TRƯỜNG ĐẠI HỌC DUY TÂN

KHOA CỒNG NGHỆ THÔNG TIN

ĐỒ ÁN CHUYÊN NGÀNH

TÌM HIỂU KIỂM THỬ PHẦN MỀM

Chuyên ngành : CÔNG NGHỆ PHẦN MỀM

GVHD : NGUYỄN THÀNH TRUNG

SVTH : DƯƠNG THỊ NGỌC TRANG

MSSV :142124568

***MỤC LỤC***

Lời mở đầu……………….

Chương 1: Tổng quan về kiểm thử phần mềm

1.1 Các khái niệm

1.1.1 Các định nghĩa

1.1.1.1 Định nghĩa kiểm thử phần mềm

1.1.1.2Các thuật ngữ

1.1.2 Quá trình kiểm thử

1.1.3 Phân loại kiểm thử

1.1.4 Sự tương quan giữa các công đoạn xây dựng phần mềm và loại kiểm thử

1.1.4 Sơ lược về các kỹ thuật và công đoạn kiểm thử.

1.2 Các loại kiểm thử phần mềm

* + 1. Các loại kiểm thử hay dùng
       1. Kiểm thử đơn vị
       2. Kiểm thử tích hợp
       3. Kiểm thử hệ thống
       4. Kiểm thử chấp nhận
    2. Một số loại kiểm thử khác
       1. Kiểm thử cấu hình
       2. Kiểm thử khả năng tương thích
       3. Kiểm thử Foreign-language
       4. Kiểm thử khả năng tiện dụng
       5. Kiểm thử tài liệu
       6. Kiểm thử khả năng bảo mật tài liệu
  1. Các phương pháp kiểm thử
     1. Phương pháp kiểm thử tĩnh
     2. Phương pháp kiểm thử động
     3. Phương pháp kiểm thử hộp đen
     4. Phương pháp kiểm thử hộp trắng
     5. Phương pháp kiểm thử hộp xám
  2. Quản lý chất lượng phần mềm
     1. Chất lượng là gì?
     2. Chất lượng sản phẩm và chất lượng qui trình.
     3. Đảm bảo chất lượng và các chuẩn chất lượng.
     4. Các hoạt động quản lý chất lượng
        1. Lập kế hoạch chất lượng
        2. Kiểm soát chất lượng.
        3. Kiểm định chất lượng

Chương 2 : Hoạt động kiểm thử phần mềm

* 1. Tổ chức
     1. Vai trò của nhóm tester độc lập.
     2. Vai trò của test thành viên
  2. Quy trình và hoạt động kiểm thử phần mềm
     1. Tìm hiểu các yêu cầu
     2. Tạo các Test Case Và Tes procedures (Create Test)
     3. Thực thi Test (Excute Tests) gồm 4 giai doạn
        1. Giai đoạn Unit Test
        2. Giai đoạn Test các chức năng (Functional Test)
        3. Giai đoạn Test tích hợp
        4. Giai đoạn Test hiệu suất.
     4. Xác định lỗi (Identify Errors)
     5. Giải quyết lại các lỗi (Resovle Errors)
     6. Thực hiện Test lại(Execute Retests)
     7. Báo cáo tình trạng
  3. Các biểu mẫu của liên quan

Chương 4: Giới thiệu Các Công cụ Kiểm thử tự động

* 1. JUNIT/NUNIT
  2. QTP
  3. Áp dụng

Chương 5: Một số vấn đề cần quan tâm khi kiểm thử phần mềm

5.1. Kiểm thử chừng nào là đủ

5.2. Những đặc tính cần có của kiểm thử viên

5.3. Hoạt động KTPM ở Viêt Nam

5.4. Một số mô hình kiểm thử : TMM, áp dụng CMMi,...

LỜI MỞ ĐẦU

## Lý do chọn đề tài:

Với sự phát triển như vũ bảo của công nghệ thông tin nói chung và công nghệ phần mềm nói riêng ,việc phát triển phần mềm ngày càng được hổ trở bởi nhiều công cụ tiên tiến giúp cho việc xây dựng phần mềm đỡ mệt nhọc và hiểu quả hơn.Tuy nhiên ,vì độ phức tạp của phân mềm và những giới hạn về thời gian và chi phí ,cho dù các hoạt động đảm bảo chất lượng phần mềm nói chung và kiểm thử nói riêng ngày càng chặt chẽ và khoa học ,vẫn không đảm bảo được rằng các sản phẩm phần mềm đang được ứng dụng không có lỗi .Lỗi vẫn luôn tiềm ẩn trong mọi sản phẩm phần mềm và cũng có thể gây những thiệt hại khôn lường.

Kiểm thử phần mềm là một quá trình liên tục ,xuyên suốt mọi giai đoạn phát triển phần mềm để đam bào rằng phần mềm thỏa mãn các yêu cầu thiết kế và các yêu cầu đó đáp ứng các nhu cầu cuả người dùng nên việc kiểm thử phần mềm đã trở thành qui trình bắt buộc trong các dự án phát triển phần mềm trên thế giới.Kiểm thử phần mềm là một hoạt động rất tốn kém ,mất thời gian ,và khó phát hiện được hết lỗi .Chính vì vậy ,việc kiểm thử phần mềm đòi hỏi phải có chiến lược phù hợp ,một kế hoạch hợp lý và viêc thực hiện được quản lý chặt chẽ.

Nhưng ở nước ta,trong thời gian qua việc kiểm thử được xem nhẹ ,với các công cụ lập trình hiện đại ,người ta cảm tính cho rằng không kiểm thử cũng không sao ,nên không có sự quan tâm nguyên cứu .Những năm gần đây tuy đã bắt đầu có những quan tâm hơn đến vấn đề kiểm thử nhưng chưa đúng mức .

Rất nhiều các giáo sư, giảng viên đã từng than phiền rằng: “ Sinh viên của chúng ta tốt nghiệp và đi làm mà không có được những kiến thực thực tế cần thiết về cách để kiểm thử một chương trình. Hơn nữa, chúng ta hiếm khi có được những lời khuyên bổ ích để cung cấp trong các khóa học mở đầu về cách một sinh viên nên làm về kiểm thử và gỡ lỗi các bài tập của họ”.

