_Toc60245759)

[2.1.1. Khái niệm chiến lược kiểm thử 21](#_Toc60245760)

[2.1.2. Mô hình chiến lược tổng thể 23](#_Toc60245761)

[2.1.3. Chiến lược kiểm thử hệ thời gian thực 24](#_Toc60245762)

[2.1.4. Kiểm thử Alpha và Beta 25](#_Toc60245763)

[2.1.5. Kiểm thử so sánh 26](#_Toc60245764)

[2.2 Các kĩ thuật kiểm thử 27](#_Toc60245765)

[2.2.1. Kiểm thử hộp đen (Black-box Testing) 27](#_Toc60245767)

[2.2.2. Kiểm thử hộp trắng 29](#_Toc60245768)

[2.3 Các ca kiểm thử 30](#_Toc60245769)

[2.3.1. Khái niệm các ca kiểm thử 30](#_Toc60245771)

[2.3.2. Vấn đề khi thiết kế ca kiểm thử 31](#_Toc60245772)

[Chương 3. Các kĩ thuật thiết kế ca kiểm thử 33](#_Toc60245773)

[3.1 Kĩ thuật bao phủ câu lệnh (Statement Coverge) 33](#_Toc60245775)

[3.1.1. Kỹ thuật bao phủ quyết định 33](#_Toc60245776)

[3.1.2. Kỹ thuật bao phủ điều kiện (Condition Coverage) 34](#_Toc60245777)

[3.1.3. Kỹ thuật bao phủ quyết định/ điều kiện (Decision/Condition coverage) 34](#_Toc60245778)

[3.1.4. Kỹ thuật bao phủ đa điều kiện (Multiple Condition Coverage) 34](#_Toc60245779)

[3.1.5. Kỹ thuật bao phủ quyết định/ điều kiện (Decision/Condition coverage) 34](#_Toc60245780)

[3.1.6. Kỹ thuật bao phủ đa điều kiện (Multiple Condition Coverage) 35](#_Toc60245781)

[3.2 Kĩ thuật phân lớp tương đương 35](#_Toc60245782)

[3.3 Kỹ thuật phân tích giá trị biên (Boundary Value Analysis) 37](#_Toc60245783)

[3.4 Kỹ thuật đồ thị nguyên nhân – kết quả (Cause – Effect Graphing) 38](#_Toc60245784)

[3.5 Kỹ thuật đoán lỗi (Error Guessing) 39](#_Toc60245785)

[Chương 4. Kiểm thử hộp đen 41](#_Toc60245786)

[4.1 Khái niệm 41](#_Toc60245788)

[4.2 Ưu điểm 42](#_Toc60245789)

[4.3 Nhược điểm 43](#_Toc60245790)

[4.4 Flow của Black-box testing 43](#_Toc60245791)

[4.5.1. Nội dung công việc trong công đoạn test 44](#_Toc60245796)

[4.5.2. Flow của plan và kế hoạch test 45](#_Toc60245797)

[4.5 Phương pháp truy xuất và tính cần thiết của quan điểm test 46](#_Toc60245798)

[4.5.1. Danh sách quan điểm test 46](#_Toc60245799)

[4.5.2. Phương pháp tạo danh sách quan điểm test\ 47](#_Toc60245800)

[4.6 Phương pháp kiểm thử hộp đen 49](#_Toc60245801)

[Chương 5. Xác định các ca kiểm thử và kỹ thuật kiểm thử cho module mã hóa văn bản 52](#_Toc60245802)

[5.1 Giới thiệu bài toán 52](#_Toc60245804)

[5.2 Thiết kế các ca kiểm thử 52](#_Toc60245805)

[Tài liệu tham khảo 54](#_Toc60245806)

# Tổng quan về đảm bảo chất lượng phần mềm

## Phần mềm và chất lượng phần mềm

### Phần mềm

#### Phần mềm dưới góc nhìn của người sử dụng:

##### Chương trình thực thi được trên máy tính hoặc các thiết bị chuyên dụng khác

##### Nhằm hỗ trợ cho các nhà chuyên môn trong từng lĩnh vực chuyên ngành thực hiện tốt hơn các thao tác nghiệp vụ của mình.

#### Phần mềm dưới góc nhìn của chuyên viên Tin học: đây là một hệ thống bao gồm 3 thành phần cơ bản:

##### Thành phần giao tiếp

##### Thành phần xử lý

##### Thành phần lưu trữ.

**Tóm lại:** Phần mềm là một tập quy tắc xử lý thể hiện thành chương trình (mã lệnh+dữ liệu) được cài đặt vào phần cứng phù hợp để tự thực hiện một vài công việc thay con người. Các mô tả cho chương trình (chức năng, giao diện,cách sử dụng, ràng buộc,..), để nhiều người cùng hợp tác với nhau làm ra & sử dụng phần mềm: phân tích viên, thiết kế viên, lập trình viên, kiểm thử viên, người sử dụng, admin,...

#### Phần mềm có các đặc điểm sau:

##### Không có tính chất vật lý (vô hình, ...), vd: gmail, search engines, cloud,…

##### Không bị hao mòn như phần cứng*,* chỉ bị lạc hậu !

##### Sao chép được !!!

##### Sự thay đổi linh hoạt là ưu thế của PM so với phần cứng. Do đó, cách làm ra phần mềm cũng khác:

###### Dựa trên sự tư duy để sáng tác ra phần mềm

###### Phần mềm được sử dụng qua các versions

### Chất lượng phần mềm

#### Chất lượng

##### Chất lượng: Sự phù hợp với đặc điểm kỹ thuật: Chất lượng được định nghĩa là vấn đề của sản phẩm và các dịch vụ có các đặc tính có thể đo lường thỏa mãn một đặc điểm kỹ thuật cố định - nghĩa là, sự phù hợp với một thông số kỹ thuật được xác định trước.

##### Đáp ứng nhu cầu của khách hàng: Chất lượng được xác định độc lập với bất kỳ đặc điểm nào có thể đo lường được. Nghĩa là, chất lượng được định nghĩa là khả năng của sản phẩm hoặc dịch vụ để đáp ứng mong đợi của khách hàng - rõ ràng hay không.

#### Chất lượng phần mềm: **Chất lượng sản phẩm phần mềm** (*Software quality*) là khả năng đáp ứng toàn diện nhu cầu của người dùng về tính năng cũng như công dụng được nêu ra một cách tường minh hoặc không tường minh trong những ngữ cảnh xác định.

#### Có rất nhiều định nghĩa về chất lượng phần mềm được đưa ra bởi các tổ chức, cá nhân khác nhau:

#### Theo IEEE(1991):

##### **Định nghĩa 1:** Chất lượng phần mềm là một mức độ mà một hệ thống, thành phần hệ thống hay tiến trình đáp ứng được yêu cầu đã được đặc tả.

##### **Định nghĩa 2:** Chất lượng phần mềm là mức độ mà một hệ thống, thành phần hệ thống hay tiến trình đáp ứng được yêu cầu và sự mong đợi của khách hàng hay người sử dụng.

Theo quan điểm thứ nhất của IEEE: chúng ta sẽ bị phụ thuộc quá nhiều vào tài liệu đặc tả của yêu cầu, dẫn đến nếu việc xác định yêu cầu bị sai, thiếu thì một phần mềm được làm đúng với đặc tả chưa chắc đã là một phần mềm có chất lượng.

Theo quan điểm thứ hai của IEEE: khách hàng đôi khi không có kiến thức về công nghệ, họ có thể đưa ra những mong muốn hết sức vô lý và có thể thay đổi yêu cầu với phần mềm nhiều lần, thậm chí thay đổi ngay trong giai đoạn cuối. Điều này gây nhiều khó khăn cho việc phát triển phần mềm.

#### Theo Pressman: Chất lượng phần mềm là sự phù hợp của các yêu cầu cụ thể về hiệu năng và chức năng, các tiêu chuẩn phát triển phần mềm được ghi lại rõ ràng bằng tài liệu với các đặc tính ngầm định của tất cả các phần mềm được phát triển chuyên nghiệp.

Định nghĩa của Pressman đề xuất ba yêu cầu với chất lượng phần mềm phải được đáp ứng khi phát triển phần mềm:

#### Các yêu cầu chức năng rõ ràng là nhân tố chính quyết định chất lượng đầu ra của phần mềm.

#### Các tiêu chuẩn chất lượng phần mềm sẽ được nói đến trong hợp đồng.

#### Các đặc tính ngầm định cần được đáp ứng trong quá trình phát triển cho dù không được nói đến rõ ràng trong hợp đồng.

Có nhiều khái niệm về chất lượng lượng phần mềm. Về cơ bản, chất lượng phần mềm:

#### Là sự thỏa mãn yêu cầu (Crosby,1979), sản phẩm có đủ những đặc tính làm thỏa mãn khách hàng để tạo ra sự hài lòng về sản phẩm.

#### Không có thiếu sót trong sản phẩm (Juran, 1988):

##### Mức độ mà một sản phẩm (hệ thống, một thành phần hay một xử lý) làm thỏa mãn cho các yêu cầu đã được định nghĩa;

##### Mức độ mà một sản phẩm làm thỏa mãn cho khách hàng, cho nhu cầu của người sử dụng hoặc cho các kỳ vọng về nó (IEEE, 1991).

Chất lượng phần mềm dưới góc nhìn của người sử dụng: Mức độ làm thỏa mãn cho yêu cầu được đặc tả (requirements) hoặc mong đợi (needs) đối với phần mềm, với chi phí và thời gian hợp lý:

##### Tính đúng đắn: đầy đủ, chính xác;

##### Tính tiện dụng: dễ học; dễ sử dụng; giao diện trực quan; tự nhiên;

##### Tính hiệu quả: tối ưu sử dụng CPU; Tối ưu sử dụng bộ nhớ; Tối ưu sử dụng thiết bị;

##### Tính tương thích: Import/Export dữ liệu; tính tương tác;

##### Tính tiến hóa: một trong các tính chất quan trọng nhất được quan tâm xem xét trong ngành Công nghệ Phần mềm. Sự hoàn thiện dần và gắn liền với sự tiến hóa của phần cứng;

##### Chất lượng phần mềm dưới góc nhìn của người phát triển:

##### Dễ làm & dễ cập nhật: sử dụng lại, hợp chuẩn, tiếp nhận công nghệ mới, mềm dẻo;

##### An toàn

### Đảm bảo chất lượng phần mềm

#### Đảm bảo chất lượng: Thiết lập một tập hợp các họat động có chủ đích và có hệ thống nhằm mang lại sự tin tưởng sẽ đạt được chất lượng của sản phẩm đòi hỏi.

#### Đảm bảo chất lượng phần mềm:

##### Đảm bảo dự án phần mềm sẽ hoàn thành đúng đặc tả, theo chuẩn mực định trước và các chức năng đòi hỏi, không có hỏng hóc và các vấn đề tiềm ẩn.

##### SQA điều khiển và cải tiến tiến trình phát triển phần mềm ngay từ khi dự án bắt đầu. Nó có tác dụng “phòng ngừa” cái xấu, cái kém chất lượng.

##### Mục tiêu cuối cùng của SQA là thỏa mãn khách hàng (costumer satisfaction):

##### Thời gian;

##### Ngân sách;

##### Chất lượng.

## Thách thức của chất lượng phần mềm

Sự phức tạp lớn cũng như khả năng vô hình của phần mềm, cùng với sự khác biệt về các đặc tính sản phẩm so với sản phẩm công nghiệp, làm cho việc phát triển phương pháp luận SQA và triển khai thành công nó là một thách thức chuyên môn cao.

