TRƯỜNG ĐẠI HỌC KINH TẾ

**KHOA THỐNG KÊ – TIN HỌC**

–––––––––––––––––––––––––––––––



**BÁO CÁO THỰC TẬP NGHỀ NGHIỆP**

**THỰC HIỆN KIỂM THỬ DỰ ÁN PHẦN MỀM QUẢN LÝ HỢP ĐỒNG ĐIỆN TỬ**

**Sinh viên thực hiện**: Huỳnh Ngọc Trâm

**Lớp:** 44K21.2

**Giảng viên hướng dẫn:** Cao Thị Nhâm

***Đà Nẵng, 7/2021***

**LỜI CẢM ƠN**

**LỜI CAM ĐOAN**

Chúng em xin cam đoan dự án **THỰC HIỆN KIỂM THỬ DỰ ÁN PHẦN MỀM QUẢN LÝ HỢP ĐỒNG ĐIỆN TỬ** là kết quả nghiên cứu độc lập dưới sự hướng dẫn của giáo viên hướng dẫn TS. Cao Thị Nhâm. Ngoài ra không có bất cứ sự sao chép nào của người khác.

Dự án, nội dung báo cáo là sản phẩm mà em đã nỗ lực nghiên cứu trong quá trình học tập cũng như tìm hiểu trên ở nhiều phương tiện. Các dữ liệu, kết quả trình bày trong báo cáo là hoàn toàn trung thực, em xin chịu hoàn toàn trách nhiệm, kỷ luật của bộ môn và nhà trường đề ra nếu như có vấn đề xảy ra.

# **MỤC LỤC**

[**MỤC LỤC i**](#_Toc76397997)

[**DANH MỤC HÌNH ẢNH ii**](#_Toc76397998)

[**DANH MỤC BẢNG BIỂU iii**](#_Toc76397999)

[**DANH MỤC CÁC TỪ VIẾT TẮT iv**](#_Toc76398000)

[**LỜI MỞ ĐẦU 1**](#_Toc76398001)

[**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 2**](#_Toc76398002)

[**1.1 Tổng quan về Testing 2**](#_Toc76398003)

[**1.1.1 Tài liệu SRS 2**](#_Toc76398004)

[**1.1.2 Test Type (Loại kiểm thử) 2**](#_Toc76398005)

[**1.1.3 Test Level (Mức độ kiểm thử) 6**](#_Toc76398006)

[**1.1.4 Test Method (Phương pháp kiểm thử) 12**](#_Toc76398007)

[**1.1.4.1.1 Định nghĩa 12**](#_Toc76398008)

[**1.1.4.1.2 Đối tượng được kiểm thử 12**](#_Toc76398009)

[**1.1.4.1.3 Phương pháp thử nghiệm 13**](#_Toc76398010)

[**1.1.4.2.1 Định nghĩa 13**](#_Toc76398011)

[**1.1.4.2.2 Đối tượng kiểm thử 13**](#_Toc76398012)

[**1.1.4.2.3 Phương pháp thử nghiệm 14**](#_Toc76398013)

[**1.1.4.3.2 Phương pháp thử nghiệm 14**](#_Toc76398014)

[**1.1.5 Kĩ thuật kiểm thử hộp đen 15**](#_Toc76398015)

[**1.2 Tổng quan về nghề nghiệp 17**](#_Toc76398016)

[**1.2.1 Vị trí làm việc 17**](#_Toc76398017)

[**1.2.2 Mô tả công việc của nhân viên Tester 17**](#_Toc76398018)

[**1.2.3 Mức lương 18**](#_Toc76398019)

[**CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG 19**](#_Toc76398020)

[**2.1 Mô tả bài toán 19**](#_Toc76398021)

[**2.2 Thiết kế các chức năng 19**](#_Toc76398022)

[**2.3 Thiết kế cơ sở dữ liệu 19**](#_Toc76398023)

[**CHƯƠNG 3. KẾT QUẢ 20**](#_Toc76398024)

[**3.1 Kết quả thực hiện thiết kế testcase 20**](#_Toc76398025)

[**3.2 Kết quả test hệ thống 20**](#_Toc76398026)

[**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 21**](#_Toc76398027)

[**TÀI LIỆU THAM KHẢO 22**](#_Toc76398028)

[**PHỤ LỤC (nếu có) 23**](#_Toc76398029)

# DANH MỤC HÌNH ẢNH

[Hình 1: Các loại kiểm thử 9](#_Toc76391368)

[Hình 2: Phương pháp kiểm thử bigbang 14](#_Toc76391369)

[Hình 3: Phương pháp kiểm thử Top down 15](#_Toc76391370)

[Hình 4: Phương pháp kiểm thử Botton up 15](#_Toc76391371)

# DANH MỤC BẢNG BIỂU

[Bảng 1: So sánh kiểm thử chức năng và kiểm thử phi chức năng 11](#_Toc76391383)

[Bảng 2: So sánh các mức độ kiểm thử 18](#_Toc76391384)

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Viết tắt** | **Viết đầy đủ** |
| TPPM | Thành phần phần mềm |

# LỜI MỞ ĐẦU

1. **Mục tiêu của đề tài**

* Nghiên cứu

1. **Nhiệm vụ của đề tài**
2. **Kết cấu của đề tài**

# CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

## Tổng quan về Testing

### **Tài liệu SRS**

#### **Khái niệm**

Tài liệu SRS là từ viết tắt của Software Requirement Specification, được dịch ra tiếng việt là tài liệu đặc tả yêu cầu. SRS là tài liệu được sử dụng để mô tả chi tiết các yêu cầu chức năng và phi chức năng của hệ thống. Tài liệu này sẽ hỗ trợ đưa ra các tính năng của hệ thống hay dùng cho việc đọc hiểu hệ thống của bên thứ ba liên quan đến công ty. Đây là một tài liệu quan trọng cho đội phát triển (system analyst, business analyst, code) và kiểm thử (tester).