Chính vì tầm quan trọng của kiểm thử phần mềm là rất quan trọng và nó có ảnh hưởng đến sự thành công của một phần mềm ,nên đã chọn để nguyên cứu và tìm hiểu về chủ đề này.

## Mục tiêu và nhiệm vụ nguyên cứu

* Cách nhìn tổng quan về các kỹ thuật kiểm thử .
* Tổ chức hoạt động kiểm thử phần mềm
* Dùng công cụ để kiểm thử một ứng dụng cụ thể.

## Đối tượng và phạm vi nguyên cứu

* Quy trình và bản chất của hoạt động kiểm thử.
* Các chiến lược kiểm thử .
* Đặc tả thiết kế kiểm thử.

## Phương pháp nguyên cứu

* Nguyên cứu ,tìm hiểu ,các kỹ thuật ,các chiến lược và các hoạt động kiểm thử phần mềm.
* Sử dụng các phương pháp kiểm thử đã nguyên cứu ,thiết kế bộ test cho chương trình cụ thể .Đưa ra tài liệu kế hoạch kiểm thử và đặc tả kiểm thử ,xây dựng chương trình thực thi kiểm thử.

## Dự kiến kết quả

* Thiết kế một quy trình kiểm thử phần mềm
* Tạo các tài liệu kiểm thử(đặc tả trường hợp kiểm thử và kết quả kiểm thử )
* Xây dựng chương trình kiểm thử phần mềm.

.

# Chương I: TỔNG QUAN VỀ KIỂM THỬ VÀ BẢO ĐẢM CHẤT LƯỢNG PHẦN MỀM

## 1.1 Các khái niệm

### 1.1.1Các định nghĩa

#### 1.1.1.1 Kiểm thử phần mềm là gì?

Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định ,quan sát và ghi lại các kết quả ,và đánh giá một khía cạnh nào đó của hệ thống hay thành phần đó. (Theo *Bảng chú giải thuật ngữ chuẩn IEEE của Thuật ngữ kỹ nghệ phần mềm- IEEE Standard Glossary of Software Engineering Terminology*).

Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tim lỗi(Theo *“The Art of Software Testing” – Nghệ thuật kiểm thử phần mềm*).

Kiểm thử phần mềm là hoạt động khảo sát thực tiễn sản phẩm hay dịch vụ phần mềm trong đúng môi trường chúng dự định sẽ được triển khai nhằm cung cấp cho người có lợi ích liên quan những thông tin về chất lượng sản phẩm hay phần mềm ấy .Mục đích của kiểm thử phần mềm là tìm ra lỗi hay khiểm khuyết phần mềm nhằm đảm bảo hiệu quả hoạt động tối ưu của phần mềm trong nhiều ngành khác nhau. (Theo *Bách khoa toàn thư mở Wikipedia*).

Có thể hiêu một cách dễ hiểu như sau : Kiểm thử phần mềm là một tiến trình hay tập hợp các tiến trình được thiết kế để đảm bảo mã hóa máy tính thực hiện theo cái gì mà chúng ta đã được thiết kế để làm,và không thực hiện bất cứ thứ gì không mong muốn .Đây là một pha quan trọng trong quá trình phát triển hệ thống ,giúp cho người xây dựng hhej thống và khách hàng thấy được hệ thống mới đã đáp ứng yêu cầu đạt ra hay chưa?

#### 1.1.1.2 Các thuật ngữ

##### 1.1.1.2.1 Lỗi là gì:

* Phần mềm không thực hiện những chức năng có trong đặc tả.
* Phần mềm thực hiện những chức năng không có trong đặc tả.
* Phần mềm có những chức năng mà đặc tả yêu cầu không nên có
* Phần mềm không có những chức năng mặc dù trong đặc tả không có nhưng nên có trong thực tế.
* Phần mềm khó hiểu ,khó sử dụng,tốc độ chậm.

##### 1.1.1.2.2 Các thuật ngữ liên quan đến lỗi:

* Lỗi (Error):là lỗi do con người gây ra.
* Sai sót (fault) :xuất hiện trong phần mềm như là kết quả của error .Có thể phân loại như sau:
* Sai sót đưa ra dư thừa-chúng ta đưa ra một vài thứ không chính xác vào mô tả yêu cầu phần mềm.
* Sai sót do bỏ sót –người thiết kế có thể gây ra sai sót do bỏ sót ,kết quả là thiếu một số phần đáng ra phải co trong mô tả yêu cầu phần mềm.
* Hỏng hóc (Failure): xảy ra khi sai sót được thực thi (khi thực thi chương trình tại các nơi bị sai sẽ xảy ra trạng thái hỏng hóc).
* Kết quả không mong đợi ,hậu quả(Incident): Là những kết quả do sai sót đem đến.Hậu quả ;à các triệu chứng liên kết với một hỏng hóc và báo hiệu cho người dùng biết được sự xuất hiện của hỏng hóc.
* Trường hợp thử (Test case) : Trường hợp thử được liên kết tương ứng với hoạt động của chương trình .Một trường hợp thử bao gồm một tập các giá trị đầu vào và một danh sách kết quả đầu ra mong muốn.
* Dữ liệu kiểm thử (Test Data): Dữ liệu đầu vào cần cung cấp cho phần mềm trong khi thực thi.
* Kịch bản kiểm thử (Test scenario):Các bước thực hiện khi kiểm thử.
* Thẩm tra(Verification): Là tiến trình nhằm xác định đầu ra của một công đoạn trong việc phát triển phần mềm phù hợp với công đoạn trước.
* Xác nhận(Validation): là tiến trình nhằm chỉ ra toàn hệ thống đã phát triển xong phù hợp với tài liệu mô tả yêu cầu.
* So sánh giữa thẩm tra(Verification) và xác nhận(Valication):
* Verification: thẩm tra quan tâm đến việc ngăn chặn lỗi giữa các công đoạn.
* Valication: Xác nhận quan tâm đến sản phẩm cuối cùng không có lỗi.
* Kiểm thử tĩnh (Static Testing)
* Quá trình kiểm tra và xem xét phần mềm mà không chạy chương trình.
* Phương pháp tiếp cận chung: Xem xét (Review)
* Kiểm thử động (Dynamic testing)
* Quá trình thử nghiệm phần mềm bằng cách chạy chương trình hoặc sử dụng phần mềm.
* Phương pháp tiếp cận: Kiểm thử chức năng và kiểm thử cấu trúc.
* Kiểm thử chức năng (Functional Testing)

Kiểm thử được tiến hành để đánh giá phần mềm có đáp ứng được các yêu cầu chức năng thành phần hoặc hệ thống trong đặc tả không.