Không nhà phát triển nào sẽ tuyên bố rằng phần mềm của họ không có lỗi, vì các nhà sản xuất phần cứng máy tính lớn sẽ không làm như vậy. Sự từ chối này thực sự phản ánh sự khác biệt cơ bản giữa phần mềm và các sản phẩm công nghiệp khác. Những khác biệt này có thể được phân loại như sau:

##### **Độ phức tạp của sản phẩm:** Độ phức tạp của sản phẩm có thể được đo lường bằng số lượng chế độ hoạt động mà sản phẩm cho phép. Một sản phẩm công nghiệp, ngay cả một máy tiên tiến, không cho phép nhiều hơn một vài nghìn chế độ hoạt động, được tạo ra bởi sự kết hợp của các thiết lập máy khác nhau của nó. Nhìn vào một gói phần mềm điển hình, người ta có thể tìm thấy hàng triệu khả năng vận hành phần mềm. Đảm bảo rằng vô số các khả năng hoạt động được xác định và phát triển một cách chính xác là một thách thức lớn đối với ngành công nghiệp phần mềm.

##### **Khả năng hiển thị của sản phẩm:** Trong khi các sản phẩm công nghiệp có thể nhìn thấy được thì các sản phẩm phần mềm là vô hình. Hầu hết các lỗi trong một sản phẩm công nghiệp có thể được phát hiện trong quá trình sản xuất. Hơn nữa, sự vắng mặt của một bộ phận nào đó trong một sản phẩm công nghiệp, theo quy luật, rất dễ thấy (hãy tưởng tượng chiếc xe mới của bạn bị thiếu một cánh cửa). Tuy nhiên, các khiếm khuyết trong sản phẩm phần mềm (cho dù được lưu trữ trên đĩa hoặc CD) là vô hình, vì thực tế là các phần của gói phần mềm có thể không có ngay từ đầu.

##### **Quá trình sản xuất và phát triển sản phẩm:** Bây giờ chúng ta hãy xem xét các giai đoạn (Bảng 1.1 ) mà tại đó khả năng phát hiện ra các khuyết tật trong một sản phẩm công nghiệp và sản phẩm phần mềm có thể phát sinh:

Bảng 1.1. Các giai đoạn sản xuất và phát triển sản phẩm

|  |  |  |
| --- | --- | --- |
| **Nội dung** | **Sản phẩm công nghiệp** | **Sản phẩm phần mềm** |
| Phát triển sản phẩm | Trong giai đoạn này, các nhà thiết kế và nhân viên đảm bảo chất lượng (QA) sẽ kiểm tra và thử nghiệm nguyên mẫu sản phẩm để phát hiện ra các khuyết tật của nó. | Trong giai đoạn này, nỗ lực của các nhóm phát triển và các chuyên gia đảm bảo chất lượng phần mềm được hướng tới việc phát hiện các lỗi cố hữu của sản phẩm. Vào cuối giai đoạn này, một nguyên mẫu đã được phê duyệt, sẵn sàng để tái sản xuất, sẽ có sẵn. |
| Lập kế hoạch sản xuất sản phẩm | Trong giai đoạn này, quá trình sản xuất và các công cụ được thiết kế và chuẩn bị. Trong một số sản phẩm, cần có một dây chuyền sản xuất đặc biệt để được thiết kế và xây dựng. Do đó, giai đoạn này cung cấp thêm cơ hội để kiểm tra sản phẩm, điều này có thể phát hiện ra những khiếm khuyết đã “thoát khỏi” các đánh giá và thử nghiệm được thực hiện trong giai đoạn phát triển. | Giai đoạn này không bắt buộc đối với quá trình sản xuất phần mềm, vì quá trình sản xuất các bản sao phần mềm và in hướng dẫn sử dụng phần mềm được tiến hành tự động. Điều này áp dụng cho bất kỳ sản phẩm phần mềm nào, cho dù số lượng bản sao nhỏ, như trong phần mềm tùy chỉnh hay lớn, như trong các gói phần mềm được bán cho công chúng. |
| Sản xuất | Ở giai đoạn này, các thủ tục QA được áp dụng để phát hiện lỗi của chính sản phẩm. Các khiếm khuyết trong sản phẩm được phát hiện trong giai đoạn đầu tiên của quá trình sản xuất thường có thể được sửa chữa bằng cách thay đổi thiết kế hoặc vật liệu của sản phẩm hoặc trong công cụ sản xuất, theo cách loại bỏ những khiếm khuyết đó trong các sản phẩm được sản xuất trong tương lai. | Như đã đề cập trước đây, việc sản xuất phần mềm chỉ giới hạn trong việc sao chép sản phẩm và in bản sao của sách hướng dẫn sử dụng phần mềm. Do đó, kỳ vọng phát hiện các khuyết tật khá hạn chế trong giai đoạn này. |

Sự khác biệt ảnh hưởng đến việc phát hiện lỗi trong sản phẩm phần mềm so với các sản phẩm công nghiệp khác được thể hiện trong Bảng 1.2

Cần lưu ý rằng các bộ phận quan trọng của máy móc tiên tiến cũng như của máy gia dụng và các sản phẩm khác bao gồm các thành phần phần mềm nhúng (thường được gọi là “phần sụn”) được tích hợp vào sản phẩm. Các thành phần phần mềm này (phần sụn) có cùng đặc điểm của các sản phẩm phần mềm được đề cập ở trên. Sau đó, so sánh được hiển thị ở trên thực sự phải là của các sản phẩm phần mềm so với các sản phẩm công nghiệp khác và các thành phần không phải phần mềm của các sản phẩm công nghiệp bao gồm phần sụn. Sau đây, khi đề cập đến phần mềm, chúng ta sẽ có nghĩa là các sản phẩm phần mềm cũng như phần sụn. Sự khác biệt cơ bản giữa quá trình phát triển và sản xuất liên quan đến các sản phẩm phần mềm và các sản phẩm công nghiệp khác đảm bảo việc tạo ra một phương pháp luận SQA khác cho phần mềm. Nhu cầu về các công cụ và phương pháp đặc biệt cho ngành công nghiệp phần mềm được phản ánh trong các ấn phẩm chuyên môn cũng như trong các tiêu chuẩn đặc biệt dành cho SQA, chẳng hạn như ISO 9000-3, “Hướng dẫn áp dụng ISO 9001 để phát triển, cung cấp và bảo trì phần mềm ”. Điểm này được hỗ trợ bởi thực tế là ISO chưa chuẩn bị các hướng dẫn mục tiêu cho các ngành khác và các hướng dẫn mục tiêu duy nhất khác đã được chuẩn bị cho các dịch vụ (ISO 9004-2, “Các yếu tố quản lý chất lượng và hệ thống chất lượng: Hướng dẫn cho dịch vụ” ). Sự phức tạp lớn cũng như khả năng tàng hình của phần mềm, cùng với các đặc tính sản phẩm khác, làm cho việc phát triển phương pháp luận SQA và triển khai thành công nó là một thách thức chuyên môn cao.

Tính độc đáo của quy trình phát triển phần mềm:

##### Độ phức tạp cao so với các sản phẩm công nghiệp khác

##### Tàng hình của sản phẩm

|  |  |  |
| --- | --- | --- |
| **Đặc tính** | **Sản phẩm phần mềm** | **Sản phẩm công nghệ khác** |
| Độ phức tạp của sản phẩm | Thông thường, sản phẩm rất phức tạp cho phép nhiều tùy chọn hoạt động | Mức độ phức tạp thấp hơn nhiều, cho phép tối đa vài nghìn tùy chọn hoạt động |
| Khả năng hiển thị của sản phẩm | Sản phẩm vô hình, không thể phát hiện ra các khiếm khuyết hoặc thiếu sót bằng mắt (ví dụ: đĩa hoặc đĩa CD lưu trữ phần mềm) | Sản phẩm có thể nhìn thấy được, cho phép phát hiện hiệu quả các khuyết tật bằng mắt |
| Bản chất của sự phát triển | Cơ hội phát hiện các khuyết tật và quy trình sản xuất chỉ phát sinh trong một giai đoạn, đó là phát triển sản phẩm | Cơ hội phát hiện các khuyết tật phát sinh trong tất cả các giai đoạn phát triển và sản xuất:  ■ Phát triển sản phẩm  ■ Lập kế hoạch sản xuất sản phẩm  ■ Sản xuất |
| Bảng 1.2. Các yếu tố ảnh hưởng đến việc phát hiện lỗi trong sản phẩm phần mềm so với các sản phẩm công nghiệp khác | | |

##### Cơ hội để phát hiện khuyết tật (“lỗi”) chỉ giới hạn ở sản phẩm giai đoạn phát triển

## Các yếu tố ảnh hưởng lên chất lượng phần mềm



### Các yếu tố chất lượng phần mềm theo mô hình cây của McCall

Từ những khái niệm cơ về chất lượng phần mềm:

##### Chất lượng phần mềm là mức độ mà nó đáp ứng được yêu cầu chức năng hoặc đặc tả yêu cầu (hoặc là mức độ mà một hệ thống, thành phần, hay tiến trình đáp ứng được nhu cầu/mong muốn của khách hàng/người dùng.)

##### Đảm bảo chất lượng phần mềm là 1 mô hình có kế hoạch và hệ thống của tất cả các hành động cần thiết để đem lại sự tin cậy rằng 1 sản phẩm phần mềm tuân thủ đầy đủ các yêu cầu về kĩ thuật đã đề ra.

[McCall](https://en.wikipedia.org/wiki/ISO/IEC_9126) đã đề ra 11 tiêu chí cho Đảm bảo chất lượng phần mềm, được chia thành 3 loại:

#### Tiêu chí vận hành sản phẩm (Product operation factors): Hệ thống có chạy tốt không, có dễ sử dụng không

##### Correctness: Tính đúng đắn: đặc tả về độ chính xác, sự toàn vẹn của output

##### Reliability: Tính tin cậy: Đề cập tới lỗi khi cung cấp dịch vụ: tỉ lệ lỗi, thời gian hệ thống chết. Khả năng chịu lỗi, khả năng tự phục hồi.

##### Efficiency: Tính hiệu quả: Đề cập tới tài nguyên phần cứng cần để thực hiện các chức năng của phần mềm.

##### Integrity: Tính toàn vẹn: Đề cập tới bảo mật của hệ thống với việc ngăn chặn truy cập trái phép

##### Usability: Tính khả dụng: Đề cập tới quy mô nguồn lực để đào tạo nhân viên mới sử dụng hệ thống

#### Tiêu chí sửa đổi sản phầm (Product revision factors): Hệ thống có dễ dàng sửa lỗi không, dễ dàng kiểm thử không

##### Maintainability: Mức công sức cần để bảo trì khi có lỗi, kiến trúc các module như thế nào

##### Flexibility: Đề cập tới nguồn lực để thay đổi phần mềm khi khách hàng thay đổi

##### Testability: Có hỗ trợ test hay không: tạo file log, backup

#### Tiêu chí chuyển giao sản phầm (Product transition factors): Hệ thống có dễ dàng chuyển đổi sang các phần cứng không, có thể tái sử dụng không

##### Portability: Nếu phần mềm cài ở môi trường mới, có giữ đc các tính năng như cũ không. Thích nghi với nhiều môi trường (lớp nền, services,…), cùng nhau tồn tại trong môi trường (no conflict), thay thế cho phần mềm khác có cùng chức năng.

##### Reusability: Có thể tái sử dụng các module nhỏ không, PM có thể tháo rời thành nhiều gói dùng lại được

##### Interoperability: Phần mềm có cần Interface với các hệ thống đã có, PM có giao tiếp & phương thức hợp chuẩn.

## Các yếu tố chất lượng phần mềm theo ISO 9126, 2001:

#### Tính chức năng (Functionality): Khả năng của phần mềm cung cấp các chức năng đáp ứng được nhu cầu sử dụng khi phần mềm làm việc trong điều kiện cụ thể, bao gồm:

##### Tính phù hợp (Suitability): là khả năng của một phần mềm có thể cung cấp một tập các chức năng thích hợp cho công việc cụ thể phục vụ mục đích của người sử dụng.

##### Tính chính xác (Accuracy): là khả năng của phần mềm có thể cung cấp các kết quả hay hiệu quả đúng đắn hoặc chấp nhận được với độ chính xác cần thiết.

##### Khả năng tương tác (Interoperability): khả năng tương tác với một hoặc mộtvài hệ thống cụ thể của phần mềm.

##### Tính bảo mật/an toàn (Security): khả năng bảo vệ thông tin và dữ liệu của sản phẩm phần mềm, sao cho người, hệ thống không được phép thì không thể truy cập, đọc hay chỉnh sửa chúng.