#### **Tầm quan trọng của tài liệu SRS**

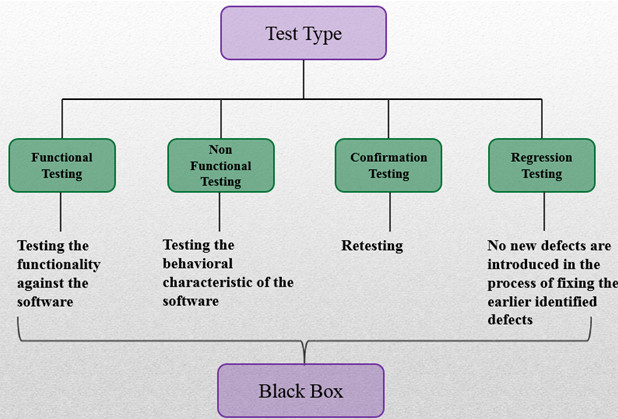
SRS là tài liệu đặc tả vô cùng quan trọng trong quá trình phát triển phần mềm, nó có vai trò:

* Giúp cho các bên thứ ba – stakeholders đều hiểu được hệ thống theo cùng một hướng, tránh trường hợp mỗi người một ý.
* Giúp cho đội phát triển xây dựng hệ thống một cách chính xác, đặc tả được các tính năng, không đi lạc hướng so với yêu cầu của khách hàng.
* SRS giúp nhà kiểm thử hệ thống đọc hiểu từ đó xây dựng nên kịch bản kiểm thử chi tiết nhất.
* Giúp cho việc bảo trì hệ thống và cải tiến những chức năng của hệ thống một cách nhanh chóng và dễ dàng.

### **Test Type (Loại kiểm thử)**

Có tất cả 4 loại Test Types:

1. Testing of function  (Functional testing)
2. Testing of software product characteristics (Non – Functional testing)
3. Testing of software structure/architecture ( Structural testing)
4. Testing related to changes (Confirmation and regression testing)

[](https://tanhongit.com/wp-content/uploads/2020/05/cac-loai-kiem-thu-test-types.jpg)

Hình 1: Các loại kiểm thử

#### **Testing of function (Functional testing)\_ Kiểm thử chức năng**

Functional testing hay còn gọi kiểm thử chức năng được các Tester(người kiểm thử) thực hiện để kiểm tra hoạt động của một chức năng nào đó có được hoạt động ổn định cũng như tốc độ phản hồi của chức năng có ổn định hay không.

Kiểm thử chức năng có thể thực hiện theo 2 quan điểm

Đó chính là requirements-based và business – process – based.

1. Requirements – based: Dùng đặc tả yêu cầu dùng để làm cơ sở để thiết kế các test case (design test).Chúng ta nên xét độ ưu tiên của yêu cầu dựa trên các tiêu chí rủi ro và sử dụng độ ưu tiên để kiểm thử. Điều này sẽ đảm bảo những phần quan trọng nhất sẽ được kiểm tra đầy đủ.
2. Business – process – based: sử dụng các kiến thức về quy trình nghiệp vụ. Quy trình nghiệp vụ mô tả các kịch bản liên quan đến nghiệp vụ hằng ngày của hệ thống.

Kiểm thử chức năng bao gồm 5 bước:

1. Xác định các chức năng mà phần mềm mong muốn sẽ thực hiện.
2. Tạo các dữ liệu đầu vào dựa trên các tài liệu đặc tả kỹ thuật SRS của các chức năng.
3. Xác định các kết quả đầu ra dựa trên các tài liệu đặc tả kỹ thuật của các chức năng.
4. Thực hiện các trường hợp kiêm thử.
5. So sánh kết quả thực tế và kết quả mong muốn.

#### **Testing of software product characteristics – Kiểm thử phi chức năng (Non – Functional testing)**

Kiểm thử phi chức năng(Non – Functional testing) là kiểm tra các đặc tính chất lượng của hệ thống, các tester sẽ quan tâm đến việc mọi chức hoạt động tốt như thế nào, tốc độ nhanh như thế nào, phản hồi và số lượng người truy cập vào cùng một thời điểm cũng như xem mức độ sử dụng CPU, Ram, Pin,… của phần mềm,…

Kiểm thử phi chức năng bao gồm:

1. Kiểm thử hiệu năng (Performance testing)
2. Kiểm thử khả năng chịu tải (Load testing)
3. Kiểm thử áp lực (Stress testing)
4. Kiểm thử tính khả dụng (Usability testing)
5. Kiểm thử bảo trì (Maintainability testing)
6. Kiểm thử độ tin cậy (Reliability testing)
7. Kiểm thử tính tương thích (Portability testing)