* Kiểm thử hộp đen (Black – box testing)

Quá trình thử nghiệm bỏ qua cơ chế hoạt động bên trong của thành phần hoặc hệ thống phần mềm, chỉ quan tâm đến giá trị đầu ra từ các giá trị đầu vào.

**Mối quan hệ**

Kiểm thử hộp đen = Kiểm thử chức năng + Kiểm thử phi chức năng.

* Kiểm thử cấu trúc (Structural Testing)

Kiểm thử nhằm tìm ra lỗi dựa trên cấu trúc bên trong chương trình hoặc tính logic của chương trình.

* Kiểm thử hộp trắng (White– box testing)

Được gọi là kiểm thử hộp kính (glass – box hoặc clear – box ).Kiểm thử trên cơ chế hoạt động bên trong của phần mềm hoặc thành phần phần mềm.

**Mối quan hệ**

Kiểm thử hộp trắng = Kiểm thử cấu trúc

* Kiểm thử (Testing)

Quá trình vận hành hệ thống theo đặc tả qui định, quan sát và ghi lại kết quả, đưa ra các đánh giá về hệ thống đó. Mục đích của kiểm thử là tìm lỗi, khiếm khuyết của phần mềm.

* SQA (Software Quality Assurance)

Các hoạt động nhằm đánh giá quy trình phát triển phần mềm. Mục đích của QA là đánh giá và cải tiến quy trình.

**Mối quan hệ**

* Đối tượng làm việc của Testing là phần mềm.
* Đối tượng làm việc của QA là quy trình phát triển phần mềm.

### 1.1.2 Phân loại kiểm thử

Có 2 mức phân loại :

* Phân biệt theo mức độ chi tiết của các bộ phận hợp thành phần mềm .
* Mức kiểm tra đơn vị.(Unit).
* Mức kiểm tra hệ thống(System).
* Mức kiểm tra tích hợp(Integration).
* Dựa trên phương pháp thử nghiệm(thường dùng ở mức kiểm tra đơn vị):
* Kiểm nghiệm hộp đen (Black box testing) dùng để kiểm tra chức năng.
* Kiểm nghiệm hộp trắng (White box testing)dùng để kiểm tra cấu trúc.

### 1.1.3 Sự tương quan giữa các công đoạn xây dựng phần mềm và các loại kiểm thử :Mô hình chữ V:

Mô hình này nhằm giải thích sự tương quan giữa các công đoạn xây dựng phần mềm và các loại kiểm nghiệm. Ở mỗi công đoạn xây dựng phần mềm sẽ tương ứng với một loại kiểm nghiệm và cần có một hồ sơ kiểm nghiệm tương ứng được thành lập để phục vụ cho việc kiểm nghiệm.

Ví dụ:

- **Công đoạn:** Yêu cầu phần mềm(*requiements*); **Loại kiểm nghiệm:** Kiểm nghiệm chấp nhận (*acceptance test*); **Hồ sơ:** hồ sơ kiểm nghiệm chấp nhận (*acceptance test spec*).

- **Công đoạn:** Mô tả chi tiết phần mềm (*specification*); **Loại kiểm nghiệm:** Kiểm nghiệm hệ thống(*system test*); **Hồ sơ:** hồ sơ kiểm nghiệm hệ thống (*system test spec*).

- **Công đoạn:** Hồ sơ kiến trúc (*architecture spec*); **Loại kiểm nghiệm:** Kiểm nghiệm tích hợp (*integration test*); **Hồ sơ:** hồ sơ kiểm nghiệm tích hợp (*integration test spec*).

- **Công đoạn:** Thiết kế chi tiết (*detailed design*); **Loại kiểm nghiệm:** Kiểm nghiệm khối (*module test*); **Hồ sơ:** hồ sơ kiểm nghiệm khối (*module test spec*).

- **Công đoạn:** Viết mã (*implementation code*); **Loại kiểm nghiệm:** Kiểm nghiệm

đơn vị (*unit test*); **Hồ sơ:** hồ sơ kiểm nghiệm đơn vị (*unit test spec*).