#### Tính tin cậy (Reability): Là khả năng của phần mềm có thể hoạt động ổn định trong những điều kiện cụ thể:

##### Tính hoàn thiện (Maturity): khả năng tránh các kết quả sai

##### Khả năng chịu lỗi (Fault tolerant): khả năng của phần mềm hoạt động ổn định tại một mức độ cả trong trường hợp có lỗi xảy ra ở phần mềm hoặc có những vi phạm trong giao diện.

##### Khả năng phục hồi (Recoverability): khả năng của phần mềm có thể tái thiết lại hoạt động tại một mức xác định và khôi phục lại những dữ liệu có liên quan trực tiếp đến lỗi.

#### Tính khả dụng (Usability): Là khả năng của phần mềm có thể hiểu được, học được, sử dụng được và hấp dẫn người sử dụng trong từng trường hợp sử dụng cụ thể:

##### Dễ hiểu, có thể hiểu được (Understandability): người dùng có thể hiểu được xem phần mềm có hợp với họ không và và sử dụng chúng thế nào cho những công việc cụ thể.

##### Dễ học (Learnability): người sử dụng có thể học các ứng dụng của phần mềm.

##### Có thể sử dụng được (Operability); khả năng của phần mềm cho phép người dùng sử dụng và điều khiển nó.

##### Tính hấp dẫn (Attractiveness): khả năng hấp dẫn người sử dụng của phần mềm

#### Tính hiệu quả (Efficiency): Là khả năng của phần mềm có thể hoạt động một cách hợp lý, tương ứng với lượng tài nguyên nó sử dụng, trong điều kiện cụ thể:

##### Đáp ứng thời gian (Time behavior): khả năng của phần mềm có thể đưa ra một trả lời, một thời gian xử lý và một tốc độ thông lượng hợp lý khi nó thực hiệncôngviệc của mình, dưới một điều kiện làm việc xác định.

##### Sử dụng tài nguyên (Utilization): khả năng của phần mềm có thể sử dụng một lượng, một loại tài nguyên hợp lý để thực hiện công việc trong những điều kiện cụthể

#### Khả năng bảo trì (Maintainability): Là khả năng của phần mềm có thể chỉnh sửa. Việc chỉnh sửa bao gồm: sửa lại cho đúng, cải tiến và làm phần mềm thích nghi được với những thay đổi của môi trường, của yêu cầu và của chức năng xác định

##### Khả năng phân tích (Analysability): phần mềm có thể được chẩn đoán để tìm những thiếu sót hay những nguyên nhân gây lỗi hoặc để xác định những phần cần sửa.

##### Khả năng thay đổi được (Changeability): phần mềm có thể chấp nhận một số thay đổi cụ thể trong quá trình triển khai.

##### Tính ổn định (Stability): khả năng tránh những tác động không mong muốn khi chỉnh sửa phần mềm.

##### Khả năng kiểm thử được (Testability): khả năng cho phép đánh giá được phần mềm chỉnh sửa.

#### Tính khả chuyển (Portability): Là khả năng của phần mềm cho phép nó có thể được chuyển từ môi trường này sang môi trường khác (hardwares, OS,…):

##### Khả năng thích nghi (Adaptability): khả năng của phần mềm có thể thích nghi với nhiều môi trường khác nhau mà không cần phải thay đổi.

##### Khả năng cài đặt (Installability): phần mềm có thể cài đặt được trên những môi trường cụ thể.

##### Khả năng chung sống (Co-existence): phần mềm có thể cùng tồn tại với những phần mềm độc lập khác trong một môi trường chung, cùng chia sẻ những tài nguyên chung.

##### Khả năng thay thế được (Replaceability): phần mềm có thể dùng thay thếcho một phần mềm khác, với cùng mục đích và trong cùng môi trường.

## Kiến trúc của hệ thống đảm bảo chất lượng phần mềm

Một hệ thống SQA luôn kết hợp nhiều thành phần SQA, tất cả đều được sử dụng để loại bỏ các lỗi phần mềm và để đạt được mức chất lượng phần mềm có thể chấp nhận được. Nhiệm vụ của SQA là duy nhất trong lĩnh vực đảm bảo chất lượng do các đặc tính đặc biệt của phần mềm. Ngoài ra, môi trường mà việc phát triển và bảo trì phần mềm được thực hiện ảnh hưởng trực tiếp đến các thành phần SQA.

Các thành phần hệ thống SQA có thể được phân loại thành sáu lớp:

#### Các thành phần chất lượng trước dự án - Pre-project SQA components;

#### Các thành phần chất lượng vòng đời dự án - Project life cycle SQA components;

#### Các thành phần cải tiến và ngăn ngừa lỗi cơ sở hạ tầng - Quality infrastructure components;

#### Các thành phần quản lý chất lượng phần mềm - Quality management;

#### Tiêu chuẩn hóa, chứng nhận và các thành phần đánh giá SQA - Standards;

#### Tổ chức cho SQA - các thành phần con người - Organizational base– human components.

#### **Các thành phần tiền dự án.** Để đảm bảo rằng (a) các cam kết của dự án đã được xác định đầy đủ khi xem xét các nguồn lực cần thiết, tiến độ và ngân sách; và (b) kế hoạch phát triển và chất lượng đã được xác định một cách chính xác.

#### **Các thành phần của đánh giá các hoạt động vòng đời của dự án.** Vòng đời của dự án bao gồm hai giai đoạn: giai đoạn vòng đời phát triển và giai đoạn vận hành - bảo trì. Các thành phần giai đoạn vòng đời phát triển phát hiện lỗi thiết kế và lập trình. Các thành phần của nó được chia thành bốn lớp con sau:

##### Nhận xét;

##### Ý kiến ​​chuyên gia;

##### Kiểm thử phần mềm.

Các thành phần SQA được sử dụng trong giai đoạn vận hành - bảo trì bao gồm các thành phần bảo trì chuyên biệt cũng như các thành phần vòng đời phát triển, được áp dụng chủ yếu cho các nhiệm vụ bảo trì cải thiện chức năng.

Một nhóm phụ bổ sung của các thành phần vòng đời dự án SQA đề cập đến việc đảm bảo chất lượng của các phần dự án được thực hiện bởi các nhà thầu phụ và những người tham gia bên ngoài khác trong quá trình phát triển và bảo trì dự án.

#### **Các thành phần của việc ngăn ngừa và cải thiện lỗi cơ sở hạ tầng.** Các mục tiêu chính của các thành phần này, được áp dụng trong toàn bộ tổ chức, là loại bỏ hoặc ít nhất là giảm tỷ lệ sai sót, dựa trên kinh nghiệm SQA tích lũy của tổ chức.

#### **Các thành phần của quản lý chất lượng phần mềm.** Nhóm thành phần này hướng tới một số mục tiêu, các mục tiêu chính là kiểm soát các hoạt động phát triển và bảo trì và đưa ra các hành động hỗ trợ quản lý sớm chủ yếu ngăn ngừa hoặc giảm thiểu các thất bại về lịch trình và ngân sách cũng như kết quả của chúng.

#### **Các thành phần của tiêu chuẩn hóa, chứng nhận và đánh giá hệ thống SQA**. Các thành phần này thực hiện các tiêu chuẩn quốc tế về chuyên môn và quản lý trong tổ chức. Các mục tiêu chính của lớp này là (a) sử dụng kiến ​​thức chuyên môn quốc tế, (b) cải thiện sự phối hợp của hệ thống chất lượng của tổ chức với các tổ chức khác, và (c) đánh giá những thành tựu của hệ thống chất lượng theo một thang điểm chung. Các tiêu chuẩn khác nhau có thể được phân thành hai nhóm chính: (a) tiêu chuẩn quản lý chất lượng và (b) tiêu chuẩn quy trình dự án.

#### **Tổ chức cho SQA - các thành phần của con người**. Cơ sở tổ chức của SQA bao gồm các nhà quản lý, nhân viên kiểm thử, đơn vị SQA và các học viên quan tâm đến chất lượng phần mềm (người được ủy thác SQA, thành viên ủy ban SQA và thành viên diễn đàn SQA). Tất cả những tác nhân này đều đóng góp vào chất lượng phần mềm; các mục tiêu chính của họ là khởi xướng và hỗ trợ việc thực hiện các thành phần của SQA, phát hiện những sai lệch so với các thủ tục và phương pháp luận của SQA, đồng thời đề xuất các cải tiến.

# Các kĩ thuật kiểm thử và các ca kiểm thử



## Các chiến lược kiểm thử

### Khái niệm chiến lược kiểm thử

Chiến lược kiểm thử là sự tích hợp các kỹ thuật thiết kế ca kiểm thử tạo thành một dãy các bước nhằm hướng dẫn quá trình kiểm thử phần mềm thành công.

Chiến lược kiểm thử được đặt ra với mục tiêu nhằm phác thảo một lộ trình để:

#### Nhà phát triển tổ chức việc bảo đảm chất lượng bằng kiểm thử,

#### Khách hàng hiểu được công sức, thời gian và nguồn lực cần cho kiểm thử.

Chiến lược kiểm thử cần phải đạt những yếu cầu sau:

#### Tích hợp được các khâu như lập kế hoạch, thiết kế ca kiểm thử, tiến hành kiểm thử, thu thập và đánh giá các thong tin kết quả.

#### Đủ mềm dẻo để cổ vũ óc sáng tạo, đáp ứng được nhu cầu khách hàng

#### Thích ứng với mức kiểm thử cụ thể

#### Đáp ứng các đối tượng quan tâm khác nhau

Kiểm thử là một tập hợp những hoạt động mà có thể được lập kế hoạch trước và tiến hành một cách có hệ thống. Một tập các bước mà trong đó chúng ta có thể vận dụng những kỹ thuật thiết kế ca kiểm thử và phương pháp kiểm thử có những đặc trưng mang tính “khuôn mẫu”:

#### Bắt đầu ở mức mô-đun và tiếp tục cho đến khi tích hợp ở mức hệ thống trọn vẹn.

#### Các kỹ thuật kiểm thử khác nhau là thích hợp cho những thời điểm khác nhau.

#### Được cả người phát triển và nhóm kiểm thử độc lập cùng tiến hành.

#### Kiểm thử đi trước gỡ lỗi, song việc gỡ lỗi phải thích ứng với từng chiến lược kiểm thử.

Chiến lược cần thích ứng với từng mức kiểm thử và phải đưa ra hướng dẫn cho người thực hành và một tập các cột mốc cho người quản lý. Có hai mức kiểm thử:

#### Kiểm thử mức thấp: thẩm định từng đoạn mã nguồn xem có tương ứng và thực thi đúng đắn hay không?

#### Kiểm thử mức cao: thẩm định và xác minh các chức năng hệ thống chủ yếu có đúng đặc tả và đáp ứng yêu cầu của khách hàng hay không?

Mỗi chiến lược đáp ứng được yêu cầu cần quan tâm:

#### Có các hướng dẫn cho người thực hiện tiến hành kiểm thử.

#### Có các cột mốc cho các nhà quản lý kiểm soát hoạt động bảo đảm chất lượng

#### Có thước đo để đo và nhận ra các vấn đề càng sớm càng tốt.

#### Khách hàng có thể nhận biết được quá trình kiểm thử.

Việc kiểm thử cung cấp một thành lũy cuối cùng để có thể thẩm định về chất lượng và có thể phát hiện ra lỗi.

Có một số quan điểm sai lầm:

#### Người phát triển không nên tham gia kiểm thử.

#### Cho phép người lạ kiểm thử một cách thô bạo.

#### Người kiểm thử chỉ quan tâm khi kiểm thử bắt đầu.

Nên xuất phát từ thực tiễn mà phân công trách nhiệm thử:

#### Người phát triển chịu trách nhiệm kiểm thử đơn vị do mình phát triển để bảo đảm thực hiện theo đúng thiết kế, có thể tham gia kiểm thử tích hợp; không khoán trắng chương trình cho người kiểm thử mà phải cùng làm việc với người kiểm thử xuyên suốt dự án.