|  |  |
| --- | --- |
| **Kiểm thử chức năng** | **Kiểm thử phi chức năng** |
| Kiểm thử chức năng được thực hiện sử dụng kỹ thuật cung cấp bởi người dùng và xác thực yêu cầu của hệ thống | Kiểm thử phi chức năng kiểm tra hiệu suất, độ tin cậy, khả năng mở rộng và các khía cạnh khác của hệ thống phần mềm |
| Kiểm thử chức năng được thực thi đầu tiên | Thực hiện sau thực hiện chức năng |
| Công cụ sử dụng: test bằng tay hoặc test tự động | Sử dụng các công cụ |
| Đầu vào là yêu cầu nghiệp vụ | Đầu vào là tốc độ , khả năng mở rộng , những tham số về hiệu suất |
| Đầu vào là yêu cầu nghiệp vụ | Mô tả hiệu quả của sản phẩm |
| Dễ dàng test bằng tay | Khó kiểm tra bằng tay |
| Các loại kiểm thử chức năng: Kiểm thử đơn vị, Kiểm thử sơ lược, Kiểm thử độ bền, Kiểm thử tích hợp, Kiểm thử hộp đen, Kiểm thử hộp trắng, Kiểm thử người dùng chấp nhận, Kiểm thử hồi quy | Các loại kiểm thử phi chức năng: Kiểm thử hiệu suất, Kiểm thử việc tải, Kiểm thửu khả năng chịu tải Kiểm thử an toàn, Kiểm thử cài đặt, Kiểm thử áp lực, Kiểm thử thâm nhập, Kiểm thử tính tương thích, Kiểm thử di chuyển |

Bảng 1: So sánh kiểm thử chức năng và kiểm thử phi chức năng

#### **Testing of software structure/architecture ( Structural testing)**

Kiểm thử cấu trúc thường được gọi là “kiểm thử hộp trắng“(white-box) hoặc “hộp thủy tinh” (glass-box) vì chúng quan tâm đến những gì đang xảy ra bên trong hộp.

Kiểm thử cấu trúc là kiểm thử dựa trên phân tích cấu trúc bên trong của thành phần hoặc hệ thống. Nó thường được sử dụng như một cách đo lường của kiểm thử, thông qua độ bao phủ của một tập hợp các yếu tố cấu trúc.

Kiểm thử cấu trúc chủ yếu được áp dụng ở Kiểm Thử Thành Phần, Kiểm Thử Tích Hợp.

Ở cấp độ thành phần, và mức thấp hơn trong kiểm thử tích hợp thành phần có hỗ trợ công cụ tốt để đo mức độ bao phủ của code. Các công cụ đo lường mức độ bao phủ đánh giá tỉ lệ phần trăm thực thi sau khi được thực hiện bởi một bộ kiểm thử. Nếu bộ bao phủ ko phải tuyệt đối thì các kiểm thử bổ sung có thể cẩn được viết và chạy để có thể bao phủ hết các phần còn lại.

Các kỹ thuật được dùng để kiểm tra cấu trúc là kỹ thuật kiểm thử  “hộp trắng”, các mô hình luồng điều khiển thông thường được dùng để kiểm thử các cấu trúc.

#### **Testing related to changes (Confirmation and regression testing)**

##### **Confirmation testing (Kiểm thử xác nhận)**

Khi kiểm thử bị lỗi, và tester phải xác định nguyên nhân lỗi là do lỗi phần mềm. Sau khi Tester phát hiện lỗi và báo cho Developer để sửa chữa thì phần mềm sau đó sẽ được cập nhật phiên bản vá lỗi tester cần thực hiện kiểm tra một lần nữa để xác định rằng lỗi thực sự đã được sửa.

Khi thực hiện kiểm tra xác nhận điều quan trọng nhất là phải đảm bảo rằng các trường hợp thử phải được thực hiện chính xác giống như lần đầu tiên, sử dụng cùng một đầu vào, dữ liệu và môi trường để kiểm thử đảm bảo rằng các lỗi đã được sửa.

Tester cần phải biết rằng trong lần kiểm thử sau khi được vá lỗi khả năng sinh ra lỗi khác trong phần mềm là điều hiển nhiên vì vậy kiểm chính xác ở phiên bản hiện tại của một phần mềm là chưa đủ. Cách phát hiện các bất lợi ngoài ý muốn của việc kiểm lỗi là thực hiện kiểm thử hồi quy.

##### **Regression testing (Kiểm thử hồi quy)**

Giống như kiểm thử xác nhận thì bên kiểm thử hồi quy liên quan đến việc thực hiện các trường hợp kiểm thử đã được thực hiện trước đó.

Mục đích của kiểm thử hồi quy là để xác minh rằng các sửa đổi trong phần mềm hoặc môi trường không gây ra bất lợi ngoài ý muốn, ảnh hưởng hoặc làm hư các chức năng liên quan không và hệ thống vẫn đáp ứng các yêu cầu của phần mềm.

Bộ kiểm thử hồi quy hoặc gói kiểm tra hồi quy là một tập hợp các trường hợp kiểm thử được sử dụng đặc biệt để kiểm tra hồi quy.  Bộ kiểm thử hồi quy được thiết kế để thực hiện kiểm hầu hết các chức năng trong một hệ thống nhưng không kiểm chi tiết bất kỳ chức năng nào. Tất cả các trường hợp trong bộ kiểm thử hồi quy sẽ được thực thi mỗi khi một phiên bản vá lỗi của phần mềm được phát hành và điều này làm cho chúng trở nên lý tưởng cho tự động hóa.

Kiểm thử hồi quy được thực hiện khi phần mềm thay đổi, do sửa lỗi, chức năng mới.

### **Test Level (Mức độ kiểm thử)**

Tất cả các giai đoạn của quá trình phát triển phầm mềm đều trải qua quá trình kiểm thử phần mềm. Có 4 cấp độ kiểm thử phần mềm là:

* Kiểm thử đơn vị (Unit Testing)
* Kiểm thử tích hợp (Integration Testing)
* Kiểm thử hệ thống (System Testing)
* Kiểm thử chấp nhận (Acceptance Testing)

#### **Unit Testing( Kiểm thử đơn vị)**

##### **Định nghĩa**

Unit Testing là một mức kiểm thử phần mềm với mục đích để xác nhận từng unit của phần mềm được phát triển đúng như được thiết kế. Unit testing là mức test nhỏ nhất trong bất kỳ phần mềm nào. các hàm (Function), thủ tục (Procedure), lớp (Class), hoặc các phương thức (Method) đều có thể được xem là Unit. Nó thường có một hoặc vài đầu vào nhưng đầu ra là duy nhất.