acceptance test spec

requiements

acceptance test

system test spec specification

system test

architecture spec Sai sót

integration test spec

integration test

module test spec

detailed design

module test

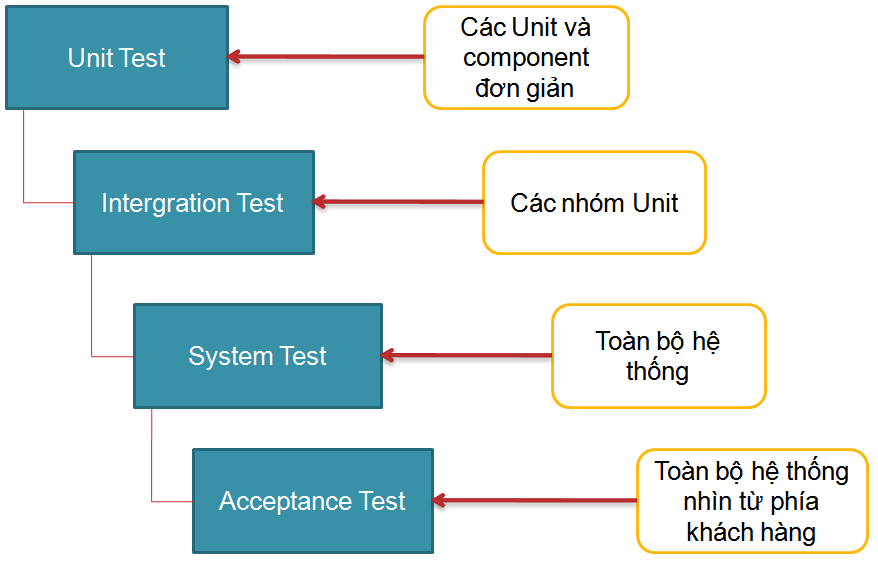
implementation code

unit test spec

unit test

## 1.2 Các loại kiểm thử phần mềm

### 1.2.1 Các loại kiểm thử hay dùng



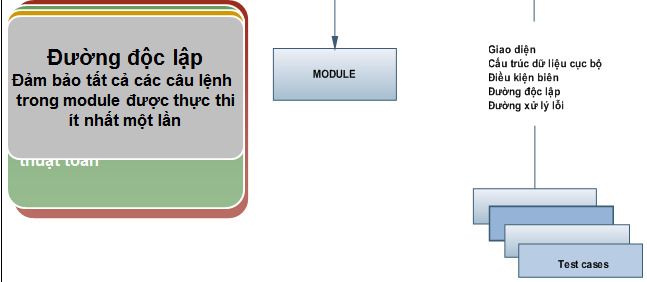
Hình 2.1.3 Cấp độ kiểm thử phần mềm

#### 1.2.1.1 Kiểm thử đơn vị

Một đơn vị là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm thử được. Ví dụ, các hàm *(Function)*, thủ tục *(Procedure),* lớp *(Class)* hay phương thức *(Method)* đều có thể được xem là Unit.

Vì Unit được chọn để kiểm tra thường có kích thước nhỏ và chức năng hoạt động đơn giản, chúng ta không khó khăn gì trong việc tổ chức kiểm thử, ghi nhận và phân tích kết quả kiểm thử. Nếu phát hiện lỗi, việc xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một đơn thể Unit đang kiểm tra. Một nguyên lý đúc kết từ thực tiễn: thời gian tốn cho Unit Test sẽ được đền bù bằng việc tiết kiệm rất nhiều thời gian và chi phí cho việc kiểm thử và sửa lỗi ở các mức kiểm thử sau đó.

Unit Test thường do lập trình viên thực hiện. Công đoạn này cần được thực hiện càng sớm càng tốt trong giai đoạn viết code và xuyên suốt chu kỳ phát triển phần mềm. Thông thường, Unit Test đòi hỏi kiểm thử viên có kiến thức về thiết kế và code của chương trình. Mục đích của Unit Test là bảo đảm thông tin được xử lý và xuất (khỏi Unit) là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Điều này thường đòi hỏi tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một Unit. Ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then ... else là một nhánh. Thực tế việc chọn lựa các nhánh để đơn giản hóa việc kiểm thử và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa.



Hình 2.2.1 Các xem xét của Unit test

* Giao diện module:được kiểm thử để đảm bảo thông tin vào ,ra hợp lệ của đơn vị chương trình .Thường gồm một số kiểm thử cần thiêt:
* Số tham số đầu vào có bằng số đối số không?
* Các thuộc tính của tham số và đối số có phù hợp không?
* Số đối số truyền vào cho module được gọi có phù hợp với thuộc tính của tham số?
* Khai báo biến toàn cục nhất quán trong các module?
* Các ràng buộc phù hợp với các tham số?
* Các đối số được truyền vào đúng thứ tự?

Khi module thực hiện vào ra ,cần thực hiện các kiểm thử giao diện bố sung:

* Các thuộc tính tập tin có đúng không?
* Các lệnh đóng/ mở tập tin có đúng không?
* Các đặc tả hình thức phù hợp với lệnh vào/ra?
* Kích thước vùng đệm phù hợp với kích thuocs bản ghi?
* Các tập tin được mở truocs khi sử dụng?
* Xử lý điều kiện kết thúc tập tin?
* Xử lý lỗi vào ra?
* Cấu trúc dữ liệu cục bộ:là nguồn lỗi phổ biến ,được kiểm tra



Hình 2.2.2 Các thủ tục của Unit Test

Cùng với các mục kiểm thử khác, Unit Test cũng đòi hỏi phải chuẩn bị trước các ca kiểm thử (*Test case*) hoặc kịch bản kiểm thử (*Test script*), trong đó chỉ định rõ dữ liệu đầu vào, các bước thực hiện và dữ liệu đầu ra mong muốn. Các Test case và Test script này nên được giữ lại để tái sử dụng.



#### 1.2.1.2 Kiểm thử tích hợp

Integration test kết hợp các thành phần của một ứng dụng và kiểm thử như một ứng dụng đã hoàn thành. Trong khi Unit Test kiểm tra các thành phần và Unit riêng lẻ thì Intgration Test kết hợp chúng lại với nhau và kiểm tra sự giao tiếp giữa chúng.

Hai mục tiêu chính của Integration Test:

* Phát hiện lỗi giao tiếp xảy ra giữa các Unit.
* Tích hợp các Unit đơn lẻ thành các hệ thống nhỏ (*Subsystem*) và cuối cùng là nguyên hệ thống hoàn chỉnh (*System*) chuẩn bị cho kiểm thử ở mức hệ thống (*System Test*).

Trong Unit Test, lập trình viên cố gắng phát hiện lỗi liên quan đến chức năng và cấu trúc nội tại của Unit. Có một số phép kiểm thử đơn giản trên giao tiếp giữa Unit với các thành phần liên quan khác, tuy nhiên mọi giao tiếp liên quan đến Unit chỉ thật sự được kiểm tra đầy đủ khi các Unit tích hợp với nhau trong khi thực hiện Integration Test.

Trừ một số ít ngoại lệ, Integration Test chỉ nên thực hiện trên những Unit đã được kiểm tra cẩn thận trước đó bằng Unit Test, và tất cả các lỗi mức Unit đã được sửa chữa. Một số người hiểu sai rằng Unit một khi đã qua giai đoạn Unit Test với các giao tiếp giả lập thì không cần phải thực hiện Integration Test nữa. Thực tế việc tích hợp giữa các Unit dẫn đến những tình huống hoàn toàn khác.

Một chiến lược cần quan tâm trong Integration Test là nên tích hợp dần từng Unit. Một Unit tại một thời điểm được tích hợp vào một nhóm các Unit khác đã tích hợp trước đó và đã hoàn tất các đợt Integration Test trước đó. Lúc này, ta chỉ cần kiểm thử giao tiếp của Unit mới thêm vào với hệ thống các Unit đã tích hợp trước đó, điều này sẽ làm cho số lượng can kiểm thử giảm đi rất nhiều, và sai sót sẽ giảm đáng kể.