#### Nhóm kiểm thử độc lập bắt đầu làm việc khi các khoản mục cấu trúc phần mềm đã đầy đủ, giúp gỡ bỏ những thành kiến: “người xây dựng không thể kiểm thử tốt sản phẩm”, gỡ bỏ mâu thuẫn giữa những người tham gia; đánh giá công sức người phát triển bỏ ra tìm lỗi; tạo ra báo cáo đầy đủ cho tổ chức bảo đảm chất lượng phần mềm.

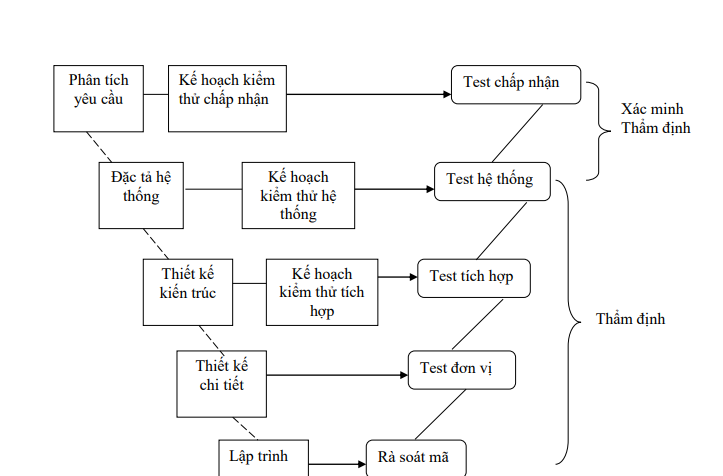
### Mô hình chiến lược tổng thể

Về mặt kỹ nghệ, việc kiểm thử gồm một số bước được thực hiện tuần tự. Ban đầu, việc kiểm thử tập trung vào từng mô-đun riêng biệt bảo đảm nó ban hành đúng đắn như một đơn vị. Do đó mới có tên kiểm thử đơn vị. Kiểm thử đơn vị dùng rất nhiều các kỹ thuật kiểm thử hộp trắng, kiểm soát các đường đặc biệt trong cấu trúc điều khiển của một lớp mô-đun nhằm phát hiện tối đa các lỗi. Mặt khác, các mô-đun phải được lắp ghép hay tích hợp lại để tạo nên phần mềm hoàn chỉnh.

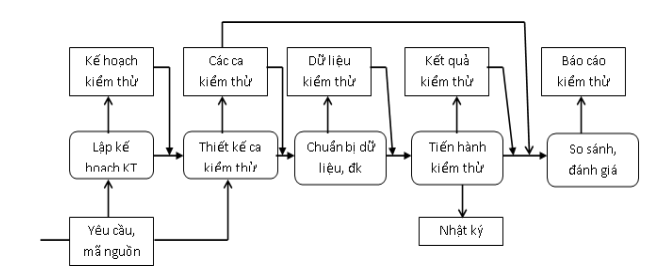
Việc kiểm thử tích hợp có liên quan đến thẩm định và xây dựng chương trình. Các kỹ thuật thiết kế kiểm thử hộp đen được dùng trong hầu hết quá trình tích hợp, mặc dù các kiểm thử hộp trắng cũng có thể được dùng để bao quát đa số các đường điều khiển. Sau khi phần mềm đã được dùng tích hợp (được xây dựng), một tập hợp các phép kiểm thử sẽ được tiến hành. Các tiêu chuẩn hợp lệ (được thiết lập trong phân tích yêu cầu) cũng phải được kiểm thử.Việc kiểm thử hợp lệ được tiến hành nhằm bảo đảm phần mềm đáp ứng đầy đủ các yêu cầu chức năng. Các kỹ thuật kiểm thử hộp đen được dùng chủ yếu trong kiểm thử hợp lệ.

Kiểm thử hệ thống nằm trong khung cảnh rộng hơn của kỹ nghệ hệ thống máy tính. Khi làm hợp lệ, phần mềm phải được tổ hợp với các phần tử 20 hệ thống khác (như phần cứng, con người, CSDL).Vì vậy, kiểm thử hệ thống là rất quan trọng.

Cuối cùng, kiểm thử chấp nhận sẽ thẩm định lại rằng tất cả các thành phần có phối khớp với nhau không, cũng như chức năng hay độ hoàn thiện của hệ thống có đạt được không.



Mô hình chiến lược kiểm thử tổng thể



Mô hình tiến trình thực hiện kiểm thử

### Chiến lược kiểm thử hệ thời gian thực

Hệ thời gian thực là hệ thống đáp ứng đúng, chính xác các sự kiện của môi trường.

Kiểm thử hệ thống thời gian thực là rất khó. Những người thiết kế ca kiểm thử không chỉ phải xem xét các trường hợp kiểm thử hộp đen và hộp trắng mà còn phải xem xét cả việc chia thời gian cho dữ liệu cà cơ chế song song cho các nhiệm vụ (tiến trình) giải quyết dữ liệu đó.Trong nhiều tình huống, trạng thái của hệ thống cũng có thể dẫn tới lỗi.

Mối quan hệ mật thiết giữa phần mềm thời gian thực và môi trường phần cứng của nó cũng có thể gây ra các vấn đề cho kiểm thử.Việc kiểm thử 22 phần mềm phải xem xét tác động của các lỗi phần cứng đến xử lý phần mềm.những lỗi như thế rất khó mô phỏng sát thực tế.

### Kiểm thử Alpha và Beta

Kiểm thử alpha (alpha testing) và kiểm thử beta (beta testing) đều do người dùng thực hiện và đều được thực hiện trong môi trường thực.

Người phát triển không thể nào lường hết được khách hàng sẽ dùng chương trình như thế nào. Các hướng dẫn sử dụng có thể bị hiểu sai, việc tổ hợp dữ liệu lạ lại có thể hay được dùng, cái ra dường như rõ ràng với người kiểm thử, nhưng có thể lại khó hiểu đối với người dùng.

#### Kiểm thử Alpha

Kiểm thử Alpha được khách hàng tiến hành tại cơ quan của người phát triển. Phần mềm được dùng theo sự sắp đặt tự nhiên với người phát triển (nhìn qua vai) người dùng và ghi lại các lỗi khi sử dụng. Kiểm thử Alpha còn có tên khác là kiểm thử “sau lưng” và được tiến hành trong một môi trường có kiểm soát. Dữ liệu cho kiểm thử Alpha là dữ liệu mô phỏng

#### Kiểm thử Beta

Kiểm thử Beta được người dùng cuối tiến hành tại một hay nhiều cơ quan khách hàng. Không giống kiểm thử Alpha, người phát triển thường không có mặt. Do đókiểm thử Beta là việc áp dụng “sống” của phần mềm trong một môi trường mà người phát triển không thể nào kiểm soát được. Khách hàng ghi lại tất cả các vấn đề (thực hay tưởng tượng) gặp phải trong khi kiểm thử Beta và báo cáo lại những vấn đề đó cho người phát triển trong những khoảng thời gian đều đặn. Xem như một kết quả của vấn đề được báo cáo trong kiểm thử Beta, người phát triển tiến hành sửa đổi rồi chuẩn bị đưa ra sản phẩm phần mềm cho toàn bộ khách hàng.

### Kiểm thử so sánh

Có một số tình huống (như điều khiển máy bay, điều khiển nhà máy năng lượng hạt nhân) mà trong đó độ tin cậy của phần mềm là yếu tố hàng đầu.Trong những ứng dụng như vậy, phần cứng và phần mềm thường được yêu cầu tối thiểu hóa khả năng lỗi. Khi phần mềm được phát triển, nhóm kỹ nghệ phần mềm tách biệt sẽ phát triển những bản độc lập ứng dụng bằng cách dùng cùng đặc tả.Trong những tình huống như thế, mỗi bản có thể được kiểm thử với cùng dữ liệu để đảm bảo rằng tất cả đều cho kết quả giống nhau.

Các nhà nghiên cứu đã gợi ý rằng các bản phần mềm độc lập được phát triển. Những bản độc lập này tạo nền cho kỹ thuật kiểm thử hộp đen được gọi là kiểm thử so sánh (comparision testing) hay kiểm thử dựa vào nhau (backto-back testing).

Khi tạo ra nhiều cài đặt cho cùng một đặc tả, các ca kiểm thử được thiết kế để dùng những kỹ thuật hộp đen khác (như phân hoạch tương đương) được xem như từng bản đầu vào của phần mềm. Nếu kết quả ra của mỗi bản là giống nhau thì người ta giả thiết rằng tất cả các cài đặt đều đúng. Nếu kết quả ra là khác nhau thì từng ứng dụng sẽ được nghiên cứu lại để xác định liệu một khiếm khuyết trong một hay nhiều bản có phải là nguyên nhân sinh ra sự khác biệt hay không.Trong phần lớn các trường hợp, việc so sánh các kết quả ra có thể được thực hiện tự động.

Việc kiểm thử so sánh không đơn giản. Nếu đặc tả mà từ đó tất cả các phiên bản đã được xây dựng ra bị lỗi thì mọi bản đều có thể phản ánh lỗi đó.Mặt khác, nếu từng bản độc lập lại tạo ra kết quả giống nhau, nhưng không đúng, thì việc kiểm thử điều kiện sẽ không phát hiện được lỗi.

Trong quá trình kiểm thử so sánh, cần chú ý:

#### Khi triển khai nhiều phiên bản phần mềm từ cùng một đặc tả, kiểm thử hộp đen cho các sản phẩm này được thực hiện với cùng ca kiểm thử và cùng các dữ liệu vào.

#### Khi so sánh các kết quả thu được, nếu có khác biệt nghĩa là có sai trong một sản phẩm nào đó.

## Các kĩ thuật kiểm thử



### Kiểm thử hộp đen (Black-box Testing)

Hay còn gọi là kiểm thử chức năng kiểm tra sự hoạt động đúng của các chức năng giao tiếp với bên ngoài của phần mềm thông qua sự quan sát chúng trong quá trình thực thi. Do phần mềm được coi như một hộp đen, với các luồng dữ liệu vào ra và một số đặc tính có thể quan sát được khác nên loại kiểm thử này còn được gọi là kiểm thử hộp đen.

Dạng đơn giản nhất của kiểm thử hộp đen là bắt đầu chạy chương trình và quan sát để tìm ra những hành vi, những hoạt động mong muốn và không mong muốn. Dạng kiểm thử này được gọi là kiểm thử vô thể thức (ad hoc testing).

Một dạng phổ biến khác của kiểm thử hộp đen là sử dụng một danh sách kiểm tra, danh sách kiểm tra là một danh sách các chức năng ngoài với các thông tin trạng thái mong muốn và thông tin vào ra. Thông tin vào ở đây bao gồm tất cả những hành động, công cụ và tài nguyên được cung cấp cho chương trình tại thời điểm trước lúc chạy chương trình hoặc tại bất kì thời điểm nào trong lúc chương trình chạy. Tương tự thông tin ra là tất các những hành vi và kết quả của chương trình tại thời điểm chương trình kết thúc hay tại bất kì thời điểm nào trong lúc chương trình đang chạy. Ví dụ thông tin vào cho một chương trình tính toán là các con số từ bàn phím, hành động là chia hai số đó cho nhau, thông tin ra có thể là kết quả đúng hay có thể là một thông báo lỗi – trong trường hợp chia cho không.

Kiểm thử hộp đen có thể tuân theo quy trình kiểm thử ở trên, bao gồm 3 hoạt động chính là lên kế hoạch, thực thi, phân tích và theo dõi:

##### Lên kế hoạch tập trung vào việc xác định các chức năng ngoài, thông tin đầu vào, thông tin đầu ra và những trạng thái mong muốn của người sử dụng.