##### **Khi nào thực hiện Unit Testing?**

Unit testing là mức kiểm thử đầu tiên trong các mức kiểm thử phần mềm. Nó được thực hiện trước khi Integration Testing. Kiểm thử “Hộp trắng”

##### **Ai thực hiện Unit Testing?**

Unit Testing thường do lập trình viên thực hiện. Unit test được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt quá trình phát triển phần mềm.

##### **Mục đích**

* Tăng sự đảm bảo khi có sự thay đổi mã
* Code dễ sử dụng, dễ hiểu, có thể tái sử dụng nhiều hơn
* Phát triển nhanh hơn
* Chi phí sửa lỗi thấp hơn so với các mức kiểm thử giai đoạn sau
* Debug dễ dàng

#### **Integration Testing( Kiểm thử tích hợp)**

##### **Định nghĩa**

Kiểm thử tích hợp là một mức của kiểm thử phần mềm kiểm tra một nhóm các module nhỏ liên quan đến nhau xem chúng có hoạt động đúng chức năng như trong thiết kế hay không. Theo ISTQB ( International Software Testing Qualifications Board):

* Kiểm thử tích hợp được thực hiện để phát hiện các lỗi về giao diện hoặc trong tương tác giữa các thành phần hoặc hệ thống tích hợp
* Kiểm thử tích hợp thành phần: kiểm tra sự tương tác giữa các thành phần với điều kiện các thành phần đã pass ở phần kiểm thử thành phần trước đó
* Kiểm thử tích hợp hệ thống: kiểm tra sự tương tác giữa các hệ thống con khác nhau và các hệ thống này đã pass ở lần kiểm thử trước đó

##### **Khi nào thực hiện Integration Testing?**

Kiểm thử tích hợp là mức thứ 2 trong các mức kiểm thử phần mềm. Nó được thực hiện sau Unit Testing và trước System testing. Kiểm thử mọi hộp (đen, trắng, xám)

##### **Ai thực hiện Integration Testing?**

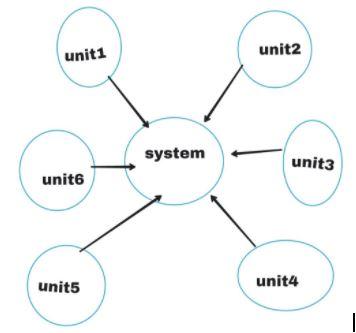
Kiểm thử integration có thể được thực hiện bởi developer, một test team chuyên biệt hay một nhóm chuyên developer/kiểm thử viên tích hợp bao gồm cả kiểm thử phi chức năng.

##### **Mục đích**

Kiểm tra sự tích hợp 1 nhóm các thành phần riêng lẻ có liên quan xem chúng có hoạt động đúng như mong đợi hay không.

##### **Một số phương pháp kiểm thử tích hợp**

###### Bigbang

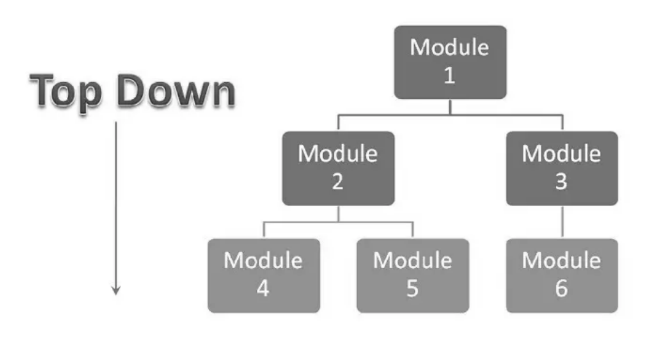


Hình 2: Phương pháp kiểm thử Bigbang

Đây là phương pháp test tích hợp mà tất cả hoặc hầu hết các unit được kết hợp với nhau và cùng được kiểm thử. Phương pháp này được thực hiện khi team kiểm thử nhận được toàn bộ phần mềm.

Như vậy big bang testing là kiểm tra sự tương tác giữa các unit , còn system test là sự tương tác của cả hệ thống.

###### **Top down**



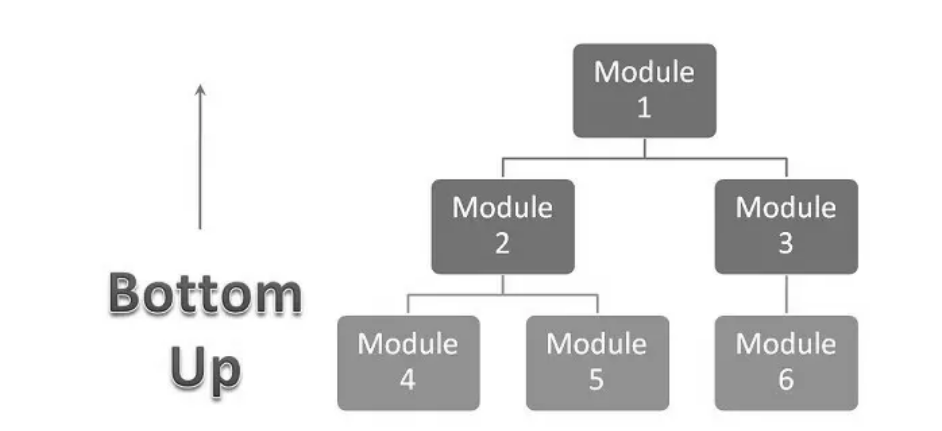
Hình 3: Phương pháp kiểm thử Top down

Đơn vị cao nhất được kiểm thử đầu tiền, đơn vị thấp hơn được kiểm thử sau đó một các tuần tự.