Các chiến lược kiểm thử tích hợp

* + Kiểm thử dưới lên (*bottom – up testing*)
  + Kiểm thử trên xuống (*up – down testing*)
  + Kiểm thử hồi qui (*regression testing)*
* **Kiểm thử dưới lên (*bottom – up testing*)**

Tích hợp từ dưới lên

* + Kiểm thử dưới lên là quá trình tích hợp và kiểm thử các module ở cấp thấp trước.
  + Các module cấp thấp được tổ hợp vào các chùm thực hiện cho một chức năng con phần mềm riêng biệt.
  + Driver được xây dựng để phối hợp việc vào ra của test case.
  + Kiểm thử các chùm.
  + Loại bỏ driver và các chùm được tổ hợp di chuyển lên trong cấu trúc chương trình.



Ưu điểm của kiểm thử từ dưới lên

* + Tránh phải tạo các stub phức tạp hay tạo các kết quả nhân tạo.

Nhược điểm của kiểm thử từ dưới lên

* + Phát hiện chậm các lỗi thiết kế.
  + Chậm có phiên bản thực hiện được của hệ thống.
* **Tích hợp từ trên xuống**
  + Các module được tích hợp bằng cách đi dần xuống qua cấp bậc điều khiển, bắt đầu với module điều khiển chính sẽ được tổ hợp dần dần vào trong cấu trúc theo chiều sâu trước hoặc chiều rộng trước.
  + Module kiểm thử chính được dùng như một bộ phận điều khiển kiểm thử (Test driver) và các stub được thế vào cho tất cả các module phụ thuộc trực tiếp vào các điều khiển chính.
* Tùy theo cách tiếp cận tích hợp mà các stub phụ thuộc được thế vào tất cả các module phụ thuộc trực tiếp vào các module điều khiển chính.
  + Việc kiểm thử được tiến hành khi từng module được tích hợp vào.
  + Khi hoàn thành từng tập các kiểm thử, stub khác được thay thế bằng các module thực.
  + Kiểm thử hồi qui được tiến hành.

Ưu điểm của kiểm thử trên xuống

* + Phát hiện sớm các lỗi thiết kế.
  + Có phiên bản hoạt động sớm.

Nhược điểm của kiểm thử trên xuống

* + Khó có thể mô phỏng được các chức năng của module cấp thấp phức tạp.
  + Không kiểm thử đầy đủ các chức năng.
* **Kiểm thử hồi qui**

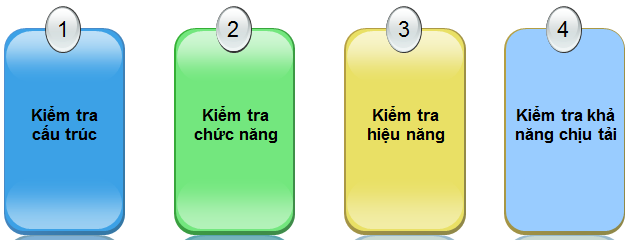
Là tiến hành lại các test case đã kiểm thử thành công mỗi khi tích hợp thêm module hoặc khi cập nhật mã nguồn.

* Vì sao phải kiểm thử hồi qui

Khi tích hợp thêm module hoặc cập nhật mã nguồn sẽ dẫn đến:

* + Xuất hiện lỗi ở các module trước đây chưa gây lỗi.
  + Khắc phục một lỗi mới có thể làm ảnh hưởng đến lỗi mà chúng ta đã sữa.
  + Sinh ra lỗi mới mà trước đây chưa có.

Có 4 loại kiểm thử trong Integration Test:



* ***Kiểm thử cấu trúc (Structure Test):*** Tương tự White Box Test, kiểm thử cấu trúc nhằm bảo đảm các thành phần bên trong của một chương trình chạy đúng và chú trọng đến hoạt động của các thành phần cấu trúc nội tại của chương trình chẳng hạn các câu lệnh và nhánh bên trong.
* ***Kiểm thử chức năng (Functional Test):*** Tương tự Black Box Test, kiểm thử chức năng chỉ chú trọng đến chức năng của chương trình, mà không quan tâm đến cấu trúc bên trong, chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật.
* ***Kiểm thử hiệu năng (Performance Test):*** Kiểm thử việc vận hành của hệ thống.
* ***Kiểm thử khả năng chịu tải (Stress Test):*** Kiểm thử các giới hạn của hệ thống.

#### 1.2.1.3 Kiểm thử hệ thống

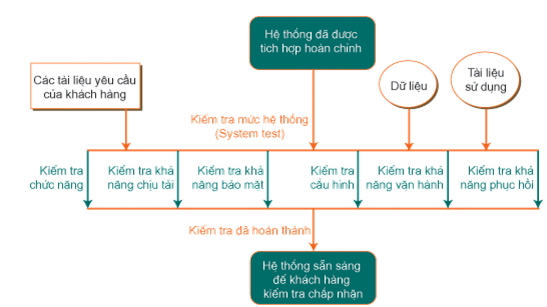
Mục đích System Test là kiểm thử thiết kế và toàn bộ hệ thống (sau khi tích hợp) có thỏa mãn yêu cầu đặt ra hay không.

System Test bắt đầu khi tất cả các bộ phận của phần mềm đã được tích hợp thành công. Thông thường loại kiểm thử này tốn rất nhiều công sức và thời gian. Trong nhiều trường hợp, việc kiểm thử đòi hỏi một số thiết bị phụ trợ, phần mềm hoặc phần cứng đặc thù, đặc biệt là các ứng dụng thời gian thực, hệ thống phân bố, hoặc hệ thống nhúng. Ở mức độ hệ thống, người kiểm thử cũng tìm kiếm các lỗi, nhưng trọng tâm là đánh giá về hoạt động, thao tác, sự tin cậy và các yêu cầu khác liên quan đến chất lượng của toàn hệ thống.