##### Thực thi kiểm thử tập trung vào việc quan sát cách hoạt động của chương trình, đảm bảo các trường hợp kiểm thử thực hiện đúng thứ tự, ghi nhận kết quả để phân tích. Nếu sự quan sát không tìm thấy các lỗi trực tiếp, nhưng thông tin ghi nhận được vẫn cần được lưu lại cho việc phân tích về sau. Như trong ví dụ trên, thông tin đầu ra, những thông tin về quá trình dịch cũng như các tham số cấu hình cần được lưu lại.

##### Thông tin thu được trong quá trình thực thi kiểm thử được so sánh với những thông tin chuẩn để xác định một chức năng nào đó thỏa mãn hay không thỏa mãn yêu cầu. Một chức năng nào đó không thỏa mãn yêu cầu sẽ được theo dõi cô lập, phát hiện và sửa chữa.

Kiểm thử hộp đen hay còn gọi kiểm thử hướng dữ liệu (data driven) hay là kiểm thử hướng vào/ra (input/output driven).

Trong kỹ thuật này, người kiểm thử xem phần mềm như là một hộp đen. Người kiểm thử hoàn toàn không quan tâm đến cấu trúc và hành vi bên trong của chương trình. Người kiểm thử chỉ cần quan tâm đến việc tìm các hiện tượng mà phần mềm không hành xử theo đúng đặc tả của nó. Do đó, dữ liệu kiểm thử sẽ xuất phát từ đặc tả.

### Kiểm thử hộp trắng

Hay còn gọi là kiểm thử cấu trúc kiểm tra sự cài đặt đúng của những đơn vị bên trong chương trình phần mềm như các câu lệnh, các cấu trúc dữ liệu, các khối ... và quan hệ giữa chúng. Việc kiểm tra được thực hiện thông qua việc quan sát kết quả của sự thực thi các đơn vị đó và mối quan hệ với các đơn vị định trước. Phần mềm được xem như là một chiếc hộp trắng (thực chất là hộp trong suốt), các đơn vị bên trong chương trình được nhìn thấy cùng với các mối tương tác giữa chúng.

Kiểm thử cấu trúc được hỗ trợ bởi một số công cụ phần mềm, dạng đơn giản nhất là kiểm thử mọi dòng lệnh thông qua một số công cụ gỡ lỗi, 30 (debugging tool hay debugger) giúp dò vết khi thực hiện chương trình. Do đó người kiểm thử có thể biết được khi một lệnh được thực thi, kết quả của nó có như mong muốn hay không. Ưu điểm của cách kiểm thử này là khi phát hiện được lỗi đồng thời cũng xác định được lỗi ngay, tuy nhiên nó yêu cầu người kiểm thử phải thông thạo mã nguồn và các lỗi về sự thiếu sót, sự sai trong thiết kế rất khó được phát hiện. Kiểm thử cấu trúc nên được thực hiện bởi chính những người viết chương trình đó thì việc phát hiện và sửa lỗi mới dễ dàng.

Kiểm thử cấu trúc cũng có thể tuân theo quy trình kiểm thử phần mềm chung, tuy nhiên do yêu cầu về sự thông thạo mã nguồn chương trình, bao quát toàn bộ sự cài đặt của hệ thống – điều này là rất khó, nên kiểm thử cấu trúc thường được giới hạn với quy mô nhỏ. Đối với các sản phẩm phần mềm nhỏ là không cần thiết phải có một quy trình đầy đủ cho việc kiểm thử, phát hiện và sửa lỗi. Đối với các chương trình lớn, việc kiểm thử cấu trúc được thực hiện trong một framework hoàn chỉnh do vậy việc lập kế hoạch kiểm thử giữ vai trò kém quan trọng hơn so với kiểm thử chức năng. Thêm vào đó việc phát hiện và sửa lỗi cũng dễ dàng do có sự liên kết chặt chẽ giữa chức năng của chương trình với các đơn vị chương trình, thêm vào đó là vai trò kép vừa là người kiểm thử vừa là người viết chương trình. Do vậy kiểm thử cấu trúc không đòi hỏi một quy trình chặt chẽ như kiểm thử chức năng.

Kiểm thử hộp trắng hay còn gọi là kiểm thử hướng logic, cho phép kiểm tra cấu trúc bên trong của phần mềm với mục đích bảo đảm rằng tất cả các câu lệnh và điều kiện sẽ được thực hiện ít nhất một lần. Người kiểm thử truy nhập vào mã nguồn chương trình và có thể kiểm tra nó, lấy đó làm cơ sở để hỗ trợ việc kiểm thử.

## Các ca kiểm thử



### Khái niệm các ca kiểm thử

Ca kiểm thử (test case) là một tình huống kiểm thử tương ứng với một mạch hoạt động của chương trình.

Nó bao gồm một tập các giá trị đầu vào và một danh sách các kết quả đầu ra mong muốn và thực tế.

Mục tiêu thiết kế ca kiểm thử nhằm:

#### Tìm ra nhiều lỗi nhất

#### Với chi phí và thời gian ít nhất

Trong các thập kỷ 80-90 của thế kỷ XX, người ta đã nghiên cứu nhiều loại phương pháp thiết kế ca kiểm thử. Trong các phương pháp này, phương pháp thiết kế ca kiểm thử. Trong các phương pháp này, phương pháp thiết kế được chọn theo cơ chế:

#### Bảo đảm tính đầy đủ (không sót phần nào)

#### Cung cấp khả năng phát hiện lỗi nhiều nhất

Việc thiết kế 1 ca kiểm thử được đặt ra với lý do sau là chủ yếu:

#### Số con đường logic/ mạch thực hiện trong chương trình là rất lớn

#### Nhiều trạng thái dữ liệu khác nhau: số đại lượng, giá trị, sự thay đổi trong tiến trình, sự kết hợp giữa chúng

Về nguyên tắc:

#### Không bao giờ kiểm thử được tất cả

#### Vận hành chương trình là đang kiểm thử

#### Kiểm thử tiếp tục khi chương trình còn hoạt động Kỹ sư phần mềm cần các tiêu chuẩn nghiêm ngặt để xác định có cần phải tiếp tục kiểm thử không.

### Vấn đề khi thiết kế ca kiểm thử

Thiết kế test – case trong kiểm thử phần mềm là quá trình xây dựng các phương pháp kiểm thử có thể phát hiện lỗi, sai sót, khuyết điểm của phần mềm để xây dựng phần mềm đạt tiêu chuẩn.

Thiết kế test-case giữ vai trò quan trọng trong việc nâng cao chất lượng phần mềm vì những lý do sau đây:

• Tạo ra các ca kiểm thử tốt nhất có khả năng phát hiện ra lỗi, sai sót của phần mềm một cách nhiều nhất.

• Tạo ra các ca kiểm thử có chi phí rẻ nhất, đồng thời tốn ít thời gian và công sức nhất.

Một trong những lý do quan trọng nhất trong kiểm thử chương trình là thiết kế và tạo ra các ca kiểm thử - các Test case có hiệu quả. Với những ràng buộc về thời gian và chi phí đã cho, thì vấn đề then chốt của kiểm thử trở thành: Tập con nào của tất cả ca kiểm thử có thể có khả năng tìm ra nhiều lỗi nhất?

Thông thường, phương pháp kém hiệu quả nhất là kiểm tra tất cả đầu vào ngẫu nhiên – quá trình kiểm thử một chương trình bằng việc chọn ngẫu nhiên một tập con các giá trị đầu vào có thể. Về mặt khả năng tìm ra nhiều lỗi nhất, tập hợp các ca kiểm thử được chọn ngẫu nhiên có rất ít cơ hội là tập hợp tối ưu hay gần tối ưu. Sau đây là một số phương pháp để chọn ra một tập dữ liệu kiểm thử một cách thông minh.

Để kiểm thử hộp đen và kiểm thử hộp trắng một cách thấu đáo là không thể. Do đó, một chiến lược kiểm thử hợp lý là chiến lược có thể kết hợp sức mạnh của cả hai phương pháp trên: Phát triển 1 cuộc kiểm thử nghiêm ngặt vừa bằng việc sử dụng các phương pháp thiết kế ca kiểm thử hướng hộp đen nào đó và sau đó bổ sung thêm những ca kiểm thử này bằng việc khảo sát tính logic của chương trình, sử dụng phương pháp hộp trắng.

Các chiến lược kiểm thử kết hợp bao gồm

|  |  |
| --- | --- |
| Hộp đen | Hộp trắng |
| Phân lớp tương đương | Bao phủ câu lệnh |
| Phân tích giá trị biên | Bao phủ quyết định |
| Đồ thị nguyên nhân kết quả | Bao phủ điều kiện |
| Đoán lỗi | Bao phủ điều kiện-quyết định |
|  | Bao phủ đa điều kiện |

Mỗi phương pháp có những ưu điểm cũng như khuyết điểm riêng, do đó để có được tập các ca kiểm thử tối ưu, chúng ta cần kết hợp hầu hết các phương pháp. Quy trình thiết kế các ca kiểm thử sẽ bắt đầu bằng việc phát triển các ca kiểm thử sử dụng phương pháp hộp đen và sau đó phát triển bổ sung các ca kiểm thử cần thiết với phương pháp hộp trắng.

# Các kĩ thuật thiết kế ca kiểm thử



## Kĩ thuật bao phủ câu lệnh (Statement Coverge)

Tư tưởng của phương pháp Thiết kế các ca kiểm thử dựa vào ý tưởng phương pháp bao phủ câu lệnh (Statement Coverage) là thực hiện mọi câu lệnh trong chương trình ít nhất 1 lần. Phương pháp này bao gồm:

### Kỹ thuật bao phủ quyết định

Ý tưởng của phương pháp bao phủ quyết định (Decision coverage) là viết đủ các ca kiểm thử mà mỗi quyết định có kết luận đúng hay sai ít nhất 1 lần.

Nói cách khác, mỗi hướng phân nhánh phải được xem xét kỹ lưỡng ít nhất 1 lần. Bao phủ quyết định thường thỏa mãn bao phủ câu lệnh. Vì mỗi câu lệnh là trên sự bắt nguồn một đường đi phụ nào đó hoặc là từ một câu lệnh rẽ nhánh hoặc là từ điểm vào của chương trình, mỗi câu lệnh phải được thực hiện nếu mỗi quyết định rẽ nhánh được thực hiện.

Tuy nhiên, có ít nhất 3 ngoại lệ:

1. Những chương trình không có quyết định.

2. Những chương trình hay thường trình con/phương thức với nhiều điểm vào. Một câu lệnh đã cho có thể được thực hiện nếu và chỉ nếu chương trình được nhập vào tại 1 điểm đầu vào riêng.

3. Các câu lệnh bên trong các ON-unit. Việc đi qua mỗi hướng rẽ nhánh sẽ là không nhất thiết làm cho tất cả các ON-unit được thực thi.

Vì chúng ta đã thấy rằng bao phủ câu lệnh là điều kiện cần thiết, nên một chiến lược tốt hơn là bao phủ quyết định nên được định nghĩa bao hàm cả bao phủ câu lệnh. Do đó, bao phủ quyết định yêu cầu mỗi quyết định phải có kết luận đúng hoặc sai, và mỗi câu lệnh đó phải được thực hiện ít nhất 1 lần. Bao phủ quyết định là một tiêu chuẩn mạnh hơn bao phủ câu lệnh, nhưng vẫn khá yếu.

### Kỹ thuật bao phủ điều kiện (Condition Coverage)

Ý tưởng của phương pháp bao phủ điều kiện (Condition coverage) là viết đủ các ca kiểm thử để đảm bảo rằng mỗi điều kiện trong một quyết định đảm nhận tất cả các kết quả có thể ít nhất 1 lần.

### Kỹ thuật bao phủ quyết định/ điều kiện (Decision/Condition coverage)

Ý tưởng của phương pháp bao phủ quyết định/ điều kiện (Decision/ condition coverage) là thực hiện đủ các ca kiểm thử mà mỗi điều kiện trong một quyết định thực hiện trên tất cả các kết quả có thể ít nhất 1 lần, và mỗi điểm vào được gọi ít nhất 1 lần. Điểm yếu của bao phủ quyết định/điều kiện là mặc dù xem ra nó có thể sử dụng tất cả các kết quả của tất cả các điều kiện, nhưng thường không phải vậy vì những điều kiện chắc chắn đã cản các điều kiện khác.