Vd: các chức năng chung, menu chính,… (ví dụ: bắt đầu từ GUI hoặc menu chính).

Modules quan trọng đang được thử nghiệm trên mức ưu tiên; lỗi trong thiết kế lớn có thể được tìm thấy và cố định đầu tiên.

###### **Bottom up**



Hình 4: Phương pháp kiểm thử Botton up

Đơn vị dưới cùng được kiểm thử đầu tiên, đơn vị cao hơn được kiểm thử tuần tự sau đó.

Vd: kiểm tra các module nhỏ hơn sau đó gộp vs module lớn hơn,…

Mỗi module ở mức thấp hơn được thử nghiệm với các module cao hơn cho đến khi tất cả các module đều được kiểm tra.

###### **Sanwich**

Sandwich là một cách tiếp cận để kiểm thử tích hợp, đó là sự kết hợp của các phương pháp Top Down và Bottom Up.

##### **Một số lưu ý trước khi thực hiện kiểm thử tích hợp**

* Đảm bảo rằng bạn có tài liệu thiết kế chi tiết phù hợp trong đó các tương tác giữa mỗi đơn vị được xác định rõ ràng.
* Đảm bảo rằng bạn có một hệ thống quản lý cấu hình phần mềm mạnh mẽ tại chỗ. Hoặc nếu không, bạn sẽ có một thời gian khó khăn theo dõi phiên bản phù hợp của từng đơn vị, đặc biệt là nếu số lượng đơn vị được tích hợp là rất lớn.
* Đảm bảo rằng mỗi đơn vị được kiểm thử đơn vị trước khi bạn bắt đầu Kiểm thử tích hợp.

#### **System Testing( Kiểm thử hệ thống)**

##### **Định nghĩa**

Kiểm thử hệ thống là một mức của kiểm thử phần mềm . Giai đoạn này sẽ hoàn thiện và hợp nhất phần mềm để kiểm thử. Theo ISTQB định nghĩa: quy trình của kiểm thử tích hợp hệ thống để xác nhận xem hệt hống phần mềm có đáp ứng đúng theo đặc tả yêu cầu.

##### **Khi nào thực hiện System Testing?**

Kiểm thử hệ thống là mức kiểm thử thứ 3 trong các mức kiểm thử phần mềm được thực hiện sau kiểm thử tích hợp và trước kiểm thử chấp nhận. Kiểm thử “hộp đen”

##### **Ai thực hiện System Testing?**

Thông thường, các tester thực hiện kiểm thử hệ thống.

##### **Mục đích**

Mục đích của giai đoạn này là để đánh giá sự hoạt động của hệ thống có đúng theo như tài liệu đặc tả.

#### **Acceptance Testing( Kiểm thử chấp nhận)**

##### **Định nghĩa**

Theo ISTQB định nghĩa: Kiểm thử chấp nhận chính thức liên quan đến yêu cầu và quy trình kinh doanh để xác định liệu hệ thống có đáp ứng tiêu chí chấp nhận hay không và cho phép người dùng, khách hàng hoặc tổ chức được ủy quyền khác xác định có chấp nhận hệ thống hay không.

##### **Khi nào thực hiện Acceptance Testing?**

Kiểm thử chấp nhận là mức thứ 4 được thực hiện sau khi hoàn thành kiểm thử hệ thống và trước khi đưa sản phẩm vào sử dụng chính thức.

##### **Ai thực hiện Acceptance Testing?**

Kiểm thử chấp nhận được chia thành 2 mức khác nhau

* Kiểm thử alpha: được thực hiện bởi những người trong tổ chức nhưng không tham gia phát triển phần mềm. Kiểm thử “hộp đen và trắng”
* Kiểm thử beta: được thực hiện bởi khách hàng/ người dùng cuối tại địa điểm của người dùng cuối. Kiểm thử “hộp đen”

##### **Mục đích**

Đảm bảo phần mềm đáp ứng đúng yêu cầu của khách hàng. Sản phẩm nhận được sự chấp nhận từ khách hàng/ người dùng cuối.