Điểm khác nhau then chốt giữa Integration Test và System Test là System Test chú trọng các hành vi và lỗi trên toàn hệ thống, còn Integration Test chú trọng sự giao tiếp giữa các đơn thể hoặc đối tượng khi chúng làm việc cùng nhau. Thông thường ta phải thực hiện Unit Test và Integration Test để bảo đảm mọi Unit và sự tương tác giữa chúng hoạt động chính xác trước khi thực hiện System Test.

Sau khi hoàn thành Integration Test, một hệ thống phần mềm đã được hình thành cùng với các thành phần đã được kiểm tra đầy đủ. Tại thời điểm này, lập trình viên hoặc kiểm thử viên bắt đầu kiểm thử phần mềm như một hệ thống hoàn chỉnh. Việc lập kế hoạch cho System Test nên bắt đầu từ giai đoạn hình thành và phân tích các yêu cầu.

System Test kiểm thử cả các hành vi chức năng của phần mềm lẫn các yêu cầu về chất lượng như độ tin cậy, tính tiện lợi khi sử dụng, hiệu năng và bảo mật. Mức kiểm thử này đặc biệt thích hợp cho việc phát hiện lỗi giao tiếp với phần mềm hoặc phần cứng bên ngoài, chẳng hạn các lỗi "tắc nghẽn" (deadlock) hoặc chiếm dụng bộ nhớ. Sau giai đoạn System Test, phần mềm thường đã sẵn sàng cho khách hàng hoặc người dùng cuối cùng kiểm thử chấp nhận sản phẩm (*Acceptance Test*) hoặc dùng thử (*Alpha/Beta Test*).



Hình 2.2.4.1 Các loại kiểm thử trong System Test

Đòi hỏi nhiều công sức, thời gian và tính chính xác, khách quan, System Test thường được thực hiện bởi một nhóm kiểm thử viên hoàn toàn độc lập với nhóm phát triển dự án. Bản thân System Test lại gồm nhiều loại kiểm thử khác nhau, phổ biến nhất gồm:

* ***Kiểm thử chức năng (Functional Test):*** Bảo đảm các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế.
* ***Kiểm thử hiệu năng (Performance Test):*** Bảo đảm tối ưu việc phân bổ tài nguyên hệ thống (ví dụ bộ nhớ) nhằm đạt các chỉ tiêu như thời gian xử lý hay đáp ứng câu truy vấn...
* ***Kiểm thử khả năng chịu tải (Stress Test hay Load Test):*** Bảo đảm hệ thống vận hành đúng dưới áp lực cao (ví dụ nhiều người truy xuất cùng lúc). Stress Test tập trung vào các trạng thái tới hạn, các "điểm chết", các tình huống bất thường như đang giao dịch thì ngắt kết nối (xuất hiện nhiều trong kiểm tra thiết bị như POS, ATM...)...
* ***Kiểm thử cấu hình (Configuration Test).***
* ***Kiểm thử bảo mật (Security Test):*** Bảo đảm tính toàn vẹn, bảo mật của dữ liệu và của hệ thống.
* ***Kiểm thử khả năng phục hồi (Recovery Test):*** Bảo đảm hệ thống có khả năng khôi phục trạng thái ổn định trước đó trong tình huống mất tài nguyên hoặc dữ liệu; đặc biệt quan trọng đối với các hệ thống giao dịch như ngân hàng trực tuyến...

Nhìn từ quan điểm người dùng, các cấp độ kiểm thử trên rất quan trọng: Chúng bảo đảm hệ thống đủ khả năng làm việc trong môi trường thực.

Lưu ý là không nhất thiết phải thực hiện tất cả các loại kiểm thử nêu trên. Tùy yêu cầu và đặc trưng của từng hệ thống, tuỳ khả năng và thời gian cho phép của dự án, khi lập kế hoạch, người Quản lý dự án sẽ quyết định áp dụng những loại kiểm thử nào.

#### 1.2.1.4 Kiểm thử chấp nhận

Thông thường, sau giai đoạn System Test là Acceptance Test, được khách hàng thực hiện (hoặc ủy quyền cho một nhóm thứ ba thực hiện). Mục đích của Acceptance Test là để chứng minh phần mềm thỏa mãn tất cả yêu cầu của khách hàng và khách hàng chấp nhận sản phẩm (và trả tiền thanh toán hợp đồng).

Acceptance Test có ý nghĩa hết sức quan trọng, mặc dù trong hầu hết mọi trường hợp, các phép kiểm thử của System Test và Acceptance Test gần như tương tự, nhưng bản chất và cách thức thực hiện lại rất khác biệt.

Đối với những sản phẩm dành bán rộng rãi trên thị trường cho nhiều người sử dụng, thông thường sẽ thông qua hai loại kiểm thử gọi là kiểm thử Alpha – ***Alpha Test*** và kiểm thử Beta – ***Beta Test***. Với Alpha Test, người dùng kiểm thử phần mềm ngay tại nơi phát triển phần mềm, lập trình viên sẽ ghi nhận các lỗi hoặc phản hồi, và lên kế hoạch sửa chữa. Với Beta Test, phần mềm sẽ được gửi tới cho người dùng để kiểm thử ngay trong môi trường thực, lỗi hoặc phản hồi cũng sẽ gửi ngược lại cho lập trình viên để sửa chữa.

Thực tế cho thấy, nếu khách hàng không quan tâm và không tham gia vào quá trình phát triển phần mềm thì kết quả Acceptance Test sẽ sai lệch rất lớn, mặc dù phần mềm đã trải qua tất cả các kiểm thử trước đó. Sự sai lệch này liên quan đến việc hiểu sai yêu cầu cũng như sự mong chờ của khách hàng. Ví dụ đôi khi một phần mềm xuất sắc vượt qua các phép kiểm thử về chức năng thực hiện bởi nhóm thực hiện dự án, nhưng khách hàng khi kiểm thử sau cùng vẫn thất vọng vì bố cục màn hình nghèo nàn, thao tác không tự nhiên, không theo tập quán sử dụng của khách hàng v.v...

Gắn liền với giai đoạn Acceptance Test thường là một nhóm những dịch vụ và tài liệu đi kèm, phổ biến như hướng dẫn cài đặt, sử dụng v.v... Tất cả tài liệu đi kèm phải được cập nhật và kiểm thử chặt chẽ.