### Kỹ thuật bao phủ đa điều kiện (Multiple Condition Coverage)

Ý tưởng của phương pháp bao phủ đa điều kiện (Multiple condition coverage) là viết đủ các ca kiểm thử mà tất cả những sự kết hợp của các kết quả điều kiện có thể trong mỗi quyết định, và tất cả các điểm vào phải được gọi ít nhất 1 lần.

### Kỹ thuật bao phủ quyết định/ điều kiện (Decision/Condition coverage)

Ý tưởng của phương pháp bao phủ quyết định/ điều kiện (Decision/ condition coverage) là thực hiện đủ các ca kiểm thử mà mỗi điều kiện trong một quyết định thực hiện trên tất cả các kết quả có thể ít nhất 1 lần, và mỗi điểm vào được gọi ít nhất 1 lần.

Điểm yếu của bao phủ quyết định/điều kiện là mặc dù xem ra nó có thể sử dụng tất cả các kết quả của tất cả các điều kiện, nhưng thường không phải vậy vì những điều kiện chắc chắn đã cản các điều kiện khác.

### Kỹ thuật bao phủ đa điều kiện (Multiple Condition Coverage)

Ý tưởng của phương pháp bao phủ đa điều kiện (Multiple condition coverage) là viết đủ các ca kiểm thử mà tất cả những sự kết hợp của các kết quả điều kiện có thể trong mỗi quyết định, và tất cả các điểm vào phải được gọi ít nhất 1 lần.

## Kĩ thuật phân lớp tương đương

Phân lớp tương đương là một phương pháp kiểm thử hộp đen chia miền đầu vào của một chương trình thành các lớp dữ liệu, từ đó suy dẫn ra các ca kiểm thử. Phương pháp này cố gắng xác định ra một ca kiểm thử mà làm lộ ra một lớp lỗi, do đó làm giảm tổng số các trường hợp kiểm thử phải được xây dựng.

Thiết kế ca kiểm thử cho phân lớp tương đương dựa trên sự đánh giá về các lớp tương đương với một điều kiện vào. Lớp tương đương biểu thị cho tập các trạng thái hợp lệ hay không hợp lệ đối với điều kiện vào.

Một ca kiểm thử được lựa chọn tốt nên có 2 tính chất:

1. Giảm thiểu được số lượng các ca kiểm thử không cần khác để hoàn thành mục tiêu kiểm thử “hợp lý”.

2. Bao phủ được một tập rất lớn các ca kiểm thử khác. Tức là, chấp nhận sự có mặt hay vắng mặt của một ít lỗi qua tập giá trị đầu vào cụ thể.

Thiết kế Test-case bằng phân lớp tương đương tiến hành theo 2 bước:

Bước 1: Xác định các lớp tương đương.

Bước 2: Xác định các ca kiểm thử. Các lớp tương đương được xác định bằng bằng cách lấy mỗi trạng thái đầu vào (thường là 1 câu hay 1 cụm từ trong đặc tả) và phân chia nó thành 2 hay nhiều nhóm.

Để xác định các lớp tương đương có thể áp dụng tập các nguyên tắc dưới đây:

1. Nếu một trạng thái đầu vào định rõ giới hạn của các giá trị, xác định 1 lớp tương đương hợp lệ và 2 lớp tương đương không hợp lệ.

2. Nếu một trạng thái đầu vào xác định số giá trị, xác định một lớp tương đương hợp lệ và 2 lớp tương đương bất hợp lệ.

3. Nếu một trạng thái đầu vào chỉ định tập các giá trị đầu vào và chương trình sử dụng mỗi giá trị là khác nhau, xác định 1 lớp tương đương hợp lệ cho mỗi loại và 1 lớp tương đương không hợp lệ.

4. Nếu một trạng thái đầu vào chỉ định một tình huống “chắc chắn – must be”, xác định 1 lớp tương đương hợp lệ và 1 lớp tương đương không hợp lệ.

Nếu có bất kỳ lý do nào để tin rằng chương trình không xử lý các phần tử trong cùng một lớp là như nhau, thì hãy chia lớp tương đương đó thành các lớp tương đương nhỏ hơn.

Với các lớp tương đương xác định được ở bước trên, bước thứ hai là sử dụng các lớp tương đương đó để xác định các ca kiểm thử. Quá trình này như sau:

1. Gán một số duy nhất cho mỗi lớp tương đương.

2. Cho đến khi tất cả các lớp tương đương hợp lệ được bao phủ bởi (hợp nhất thành) các ca kiểm thử, viết một ca kiểm thử mới bao phủ càng nhiều các lớp tương đương đó chưa được bao phủ càng tốt.

3. Cho đến khi các ca kiểm thử của bạn đã bao phủ tất cả các lớp tương đương không hợp lệ, viết một ca kiểm thử mà bao phủ một và chỉ một trong các lớp tương đương không hợp lệ chưa được bao phủ.

4. Lý do mà mỗi ca kiểm thử riêng bao phủ các trường hợp không hợp lệ là vì các kiểm tra đầu vào không đúng nào đó che giấu hoặc thay thế các kiểm tra đầu vào không đúng khác.

Mặc dù việc phân lớp tương đương là rất tốt khi lựa chọn ngẫu nhiên các ca kiểm thử, nhưng nó vẫn có những thiếu sót. Ví dụ, nó bỏ qua các kiểu ca kiểm thử có lợi nào đó. Hai phương pháp tiếp theo, phân tích giá trị biên và đồ thị nguyên nhân – kết quả, bao phủ được nhiều những thiếu sót này.

Kỹ thuật Phân hoạch tương đương là một kỹ thuật của kiểm thử hộp đen. Về mặt nguyên tắc, kỹ thuật này chia (phân hoạch) miền vào của chương trình thành các lớp dữ liệu để lập ca kiểm thử theo mỗi lớp đó.

Cơ sở của kỹ thuật này là dữ liệu trong một lớp tương đương tác động như nhau lên chương trình, tạo ra cùng một trạng thái: đúng hay sai của chương trình. Mục tiêu của phương pháp phân hoạch là tìm ra một ca kiểm thử để bộc lộ một lớp sai, từ đó rút gọn số ca kiểm thử cần phát triển. Ca kiểm thử được thiết kế cho từng lớp tương đương.

## Kỹ thuật phân tích giá trị biên (Boundary Value Analysis)

Kinh nghiệm cho thấy các ca kiểm thử mà khảo sát tỷ mỷ các điều kiện biên có tỷ lệ phần trăm cao hơn các ca kiểm thử khác. Các điều kiện biên là những điều kiện mà các tình huống ngay tại, trên và dưới các cạnh của các lớp tương đương đầu vào và các lớp tương đương đầu ra.

Phân tích các giá trị biên là phương pháp thiết kế ca kiểm thử bổ sung thêm cho phân lớp tương đương, nhưng khác với phân lớp tương đương ở 2 khía cạnh:

1. Phân tích giá trị biên không lựa chọn phần tử bất kỳ nào trong 1 lớp tương đương là điển hình, mà nó yêu cầu là 1 hay nhiều phần tử được lựa chọn như vậy mà mỗi cạnh của lớp tương đương đó chính là đối tượng kiểm tra.

2. Ngoài việc chỉ tập trung chú ý vào các trạng thái đầu vào (không gian đầu vào), các ca kiểm thử cũng nhận được bằng việc xem xét không gian kết quả (các lớp tương đương đầu ra).

Phân tích giá trị biên yêu cầu óc sáng tạo và lượng chuyên môn hóa nhất định và nó là một quá trình mang tính kinh nghiệm rất cao. Tuy nhiên, có một số quy tắc chung như sau:

1. Nếu một trạng thái đầu vào định rõ giới hạn của các giá trị, hãy viết các ca kiểm thử cho các giá trị cuối của giới hạn, và các ca kiểm thử đầu vào không hợp lệ cho các trường hợp vừa ra ngoài phạm vi.

2. Nếu một trạng thái đầu vào định rõ số lượng giá trị, hãy viết các ca kiểm thử cho con số lớn nhất và nhỏ nhất của các giá trị và một giá trị trên, một giá trị dưới những giá trị này.

3. Sử dụng quy tắc 1 cho mỗi trạng thái đầu vào.

## Kỹ thuật đồ thị nguyên nhân – kết quả (Cause – Effect Graphing)

Một yếu điểm của phân tích giá trị biên và phân lớp tương đương là chúng không khảo sát sự kết hợp của các trường hợp đầu vào. Việc kiểm tra sự kết hợp đầu vào không phải là một nhiệm vụ đơn giản bởi vì nếu phân lớp tương đương các trạng thái đầu vào, thì số lượng kết hợp thường là rất lớn. Nếu không có cách lựa chọn có hệ thống một tập con các trạng thái đầu vào thì khi chọn ra một tập tùy hứng các điều kiện, điều này có thể dẫn tới việc kiểm thử không có hiệu quả.

Đồ thị nguyên nhân – kết quả hỗ trợ trong việc lựa chọn một cách có hệ thống tập các ca kiểm thử có hiệu quả cao. Nó có tác động có lợi ảnh hưởng tới việc chỉ ra tình trạng chưa đầy đủ và nhập nhằng trong đặc tả. Nó cung cấp cả cách biểu diễn chính xác cho các điều kiện logic và hành động tương ứng

Quá trình dưới đây được sử dụng để xây dựng được các ca kiểm thử:

1. Đặc tả được chia thành các phần có thể thực hiện được. Điều này là cần thiết bởi vì đồ thị nguyên nhân – kết quả trở nên khó sử dụng khi được sử dụng trên những đặc tả lớn.

2. Nguyên nhân và kết quả trong các đặc tả được nhận biết. Một nguyên nhân là một trạng thái đầu vào nhất định hay một lớp tương đương của các trạng thái đầu vào. Một kết quả là một trạng thái đầu ra hay một sự biến đổi hệ thống (kết quả còn lại mà một đầu vào có trạng thái của một chương trình hay hệ thống). Bạn nhận biết nguyên nhân và kết quả bằng việc đọc từng từ của đặc tả và gạch chân các từ hoặc cụm từ mô tả nguyên nhân và kết quả. Khi được nhận biết, mỗi nguyên nhân và kết quả được gán cho mộtsố duy nhất.

3. Xây dựng đồ thị nguyên nhân – kết quả bằng cách phát triển và biến đổi nội dung ngữ nghĩa của đặc tả thành đồ thị Boolean nối giữa nguyên nhân và kết quả.

4. Đồ thị được được diễn giải với các ràng buộc mô tả những sự kết hợp của nguyên nhân và/hoặc kết quả là không thể vì các ràng buộc ngữ nghĩa và môi trường.

5. Bằng việc dò theo các điều kiện trạng thái trong đồ thị một cách cẩn thận, bạn chuyển đổi đồ thị thành một bảng quyết định mục vào giới hạn. Mỗi cột trong bảng mô tả một ca kiểm thử.

Kỹ thuật đồ thị nhân- quả là một kỹ thuật để thiết kế ca kiểm thử. Nó cung cấp một biểu diễn chính xác giữa các điều kiện logic (đầu vào) và các hành động tương ứng (đầu ra- kết quả).

Kỹ thuật đồ thị nhân- quả được xây dựng dựa trên các mô-đun chức năng, lôgíc tiến trình và đặc tả hệ thống.

## Kỹ thuật đoán lỗi (Error Guessing)

Một kỹ thuật thiết kế ca kiểm thử khác là error guessing – đoán lỗi. Người kiểm thử được đưa cho một chương trình đặc biệt. Họ phỏng đoán, cả bằng trực giác và kinh nghiệm, các loại lỗi có thể và sau đó viết các ca kiểm thử để đưa ra các lỗi đó.

Thật khó để đưa ra một quy trình cho kỹ thuật đoán lỗi vì nó là một quy trình có tính trực giác cao và không thể dự đoán trước.