##### **Kết luận**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Các mức kiểm thử** | **Unit Testing** | **Integration Testing** | **System testing** | **Alpha Testing** | **Beta Testing** |
| Định nghĩa | Là mức kiểm thử nhỏ nhất, test từng module nhỏ trong hệ thống | Kiểm thử tích hợp một nhóm các module riêng lẻ có liên quan với nhau | Mức kiểm thử toàn bộ, hợp nhất tất cả các chức năng của phần mềm | Hoạt động kiểm thử chấp nhận nội bộ | Kiểm thử chấp nhận khi phát hành, bàn giao cho khách hàng. |
| Người thực hiện | Do chính dev viết module đó | Kiểm thử độc lập hoặc do lập trình viên | Tester | Người dùng/ nhóm test độc lập( không phải người phát triển phần mềm) thực hiện tại nơi sản xuất phần mềm | Thực hiện bởi khách hàng/ người dùng cuối tại địa điểm của họ |
| Thời gian thực hiện | \* Trong quá trình build module đó. \* Phải hoàn thành trước khi chuyển sang Integration Testing | Sau khi các module nhỏ liên quan đã được unit test | Sau Integration Testing và trước Acceptance Testing | Sau system Testing , và trước Beta Testing | Sau alpha testing |
| Mục đích | Đảm bảo mỗi module nhỏ của phần mềm được thực hiện đúng với thiêt kế | Kiểm tra chức năng, độ tin cậy và hiệu suất của hệ thống khi tích hợp | Đánh giá toàn bộ hệ thống hoạt động đúng theo đặc tả yêu cầu hay không | Đảm bảo phần mềm hoạt động đúng trước khi bàn giao cho khách hàng | Đảm bảo khách hàng xác nhận phần mềm hoạt động đúng như yêu cầu của họ |
| Ưu điểm | \* Tăng sự đảm bảo khi có sự thay đổi mã. \* Code dễ sử dụng, dễ hiểu, có thể tái sử dụng nhiều hơn. \* Phát triển nhanh hơn. \* Chi phí sửa lỗi thấp hơn so với các mức kiểm thử giai đoạn sau. \* Debug dễ dàng | \* Phát hiện lỗi ở đơn vị thấp của sản phẩm. \* Thuận tiện với các dự án nhỏ | Kiểm tra được tổng thể hoạt động của toàn hệ thống trước khi phát hành | Các vấn đề xuất hiện có thể được giải quyết ngay bởi bên sản xuất | Chỉ đòi hỏi thực hiện trong thời gian ngắn |
| Nhược điểm | Dev ít kinh nghiệm sẽ mất khá nhiều thời gian để fix bug | \* Các lỗ hổng về thiết kế ở cấp cao hơn sẽ được phát hiện chỉ ở cuối vòng đời. \* Có thể bỏ qua các bug giao diện nhỏ trong quá trình tìm bug | Chỉ có thể thực hiện khi tất cả các module riêng lẻ đã hoàn tất và không có lỗi lớn | Có thể kéo dài lâu hơn | \* Giới hạn số lượng người dùng. \* Chi phí sửa lỗi cao hơn nếu có |
| Kỹ thuật kiểm thử | Kiểm thử hộp trắng | Kiểm thử hộp trắng, đen và xám | Kiểm thử hộp đen | Kiểm thử hộp trắng và đen | Kiểm thử hộp đen |

Bảng 2: So sánh các mức độ kiểm thử

### **Test Method (Phương pháp kiểm thử)**

#### **BACK BOX TEST**

##### **Định nghĩa**

Kiểm tra hộp đen (Black box testing) là một phương pháp kiểm thử phần mềm mà việc kiểm tra các chức năng của một ứng dụng không cần quan tâm vào cấu trúc nội bộ hoặc hoạt động của nó.

##### **Đối tượng được kiểm thử**

Là thành phần phần mền (TPPM) có thể là 1 hàm chức năng, 1 modul chức năng, 1 phân hệ chức năng...

##### **Phương pháp thử nghiệm**

Dựa vào chức năng Kiểm thử hộp đen (Black box test) có thể được áp dụng hầu như đến mọi cấp độ của kiểm thử phần mềm:

* Kiểm thử đơn vị (Unit test)
* Kiểm thử tích hợp (Intergration test)
* Kiểm thử hệ thống (System test)
* Kiểm thử chấp nhận (Acceptance test).

Tuy nhiên, Black box test được sử dụng thích hợp nhất trong kiểm thử hệ thống (System test) và Kiểm thử chấp nhận (Acceptance test)

##### **Đặc điểm**

* Là chiến lược kiểm thử TPPM dựa vào thông tin duy nhất là các đặc tả về yêu cầu chức năng của TPPM tương ứng.
* Người kiểm thử không cần thiết phải có kiến thức về việc mã hoá, cấu trúc bên trong của TPPM, cũng như không yêu cầu phải biết lâp trình phần mềm.
* Việc kiểm thử được tiến hành dựa vào việc kiểm thử TPPM làm được gì, có phù hợp với yêu cầu của người dùng hay không. Các tester nhập số liệu vào phần mềm và chỉ cần xem kết quả của phần mềm và các mục tiêu kiểm tra.
* Mức test này thường yêu cầu các tester phải viết test case đầy đủ trước khi test; khi test, đơn giản chỉ cần thực hiện theo các bước mô tả trong test case thao tác và nhập data vào, sau đó xem kết quả trả về hoặc hành vi của phần mềm, rồi so sánh với kết quả mong đọi được viết trong testcase

##### **Tạo test case và Thực hiện test case**

* Khi viết test case: Dựa vào yêu cầu và giao diện bên ngoài của chương trình (Không can thiệp vào bên trong code của chương trình)
* Khi thực hiện test: Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code)

#### **WHITE BOX TEST**

##### **Định nghĩa**

Kiểm thử hộp trắng (While box test) là phương pháp thử nghiệm phần mềm, trong đó các thiết kế, cấu trúc giải thuật bên trong, và việc thực hiện các công việc đều được biết đến

##### **Đối tượng kiểm thử**

Là 1 thành phần của phần mềm (1 chức năng, 1 module chức năng, 1 phân hệ chức năng....)

##### **Phương pháp thử nghiệm**

Dựa vào thuật giải Kiểm thử hộp trắng dựa vào thuật giải cụ thể, vào cấu trúc dữ liệu bên trong của đơn vị phần mềm cần kiểm thử để xác định đơn vị phần mềm đó có thực hiện đúng không.

* Với những TPPM quá lớn sẽ tốn rất nhiều thời gian và công sức để kiểm thử nếu như dùng kiểm thử tích hợp (Integration test) hay kiểm thử chức năng (Functional test)).
* Kỹ thuật white box test thích hợp dùng để kiểm thử đơn vị (Unit test)

##### **Đặc điểm**

* Là chiến lược kiểm thử TPPM dựa vào giải thuật, cấu trúc bên trong chức năng của TPPM tương ứng.
* Người kiểm thử phải có kiến thức nhất định về việc mã hoá, cấu trúc bên trong của chức năng, biết lâp trình phần mềm.
* Việc kiểm thử được tiến hành dựa vào việc kiểm xem giải thuật, mã lệnh đã làm có đúng không.
* Mức test này thường yêu cầu các tester phải viết test case đầy đủ các nhánh trong code; khi test, sẽ set điều kiện và data để chạy vào đủ tất cả các nhánh trong giải thuật, đảm bảo thực hiện đầy đủ.