### 1.2.2 Một số loại kiểm thử

#### Kiểm thử cấu hình:

* Định nghĩa **:** Kiểm thử cấu hình là quá trình kiểm tra sự vận hành của phần mềm được test với tất cả các loại phần cứng khác nhau.
* Tại sao phải kiểm thử cấu hình**:**
* Phần mềm được test có thể chạy trên nhiều loại cấu hình khác nhau được cung cấp bởi nhiều nhà sản xuất khác nhau. Điều quan trọng cần nhớ là phần mềm của bạn có thể chạy tốt, trơn tru trên một loại cấu hình nhưng sẽ bị lỗi (treo) trên các loại cấu hình khác.
* Phần mềm khác nhau có thể liên kết đến các vùng cấu hình khác nhau,

ví dụ:

* Một game đồ họa cao đòi hỏi rất nhiều đến âm thanh, video.
* Một chương trình in thiệp sẽ dễ bị lỗi nếu máy in có vấn đề
* .Một chương trình truyền thông cần phải được thử nghiệm với nhiều modem và cấu hình mạng
* Khi nào thực hiện kiểm thử cấu hình: Thông thường việc lập lịch kiểm thử cấu hình sau khi các chức năng trong giai đoạn kiểm thử hệ thống đã được thực hiện xong và đã baseline. Nếu không như vậy sẽ gặp khó khăn trong việc xác định các sai sót, khiếm khuyết nằm trong phần mềm hay phần cứng.

#### Kiểm thử khả năng tương thích

* Định nghĩa: Kiểm thử khả năng tương thích là quá trình kiểm tra xem phần mềm có tương tác và chia sẻ thông tin với các phần mềm khác chính xác như thế nào.
* Tương tác có thể xảy ra giữa**:**
* Hai chương trình đang chạy đồng thời trên cùng một máy tính.
* Hai chương trình đang chạy trên những máy tính khác nhau kết nối nhau qua mạng.
* Đơn giản như lưu dữ liệu lên ổ đĩa mềm và mang nó đến một máy tính khác.
* Tại sao kiểm thử khả năng tương thích lại cần thiết : Các phần mềm có thể tương tác với nhau. Nếu như tương tác là thất bại, thì chức năng cũng thất bại.
* Khi nào thực hiện kiểm thử khả năng tương thích
  + Cũng giống như kiểm thử cấu hình chúng ta thường lập lịch kiểm thử khả Đinăng tương thích sau khi đã kiểm thử xong các chức năng trong giai đoạn kiểm thử hệ thống và đã được baseline. Nếu không việc tìm ra lỗi tương thích là rất khó.
  + Hầu hết các đội dự án thường kết hợp kiểm thử cấu hình với kiểm thử khả năng tương thích với nhau.

#### Kiểm thử Foreign-language

* Định nghĩa**:**  Là quá trình kiểm tra xem phần mềm có thể làm việc chính xác với các ngôn ngữ quốc tế?

Điều này yêu cầu khả năng ngoại ngữ của tester để kiểm tra các phiên bản ngôn ngữ khác nhau nếu không có sẵn công cụ trợ giúp. Các chuyên gia ngôn ngữ thường tham gia vào kiểm thử này.

* Tại sao phải kiểm thử foreign - language

Hầu hết các phần mềm hiện nay đều được phát hành trên toàn thế giới và điều đó tạo một tiền đề tốt để thiết kế và kiểm thử các phần mềm được phân phối trên toàn thế giới.

* + Khi nào thực hiện kiểm thử Foreign- language
  + Dịch thuật:
    - * Văn bản mở rộng: khi một từ được dịch sang ngôn ngữ khác, nó thường đòi hỏi nhiều đặc trưng khác để nói về một điều tương tự.
      * Code khác nhau cho những ngôn ngữ khác nhau: ASCII, DBCS và Unicode.
      * Phím nóng và phím tắt: tất cả các phím nóng và phím tắt nên được kiểm tra để đảm bảo chúng hoạt động đúng và dễ dàng sử dụng.
      * Đọc từ trái sang phải và từ phải sang trái: một số ngôn ngữ đọc từ phải sang trái và ngược lại.
      * Văn bản trong đồ họa
* Các vấn đề về địa lý
* Nội dung: Cùng một từ có thể có ý nghĩa khác nhau trong các quốc gia khác nhau.
* Định dạng dữ liệu: ngôn ngữ khác nhau có các định dạng dữ liệu khác nhau như: số đo, số, tiền tệ, ngày, giờ, lịch, địa chỉ, số điện thoại, và kích thước.

#### Kiểm thử khả năng sử dụng

* Định nghĩa:
  + Tính tiện dụng là đặc tính dễ hiểu, dễ sử dụng bởi người dùng cuối.
  + Kiểm thử tính tiện dụng là kiểm tra những đặc tính này của phần mềm.
* Kiểm thử khả năng tiện dụng bao gồm:
  + Kiểm tra cài đặt
  + Kiểm tra chức năng khả dụng
  + Kiểm tra giao diện người dùng.
  + Kiểm tra thông tin

#### Kiểm thử tài liệu

* Định nghĩa: Kiểm thử tài liệu là quá trình xác nhận tính chính xác của tài liệu phần mềm, và đảm bảo tính đúng đắn của các bước hoạt động được mô tả trong hướng dẫn sử dụng,
* Tại sao phải kiểm thử tài liệu:
* Tài liệu là một phần quan trọng của toàn bộ sản phẩm phần mềm và tiếp xúc trực tiếp với người dùng cuối.
* Người dùng có thể vận hành phần mềm theo quy định của một số tài liệu. Nếu có lỗi trong tài liệu, khách hàng sẽ khó chịu.
* Các tài liệu phổ biến**:**
  + Tài liệu người dùng:
    - Hướng dẫn sử dụng
    - Tài liệu hướng dẫn hoạt động
    - Tài liệu bảo trì
  + Tài liệu phát triển phần mềm
    - Đặc tả yêu cầu phần mềm
    - Đặc tả thiết kế ở mức cao
    - Đặc tả thiết kế ở mức thấp
    - Kế hoạch kiểm thử
    - Kế hoạch dự án

#### Kiểm thử khả năng bảo mật

* Định nghĩa**:**
* Bảo mật là thuộc tính của phần mềm để đánh giá khả năng bảo mật chống lại các cuộc tấn công trái phép của hệ thống.
* Kiểm thử bảo mật là quá trình đảm bảo chiến lược bảo mật được xây dựng trong hệ thống làm việc một cách chính xác.
* Bảo mật là thuộc tính quan trọng của hệ thống nên cần thực hiện kiểm tra càng sớm càng tốt.