Ý tưởng cơ bản là liệt kê một danh sách các lỗi có thể hay các trường hợp dễ xảy ra lỗi và sau đó 61 viết các ca kiểm thử dựa trên danh sách đó. Một ý tưởng khác để xác định các ca kiểm thử có liên đới với các giả định mà lập trình viên có thể đã thực hiện khi đọc đặc tả (tức là, những thứ bị bỏ sót khỏi đặc tả, hoặc là do tình cờ, hoặc là vì người viết có cảm giác những đặc tả đó là rõ ràng).

Nói cách khác, cần liệt kê những trường hợp đặc biệt đó mà có thể đã bị bỏ sót khi chương trình được thiết kế. Các phương pháp thiết kế ca kiểm thử có thể được kết hợp thành một “kinh nghiệm” thiết kế ca kiểm thử. Vì mỗi phương pháp có thể đóng góp một tập riêng các ca kiểm thử hữu dụng, nhưng không cái nào trong số chúng tự nó đóng góp một tập trọn vẹn các ca kiểm thử.

Kinh nghiệm thiết kế ca kiểm thử hợp lý có thể tóm lược như sau:

1. Nếu đặc tả có chứa sự kết hợp của các điều kiện đầu vào, hãy bắt đầu với việc vẽ đồ thị nguyên nhân – kết quả.

2. Trong trường hợp bất kỳ, sử dụng phương pháp phân tích giá trị biên. Hãy nhớ rằng đây là một sự phân tích của các biên đầu vào và đầu ra. Phương pháp phân tích giá trị biên mang lại một tập các điều kiện kiểm tra bổ sung, và rất nhiều hay toàn bộ các điều kiện này có thể được hợp nhất thành các kiểm thử nguyên nhân – kết quả.

3. Xác định các lớp tương đương hợp lệ và không hợp lệ cho đầu vào và đầu ra, và bổ sung các ca kiểm thử được xác định trên nếu cần thiết.

4. Sử dụng kỹ thuật đoán lỗi để thêm các ca kiểm thử thêm vào.

5. Khảo sát tính logic của chương trình liên quan đến tập các ca kiểm thử. Sử dụng tiêu chuẩn bao phủ quyết định, bao phủ điều kiện, bao phủ quyết định/điều kiện, hay bao phủ đa điều kiện (trong đó bao phủ đa điều kiện là được sử dụng nhiều nhất). Nếu tiêu chuẩn bao phủ không đạt được bởi các ca kiểm thử được xác định trong bốn bước trước, và nếu việc đạt được tiêu chuẩn là không thể (tức là, những sự kết hợp chắc chắn của các điều kiện có thể là không thể tạo vì bản chất của chương trình), hãy thêm vào các ca kiểm thử có khả năng làm cho thỏa mãn tiêu chuẩn.

Các kinh nghiệm trên tuy chưa bảo đảm rằng tất cả các lỗi sẽ được tìm thấy nhưng nó rất hữu ích cho việc thực hành thiết lập các ca kiểm thử.

# Kiểm thử hộp đen



## Khái niệm

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.

Nó còn được gọi là kiểm thử hướng dữ liệu hay là kiểm thử hướng in/out.

Người kiểm thử nên xây dựng các nhóm giá trị đầu vào mà sẽ thực thi đầy đủ tất cả các yêu cầu chức năng của chương trình.

Cách tiếp cận của các tester 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.

Black Box Testing chủ yếu là được thực hiện trong Function test và System test.

Phương pháp này được đặt tên như vậy bởi vì các chương trình phần mềm, trong con mắt của các tester, giống như một hộp đen; bên trong mà người ta không thể nhìn thấy. Phương pháp này cố gắng tìm ra các lỗi trong các loại sau:

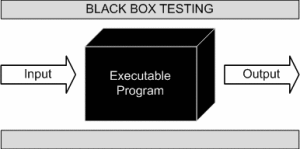
#### Chức năng không chính xác hoặc thiếu.

#### Lỗi giao diện.

#### Lỗi trong cấu trúc dữ liệu hoặc truy cập cơ sở dữ liệu bên ngoài.

#### Hành vi hoặc hiệu suất lỗi.

#### Khởi tạo và chấm dứt các lỗi.



Mọi kỹ thuật nào cũng có ưu điểm và nhược điểm của nó. Các hệ thống thường phải được sử dụng nhiều phương pháp kiểm thử khác nhau để đảm bảo được chất lượng của hệ thống khi đến tay người dùng.

## Ưu điểm

#### Các tester theo phương pháp black box không có “mối ràng buộc” nào với code, và nhận thức của một tester rất đơn giản: một source code có nhiều lỗi. Sử dụng nguyên tắc, "Hỏi và bạn sẽ nhận" các tester black box tìm được nhiều bug ở nơi mà các DEV không tìm thấy.

#### Các tester được thực hiện từ quan điểm của người dùng và sẽ giúp đỡ trong việc sáng tỏ sự chênh lệch về thông số kỹ thuật.

#### 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 được thực hiện bởi một cơ quan độc lập từ các developer, cho phép một cái nhìn khách quan và tránh sự phát triển thiên vị.

#### Hệ thống thật sự với toàn bộ yêu cầu của nó được kiểm thử chính xác.

#### Thiết kế kịch bản kiểm thử khá nhanh, ngay khi mà các yêu cầu chức năng được xác định.

## Nhược điểm

#### 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.

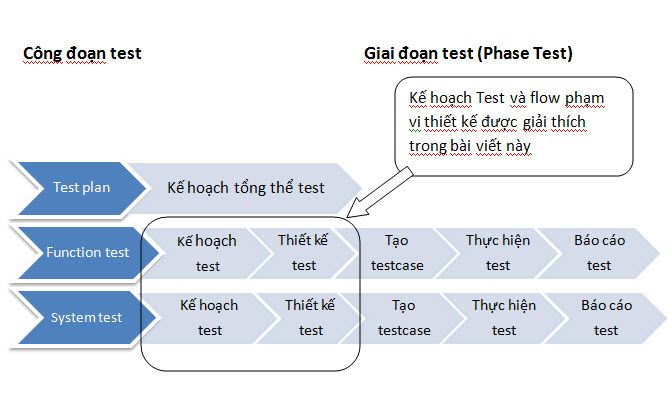
#### Dữ liệu đầu vào yêu cầu một khối lượng mẫu (sample) khá lớn

#### Khả năng để bản thân kỹ sư lạc lối trong khi kiểm thử là khá cao.

#### Chỉ có một số nhỏ các đầu vào có thể được kiểm tra và nhiều đường dẫn chương trình sẽ được để lại chưa 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.

## Flow của Black-box testing



Hình 1: Phase test trong công đoạn test



### Nội dung công việc trong công đoạn test

Công đoạn test có lần lượt các phase sau: Kế hoạch test, thiết kế test, tạo testcase, thực hiện test, báo cáo test.

Trong đó: "kế hoạch test" và "thiết kế test" là phase quan trọng để phát hiện ra lỗi và xác nhận chất lượng.

#### **Kế hoạch test**: Chỉ ra rõ ràng mục đích và phạm vi của công đoạn test để kiểm tra xem là test bằng approach như thế nào. Điều chỉnh resource thành viên và quyết định cả schedule.

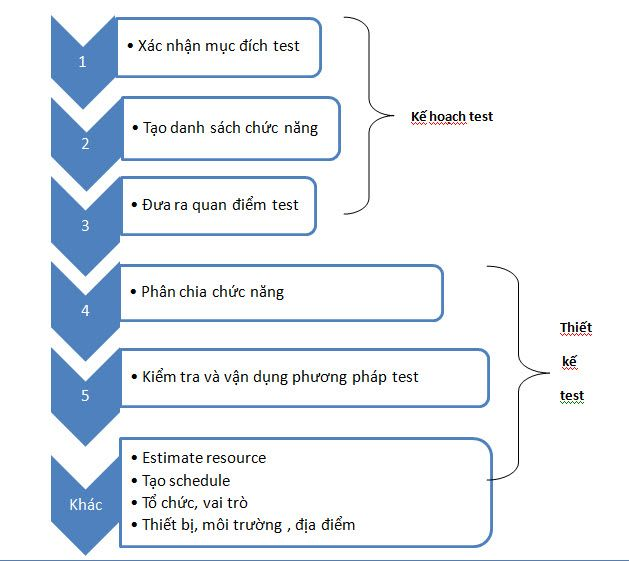
#### **Thiết kế test**: Quyết định xem là sẽ sử dụng cái gì cho mục đích và loại test càn được thực hiện trong công đoạn test đó, chức năng đối tượng test, phương pháp test, import và export test. Ngoài ra cũng quyết định cụ thể hơn nguyên liệu cần thiết để thực hiện test hay tiêu chuẩn quyết định thành công/ không thành công.

#### **Tạo testcase**: Tạo document ghi trạng thái trước khi bắt đầu test và kết quả mong đợi (kết quả chạy đối tượng test theo điều kiện và trình tự thao tác khi thực hiện test sẽ như thế nào) và cột trạng thái (cột ghi lại kết quả thao tác của đối tượng test).

#### **Thực hiện test**: Vừa xem testcase vừa cho chạy phần mềm thực tế để tiến hành test, sau đó đánh dấu kết quả bằng dấu pass hoặc fail vào cột trạng thái testcase. Trường hợp có testcase khác với kết quả mong đợi thì ghi dấu fail vào cột trạng thái, rồi tạo bản báo cáo lỗi. Trong bản báo cáo lỗi: trình bày nội dung mô tả hiện tượng khác với kết quả mong đợi và hiện tượng đó phát sinh trong trường hợp như thế nào (thao tác, giả nhập, điều kiện,...)

#### **Báo cáo test**: Tóm tắt kết quả để báo cáo. Căn cứ vào các loại dữ liệu (mục thực hiện, hiệu quả của việc test, công số thực hiện,...) và dữ liệu lỗi (số lỗi được tìm ra, số lỗi theo mức độ quan trọng,...) để đánh giá xem có thỏa mãn tiêu chuẩn pass/ fail của test không? Ngoài ra cũng đề xuất thêm risk có thể sinh ra sau khi release và mục cần bổ sung trong dự án cho giai đoạn tiếp theo.

### Flow của plan và kế hoạch test



Hình 2: Flow kế hoạch test và thiết kế test

#### Xác nhận mục đích test

Xem bản kế hoạch tổng thể test để xác nhận mục đích của test(đặc tính chất lượng, những điểm quan trọng ...).

Quyết định phạm vi của test, nội dung của test, phương pháp test.

##### Tạo danh sách chức năng

##### Đưa ra toàn bộ chức năng làm đối tượng test.

##### Cần hiểu trước về phần lớn các hoạt động của chức năng.

##### Không phán đoán đối tượng hoặc phi đối tượng của test.

##### Đưa ra qua điểm test

##### Quan điểm test là " cánh cửa của Test ".

##### - Quan điểm test đã đáp ứng được đúng mục đích test?

##### Phân chia chức năng cho từng quan điểm test

##### Áp dụng quan điểm test cho từng chức năng để tránh bị quên.

##### Có thể hình dung việc test một cách cụ thể.

##### Có thể nắm bắt được quy mô test và mức độ quan trọng của các quan điểm test.

##### Kiểm tra vận dụng phương pháp và kỹ thuật test

##### Tiến hành thiết kế test dối với lần lượt từng kết hợp.

##### Lựa chọn và quyết định phương pháp test có thể phát hiện ra lỗi một cách hiệu quả nhất từ một trong số các phương pháp test.

##### Các mục kiểm tra khác

##### Resouce cần thiết.

##### Schedule.

##### Cơ cấu & tổ chức, vai trò khi thực hiện test.

##### Thiết bị, môi trường, địa điểm làm việc cần để thực hiện test.

## Phương pháp truy xuất và tính cần thiết của quan điểm test

### Danh sách quan điểm test

Khái niệm: Danh sách quan điểm test là danh sách tóm tắt quan điểm test theo hình thức có thể tái sử dụng.

Có thể thực hiện test hệ thống, mà không phải test nhờ vào kinh nghiệm cá nhân.