##### **Tạo testcase và thực hiện test**

* Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)
* Khi thực hiện test: Thực thi test trong code (không cần thực thi chương trình, vì thực hiện test white box sẽ sử dụng framework nào đó hỗ trợ (Ví dụ như test kiểu debug)
* Trong kiểm tra này, đòi hỏi người tester phải có kiến thức và kỹ năng nhất định về ngôn ngữ lập trình được dùng, hiểu thuật giải trong thành phần phần mềm, để có thể hiểu được chi tiết về đoạn code cần kiểm thử .

#### **GRAY BOX TEST**

Ngoài 2 kỹ thuật đã được nhắc đến: Black box test và white box test, thì có 1 kỹthuật, Gray box test là sự kết hợp giữa black box test và white box test.

##### **Định nghĩa**

Gray Box Testing là một phương pháp kiểm thử phần mềm được kết hợp giữa Phương pháp Kiểm thử Black Box (hộp đen) và White Box (hộp trắng). Trong Kiểm thử Hộp xám, cấu trúc bên trong sản phẩm chỉ được biết một phần

##### **Phương pháp thử nghiệm**

Dựa vào giải thuật và chức năng

Gray box test có thể được sử dụng ở nhiều mức kiểm thử khác nhau. Tuy nhiên, chủ yếu được ứng dụng trong Kiểm thử tích hợp (Intergration test)

##### **Tạo testcase và thực hiện test**

* Khi viết test case: Dựa vào yêu cầu và nội dung Source Code (can thiệp vào bên trong Code của chương trình)
* Khi thực hiện test: Thực hiện trên giao diện của chương trình (yêu cầu chương trình phải chạy được mới test được, không can thiệp vào code)

### **Kĩ thuật kiểm thử hộp đen**

#### **Phân vùng tương đương (equivalence class)**

Ý tưởng của kỹ thuật này là cố gắng phân các testcase ra thành nhiều nhóm khác nhau: các testcase trong mỗi nhóm xác định TPPM thực hiện cùng 1 hành vi.

Mỗi nhóm testcase thỏa mãn tiêu chuẩn trên được gọi là 1 lớp tương đương, ta chỉ cần xác định 1 testcase đại diện cho nhóm và dùng testcase này để kiểm thử TPPM. Như vậy ta đã giảm rất nhiều testcase cần định nghĩa và kiểm thử, nhưng chất lượng kiểm thử không bị giảm sút bao nhiêu so với vét cạn.

Điều này là dựa vào kỳ vọng:

* Nếu 1 testcase trong lớp tương đương nào đó gây lỗi TPPM thì các testcase trong lớp này cũng sẽ gây lỗi như vậy.
* Nếu 1 testcase trong lớp tương đương nào đó không gây lỗi TPPM thì các testcase trong lớp này cũng sẽ không gây lỗi.
* Với các giá trị không hợp lệ: Ta nên tạo 1 lớp tương đương đại diện các testcase chứa các giá trị không hợp lệ theo đặc tả để xem TPPM phản ứng như nào với những trường hợp này

#### **Kỹ thuật phân tích các giá trị ở biên**

Khi tạo testcase, ta chỉ dùng Kỹ thuật phân lớp tương đương thì hẳn là chưa đủ. Kinh nghiệm cho thấy rằng lỗi thường nằm ở biên (đầu hay cuối) của 1 khoảng liên tục nào đó (lớp tương đương).

Do đó với Kỹ thuật phân tích giá trị biên tập trung tạo các testcase ứng với những giá trị ở biên này.

Nên thông thường là có sự kết hợp cả 2 kỹ thuật: Phân lớp tương đương và Phân tích giá trị biên để viết các testcase.

Ý tưởng của kỹ thuật là chỉ định nghĩa các testcase ứng với các giá trị ngay trên biên hay lân cận biên của từng lớp tương đương. Do đó kỹ thuật này chỉ thích hợp với các lớp tương đương xác định bởi các giá trị liên tục (số nguyên, số thực), chứ nó không thích hợp với lớp tương đương được xác định bởi các giá trị liệt kê mà không có mối quan hệ lẫn nhau.

Quy trình cụ thể để thực hiện kiểm thử dựa trên các giá trị ở biên:

* Nhận dạng các lớp tương đương dựa trên đặc tả về yêu cầu chức năng của TPPM.
* Nhận dạng 2 biên của mỗi lớp tương đương. Tạo các testcase cho mỗi biên của mỗi lớp tương đương :
* 1 testcase tại giá trị biên.
* 1 testcase ngay dưới biên.
* 1 testcase ngay trên biên.

Ý nghĩa ngay trên và ngay dưới biên phụ thuộc vào đơn vị đo lường cụ thể : Số nguyên , số thập phân...