## Các phương pháp kiểm thử phần mềm

### 1.3.1 Phương pháp kiểm thử tĩnh

* Kiểm thử tĩnh là phương pháp kiểm thử phần mềm đòi hỏi phải duyệt lại các yêu cầu và các đặc tả bằng tay thông qua việc sử dụng giấy, bút để kiểm tra logic, lần từng chi tiết mà không cần chạy chương trình.
* Kiểu kiểm thử này thường được sử dụng bởi chuyên viên thiết kế hay những người viết mã lệnh một mình.
* Mục đích của kiểm thử tĩnh**:**
* Tìm lỗi sớm trong code và trong mô hình phần mềm, để giảm chi phí cho việc sửa lỗi.
* Dễ dàng cô lập lỗi.
* Đảm bảo cho phần mềm hoạt động đúng theo yêu cầu khách hàng
* Kỹ thuật chính trong kiểm thử tĩnh là review
* Review chéo (Peer – review)
* Thanh tra mã nguồn (Code Inspection)
* Tổng duyệt (Walk Through)

Tổng duyệt – Walkthrough

*Walkthrough* là một thuật ngữ mô tả sự xem xét kỹ lưỡng của một quá trình ở mức trừu tượng trong đó nhà thiết kế hay lập trình viên lãnh đạo các thành viên trong nhóm và những người có quan tâm khác thông qua một sản phẩm phần mềm, và những người tham gia đặt câu hỏi, và ghi chú những lỗi có thể có, sự vi phạm các chuẩn phát triển và các vấn đề khác. Walkthrough mã lệnh là 1 tập các thủ tục và các công nghệ dò lỗi cho việc đọc nhóm mã lệnh. Trong một *Walkthrough,* nhóm các nhà phát triển – với 3 hoặc 4 thành viên là tốt nhất – thực hiện xét duyệt lại. Chỉ 1 trong các thành viên là tác giả của chương trình.

Một ưu điểm khác của *walkthroughs*, hiệu quả trong chi phí gỡ lỗi, là một thực tế mà khi một lỗi được tìm thấy, nó thường được định vị chính xác trong mã lệnh. Thêm vào đó, phương pháp này thường tìm ra một tập các lỗi, cho phép sau đó các lỗi đó được sửa tất cả với nhau. Mặt khác, kiểm thử dựa trên máy tính,chỉ tìm ra triệu chứng của lỗi (chương trình không kết thúc hoặc đưa ra kết quả vô nghĩa), và các lỗi thường được tìm ra và sửa lần lượt từng lỗi một.

Thanh tra mã nguồn – Code Inspection

Thanh tra mã nguồn là 1 tập hợp các thủ tục và các kỹ thuật dò lỗi cho việc đọc các nhóm mã lệnh. Một nhóm kiểm duyệt thường gồm 4 người. Một trong số đó đóng vai trò là người điều tiết – một lập trình viên lão luyện và không được là tác giả của chương trình và phải không quen với các chi tiết của chương trình. Người điều tiết có nhiệm vụ: phân phối nguyên liệu và lập lịch cho các buổi kiểm duyệt, chỉ đạo phiên làm việc, ghi lại tất cả các lỗi được tìm thấy và đảm bảo là các lỗi sau đó được sửa. Thành viên thứ hai là một lập trình viên. Các thành viên còn lại trong nhóm thường là nhà thiết kế của chương trình ( nếu nhà thiết kế khác lập trình viên) và một chuyên viên kiểm thử.

### Phương pháp kiểm thử động

Kiểm thử độnglà phương pháp thử phần mềm thông qua việc dùng máy chạy chương trình để kiểm tra trạng thái tác động của chương trình.

Phương pháp kiểm thử dựa trên các ca kiểm thử xác định bằng sự thực hiện của đối tượng kiểm thử hay chạy các chương trình.

Mục đích:kiểm tra cách thức hoạt động động của mã lệnh, tức là kiểm tra sự phản ứng vật lý từ hệ thống tới các biến luôn thay đổi theo thời gian

Trong kiểm thử động thì chương trình phải được biên dịch và vận hành (chạy chương trình).

Kiểm thử động thực sự bao gồm làm việc với phần mềm, nhập các giá trị đầu vào và kiểm tra xem đầu ra có như mong muốn hay không?

Có 2 cách kiểm thử:

* + Hiểu rõ chức năng của một hàm hay module, nhằm tìm ra lỗi chức năng.
  + Hiểu rõ cách hoạt động của một hàm/module. Kiểm tra tất cả các thao tác bên trong của một hàm dựa theo mô tả và tất cả các thành phần nội bộ.

### Phương pháp kiểm thử hộp đen

Phương pháp kiểm thử tập trung vào yêu cầu về mặt chức năng của phần mềm.Nó cho phép tạo ra một tập các điều kiện input để kiểm thử tất cả các chức năng của một chương trình .

Mục đích của kiểm thử hộp đen là tập trung trên các vùng thông tin, hướng dữ liệu hay hướng vào/ ra.

Các trường hợp kiểm thử được tạo ra dựa vào bản mô tả chức năng chứ không phải tập trung trên vùng mã chương trình.

* Kiểm thử hộp đen nhằm phát hiện các lỗi sau**:**
  + Không đúng hay thiếu một số chức năng, hàm/module.
  + Giao diện không phù hợp, lỗi về interface.
  + Lỗi về cấu trúc dữ liệu hay thao tác lên dữ liệu bên ngoài.
  + Lỗi thực thi…
* Các kỹ thuật kiểm thử hộp đen
  + Kỹ thuật phân vùng tương đương.
  + Kỹ thuật phân tích giá trị biên
  + Kỹ thuật đoán lỗi
  + Kỹ thuật dựa trên use case.

### 