Có thể giảm nhẹ được sự chênh lệch kỹ năng cá nhân. (chênh lệch giữa người mới và chuyên gia,...).

Có thể tránh trùng, hoặc thiếu sót quan điểm phải test.

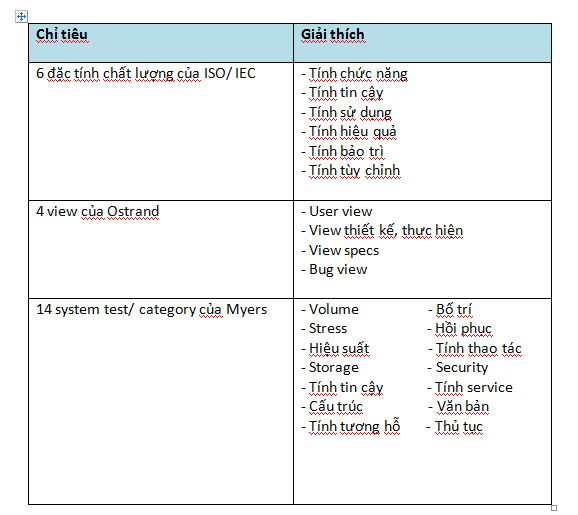
Phán đoán mức độ quan trọng trong dự án dễ dàng.

### Phương pháp tạo danh sách quan điểm test\

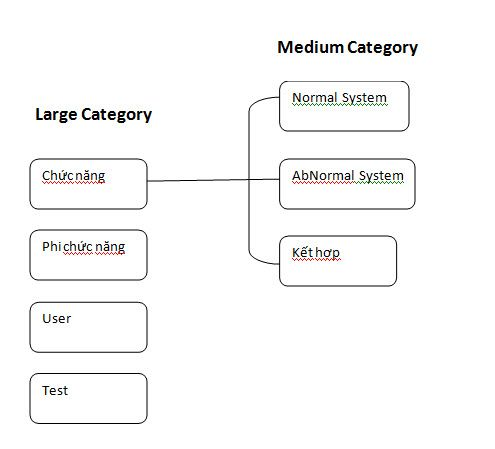
Tạo danh sách quan điểm test, phân quan điểm test thành các giai đoạn, tiếp tục phân quan điểm test lớn thành các quan điểm test bé hơn.

Nếu có tiêu chuẩn phân loại, thì công việc trở nên dễ dàng hơn.

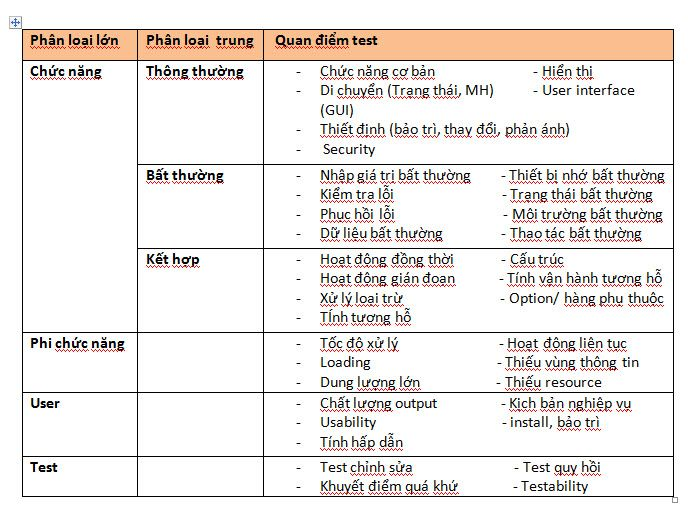
Ví dụ: Lấy các tiêu chí dưới đây làm tiêu chuẩn phân loại.



Ví dụ phân loại quan điểm test



Ví dụ danh sách quan điểm Test



## Phương pháp kiểm thử hộp đen

**Đoán lỗi**: là một kỹ năng quan trọng của tester, thậm chí có thể gọi là nghệ thuật. Một kiệt tác của trực giác. Phương pháp này đặc biệt dựa vào kinh nghiệm và kiến thức của tester. Nhiều tester cố gắng đoán xem phần nào của hệ thống mà có khả năng ẩn chứa lỗi. Với phương pháp này, họ không cần một công cụ hay một kịch bản kiểm thử nào khi bắt đầu vào việc.

**Kiểm thử dựa vào đồ thị nguyên nhân - kết quả (Cause Effect Graphing**): là một kỹ thuật thiết kế kiểm thử phần mềm liên quan đến việc xác định các trường hợp (điều kiện đầu vào) và các hiệu ứng (điều kiện đầu ra). Vì các hệ thống hiện nay đều được phát triển trên nền tảng OOP, do đó, chúng ta có thể có được một đồ thị các đối tượng mà hệ thống định nghĩa và kết nối. Từ đồ thị này, chúng ta dễ dàng biết các mối quan hệ của những đối tượng mà hệ thống xử lý, từ đó sẽ cho chúng ta các kịch bản kiểm thử phù hợp.

**Phân vùng tương đương (Equivalence Class):** là một kỹ thuật kiểm thử phần mềm có liên quan đến phân chia các giá trị đầu vào thành các phân vùng hợp lệ và không hợp lệ, sau đó chúng ta sẽ viết ra các kịch bản kiểm thử cho từng phần, chọn giá trị đại diện từ mỗi phân vùng làm dữ liệu thử nghiệm.

Phân vùng tương đương: là kỹ thuật thực hiện test theo từng class đồng giá trị (tập hợp điều kiện cùng một thao tác).

Tập hợp giá trị input có cùng một kết quả xử lý, tập hợp thời gian có cùng một kết quả xử lý, tập hợp kết quả export được xử lý cùng một giá trị nhập.

Mục đích :Giảm đáng kể số lượng test case cần phải thiết kế vì với mỗi lớp tương đương ta chỉ cần test trên các phần tử đại diện.

Chọn tối thiểu một giá trị đại diện từ các class đồng giá trị để tiến hành test.

--> Nếu có lỗi xảy ra thì các giá trị khác trong class đồng giá trị cũng sẽ có lỗi giống nhau.

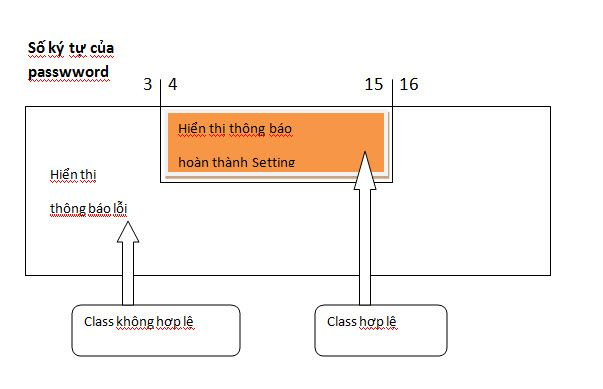
Ví dụ:

Spec yêu cầu nhập 4 <= password <= 15

Nhập đúng đưa ra mess hoàn thành thiết lập.

Nhập sai: mess yêu cầu nhập lại.

Như vậy ta sẽ thực hiện 2 testcase: một giá trị cho phần thông báo hoàn thành Setting và một giá trị phần thông báo lỗi.



**Phân tích giá trị biên (Boundary Value Analysis):** là một kỹ thuật kiểm thử phần mềm có liên quan đến việc xác định biên (ranh giới) của điều kiện mô tả cho các giá trị đầu vào và chọn giá trị ở biên và bên cạnh giá trị biên làm dữ liệu kiểm thử. Phương pháp phân tích giá trị biên sẽ đưa ra các giá trị đặc biệt, bao gồm loại dữ liệu, giá trị lỗi, bên trong, bên ngoài biên giá trị, lớn nhất và nhỏ nhất.

Những kỹ sư nhiều kinh nghiệm chắc chắn đã từng gặp phải các lỗi của hệ thống ngay tại giá trị biên. Đó là lý do tại sao phân tích giá trị biên lại quan trọng khi kiểm thử hệ thống.

Test giá trị biên được thực hiện theo trình tự dưới đây:

1) Tìm ra đường biên

2) Quyết định giá trị biên

3) Quyết định giá trị để test

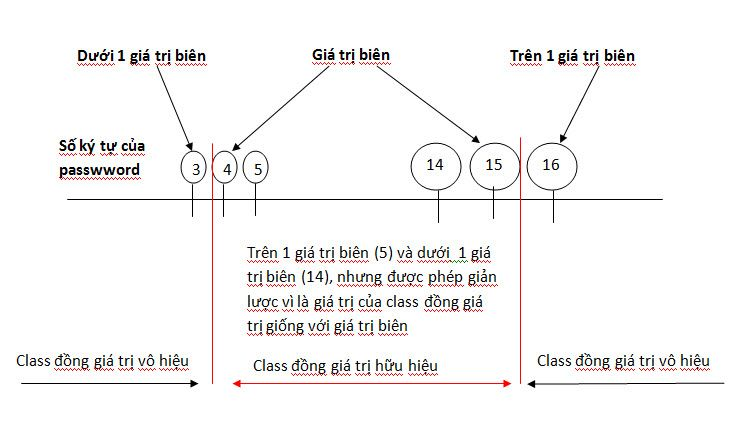
- Giá trị biên.

- Dưới giá trị biên. (Nếu là class đồng giá trị)

- Trên 1 giá trị biên. (Nếu là class đồng giá trị)

Ví dụ

Spec yêu cầu nhập 4 <= password <= 15 Giá trị được test sẽ như sau:



Sử dụng bảng quyết định (Decision Tables)

#### Là dùng bảng để hiển thị danh sách các thao tác phần mềm được quyết định trên các điều kiện khác nhau.

#### Decision table testing chú trọng vào nhiều điều kiện để thực hiện test.

Ngoài ra, kiểm thử hộp đen còn có một số kỹ thuật như: Domain Tests, Orthogonal Arrays, State Models, Exploratory Testing (kiểm thử chủ yếu dựa vào kinh nghiệm và khả năng focus vào việc test các chức năng của tester), All-pairs testing (kiểm thử tất cả các cặp), ...

# Xác định các ca kiểm thử và kỹ thuật kiểm thử cho module mã hóa văn bản



## Giới thiệu bài toán

#### Ứng dụng kiểm thử hộp đen cho mô đun mã hoá dữ liệu văn bản

#### Việc phát triển không ngừng của lĩnh vực công nghệ thông tin trong giai đoạn hiện nay đã giúp cuộc sống có những thay đổi đáng. Công nghệ thông tin là một phần không thể thiếu của cuộc sống văn minh, góp phần đẩy mạnh công cuộc công nghiệp hóa hiện đại hóa đất nước.

#### Đi kèm với những phát triển của ngành công nghệ thông tin không thể không kể đến những vấn đề bảo mật về thông tin trên không gian mạng. Module mã hoá dữ liệu văn bản của chúng em sử dụng hệ mã hoá DES giúp mã hoá và bảo mật các thông tin văn bản người dùng gửi đi.

#### Trong module này nhóm sẽ sử dụng kỹ thuật kiểm thử hộp đen (phương pháp phân vùng tương đương)

## Thiết kế các ca kiểm thử

#### Ta xác định phân vùng tương đương theo yêu cầu sau

##### File đầu vào là file text

##### Nếu đầu vào không phải file text thì sẽ báo lỗi

##### Nếu đầu vào để trống thì sẽ báo lỗi

|  |  |  |  |
| --- | --- | --- | --- |
| Đầu vào | Không nhập file | File text | File khác |
| Kết quả | Không hợp lệ | Mã hóa | Hợp lệ |
| Lớp tương đương | Không hợp lệ | Hợp lệ | Không hợp lệ |

#### Dựa vào bảng trên ta có thể thiết kế các ca kiểm thử sau

##### Case 1: nhập file đầu vào là file text => pass

##### Case 2: không nhập file đầu vào => báo lỗi

##### Case 3: nhập file đầu vào không phải file text=>báo lỗi

## Tài liệu tham khảo

- “*Software quality assurance from theory to implementation*”

- “*Mastering software quality assurance best practices tools and techniques for software developers*”