#### **Kỹ thuật dùng bảng quyết định (decision table)**

Bảng quyết định là 1 công cụ rất hữu ích để đặc tả các yêu cầu phần mềm hoặc để đặc tả bảng thiết kế hệ thống phần mềm. Nó miêu tả các qui tắc nghiệp vụ phức tạp mà phần mềm phải thực hiện dưới dạng dễ đọc và dễ kiểm soát :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Rule 1** | **Rule 2** | **...** | **Rule p** |
| **Conditions** |  |  |  |  |
| Conditions-1 |  |  |  |  |
| ... |  |  |  |  |
| Conditions-m |  |  |  |  |
| **Actions** |  |  |  |  |
| Actions-1 |  |  |  |  |
| ... |  |  |  |  |
| Actions-n |  |  |  |  |

Trong đó:

* Condition-1 tới Condition-m miêu tả m điều kiện dữ liệu nhập khác nhau có thể có.
* Action-1 tới Action-n miêu tả n hoạt động khác nhau mà hệ thống có thể thực hiện phụ thuộc vào tổ hợp điều kiện dữ liệu nhập nào.
* Mỗi cột miêu tả 1 luật cụ thể : tổ hợp điều kiện nhập cụ thể và các hoạt động cụ thể cần thực hiện. Lưu ý các hoạt động cần thực hiện không phụ thuộc vào thứ tự các điều kiện nhập, nó chỉ phụ thuộc vào giá trị các điều kiện nhập. Tương tự, các hoạt động cần thực hiện không phụ thuộc vào trạng thái hiện hành của TPPM, chúng cũng không phụ thuộc vào các điều kiện nhập đã có trước đó.

#### **Sơ đồ chuyển đổi trạng thái (state transition testing)**

Sơ đồ chuyển đổi trạng thái là một kỹ thuật được sử dụng để kiểm tra các trạng thái khác nhau của hệ thống. Trạng thái của hệ thống thay đổi tùy theo điều kiện hoặc sự kiện. Các sự kiện kích hoạt các trạng thái trở thành kịch bản và người kiểm tra cần kiểm tra chúng.

Một sơ đồ chuyển trạng thái cung cấp một cái nhìn rõ ràng về các thay đổi trạng thái nhưng chỉ có hiệu quả đối với các ứng dụng đơn giản. Các dự án phức tạp hơn có thể dẫn đến các sơ đồ chuyển đổi trạng thái phức tạp hơn do đó làm cho nó kém hiệu quả hơn.

## Tổng quan về nghề nghiệp

### **Vị trí làm việc**

Nhân viên Tester là người kiểm thử phần mềm, chịu trách nhiệm phát triển chất lượng và triển khai phần mềm. Họ tham gia vào việc thực hiện các thử nghiệm tự động và thủ công để đảm bảo phần mềm do lập trình viên viết phù hợp với mục đích sử dụng. Một số nhiệm vụ bao gồm phân tích phần mềm và hệ thống, giảm thiểu rủi ro và ngăn ngừa sự cố phần mềm.

### **Mô tả công việc của nhân viên Tester**

Là một nhân viên Tester, bạn sẽ là một trong những người đầu tiên tiếp xúc và kiểm thử phần mềm, thường xuyên được làm việc trong các dự án mới, thú vị. Đặc biệt, với vị trí Tester, cho dù chưa có kinh nghiệm bạn vẫn có thể ứng tuyển. Sự tập trung và khả năng làm việc độc lập là một trong những yếu tố quan trọng hàng đầu mà nhà tuyển dụng tìm kiếm.

#### **Công việc của nhân viên Tester**

* Là Tester cần phải hiểu rõ được các sản phẩm cần phải kiểm tra.
* Cần nắm rõ cạc lập kế hoạch các chiến lược thử nghiệm để tìm ra được những vấn đề cần giải quyết và thực hiện các thử nghiệm.
* Thực hiện phân tích rõ ràng các ưu điểm, nhược điểm hoặc giải quyết dễ dàng hơn các rủi ro liên quan đến từng thành phần cũng như giao diện của sản phẩm.
* Cần hiểu rõ về code để có thể dễ dàng check cũng như kiểm tra lại chúng.
* Cần phải có các kỹ năng làm việc với các tập lệnh cũng như công cụ tự động hóa.
* Cần thường xuyên cập nhật cho mọi khía cạnh kỹ thuật của các cơ sở hạ tầng sản phẩm.
* Chú ý phân tích, ghi nhận được các vấn đề và có trách nhiệm cung cấp các phản hồi thích hợp nhất.

#### **Yêu cầu trình độ và kỹ năng với nhân viên Tester**

* Bằng cử nhân/cao đẳng các chuyên ngành khoa học máy tính, công nghệ thông tin hoặc liên quan.
* Kiến thức về thiết kế, kiểm thử phần mềm và thành thạo các phương pháp thử nghiệm.
* Thành thạo các kỹ thuật kiểm thử phần mềm cũng như đánh giá khả năng tương thích với các chương trình phần mềm khác nhau.
* Kiến thức về lập trình.
* Giao tiếp tốt và kỹ năng tư duy phản biện.
* Kỹ năng tổ chức tốt và tư duy định hướng chi tiết.

### **Mức lương**

Theo khảo sát thực tế, mức lương của nhân viên Tester có thể dao động trong khoảng **từ 4 đến 34 triệu/tháng**, tuỳ thuộc vào hình thức làm việc, kinh nghiệm, trình độ và hiệu suất.

# CHƯƠNG 2. PHÂN TÍCH VÀ KIỂM THỬ

## 2.1 Mô tả bài toán

## 2.2 Phân tích hệ thống

Sử dụng phương pháp phân tích hướng đối tượng (use case, mô tả chi tiết use case) để mô tả về hệ thống cần kiểm thử

Mô hình usecase, workflow.

# CHƯƠNG 3. KẾT QUẢ

## Kết quả thực hiện thiết kế testcase

## Kết quả test hệ thống

Trình bày kết quả của thiết kế test case và thực hiện test

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

# TÀI LIỆU THAM KHẢO

# PHỤ LỤC (nếu có)