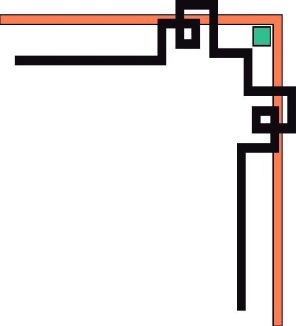
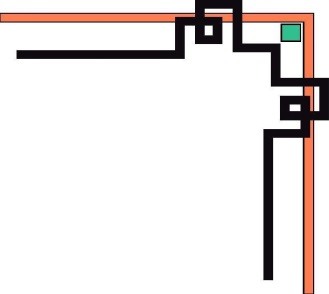
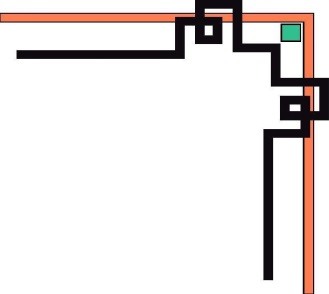
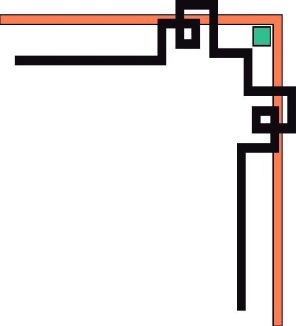
**TRƯỜNG ĐẠI HỌC ĐÔNG Á**



**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI MÔN**

**MÔN: KIỂM THỬ PHẦN MỀM**

**Họ và tên SV : Hoàng Đình Tuấn**

**ID Sinh Viên : 48859**

**Lớp : ST19A1B**

**Ngành : Công nghệ phần mềm**

**GVHD : Tạ Quốc Ý**

1. **Software testing**
2. **Khái niệm**

* Kiểm thử phần mềm là một phương pháp để kiểm tra xem sản phẩm phần mềm thực tế có phù hợp với các yêu cầu mong đợi hay không và để đảm bảo rằng sản phẩm phần mềm đó không có khiếm khuyết . Nó liên quan đến việc thực thi các thành phần phần mềm / hệ thống bằng cách sử dụng các công cụ thủ công hoặc tự động để đánh giá một hoặc nhiều thuộc tính quan tâm. Mục đích của kiểm thử phần mềm là xác định các lỗi, khoảng trống hoặc các yêu cầu còn thiếu đối lập với các yêu cầu thực tế.

1. **Tầm quan trọng**

* Kiểm thử phần mềm là rất quan trọng vì nếu có bất kỳ lỗi hoặc lỗi nào trong phần mềm, nó có thể được xác định sớm và có thể được giải quyết trước khi giao sản phẩm phần mềm. Sản phẩm phần mềm được kiểm tra thích hợp đảm bảo độ tin cậy, bảo mật và hiệu suất cao, giúp tiết kiệm thời gian, hiệu quả chi phí và sự hài lòng của khách hàng.

1. **Lợi ích của kiểm thử**

* Hiệu quả về chi phí: Đây là một trong những lợi thế quan trọng của kiểm thử phần mềm. Kiểm tra bất kỳ dự án CNTT nào đúng thời hạn sẽ giúp bạn tiết kiệm tiền của mình trong dài hạn. Trong trường hợp nếu các lỗi được phát hiện trong giai đoạn kiểm thử phần mềm trước đó, thì chi phí sửa chữa sẽ thấp hơn.
* Bảo mật: Đây là lợi ích nhạy cảm và dễ bị tấn công nhất của kiểm thử phần mềm. Mọi người đang tìm kiếm các sản phẩm đáng tin cậy. Nó giúp loại bỏ các rủi ro và vấn đề sớm hơn.
* Chất lượng sản phẩm: Đây là yêu cầu thiết yếu của bất kỳ sản phẩm phần mềm nào. Kiểm tra đảm bảo một sản phẩm chất lượng được cung cấp cho khách hàng.
* Sự hài lòng của khách hàng: Mục đích chính của bất kỳ sản phẩm nào là mang lại sự hài lòng cho khách hàng của họ. Kiểm tra UI / UX đảm bảo trải nghiệm người dùng tốt nhất.

1. **Kiểm thử trong Kỹ thuật phần mềm**

* Kiểm thử trong Kỹ thuật Phần mềm là một quá trình đánh giá một sản phẩm phần mềm để tìm xem liệu sản phẩm phần mềm hiện tại có đáp ứng các điều kiện bắt buộc hay không. Quá trình kiểm tra liên quan đến việc đánh giá các tính năng của sản phẩm phần mềm đối với các yêu cầu về bất kỳ yêu cầu nào còn thiếu, lỗi hoặc lỗi, bảo mật, độ tin cậy và hiệu suất.

1. **White box testing**
2. **Khái niệm**

* Kiểm tra các đoạn mã chương trình xem nó có vận hành đúng như thiết kế hay không.
* Kiểm thử hộp trắng dựa trên việc xem xét cấu trúc bên trong của chương trình theo cấu trúc điều khiển và hoạt động của chúng
* Các tên gọi khác: kiểm thử cấu trúc (structural testing), kiểm thử hộp kính (glass box), kiểm thử rõ ràng (clear box testing).
* Đối tượng chính của kiểm thử hộp trắng là tập trung vào cấu trúc bên trong chương trình và tìm ra tất cả những lỗi bên trong chương trình.
* Việc kiểm tra tập trung chủ yếu vào:
* Logic bên trong chương trình và cấu trúc dữ liệu.
* Cấu trúc chương trình: Những câu lệnh và các nhánh, các loại đường dẫn chương trình.
* Những hành động và trạng thái bên trong chương trình.

1. **Ưu điểm và nhược điểm của kiểm thử hộp trắng**
2. **Ưu điểm:**

* Khi sử dụng kiểm thử hộp trắng, kiểm thử viên có thể chắc chắc rằng mọi đường xuyên qua phần mềm cần kiểm thử đã được xác định và kiểm thử.

1. **Nhược điểm:**

* Không đủ khả năng kiểm thử hết các đường thi hành vì số lượng quá nhiều
* Kiểm thử bằng hộp trắng không thể đảm bảo rằng chương trình đã tuân theo đặc tả
* Không phát hiện ra chương trình sai do thiếu đường dẫn
* Không phát hiện được lỗi do sai dữ liệu
* Kiểm thử viên cần có các kỹ năng về lập trình để hiểu và đánh giá được phần mềm. Không may là hiện nay có nhiều kiểm thử viên không có được nền tảng tốt về lập trình

1. **Kiểm thử luồng điều khiển**

**a. Đường thi hành**

* Đường thi hành (Execution path):
* Là 1 kịch bản thi hành đơn vị phần mềm tương ứng
* Là danh sách có thứ tự các lệnh được thi hành ứng với 1 lần chạy cụ thể của đơn vị phần mềm, bắt đầu từ điểm nhập của đơn vị phần mềm đến điểm kết thúc của đơn vị phần mềm.
* Cho dù có kiểm thử hết được toàn bộ các đường thi hành thì vẫn không thể phát hiện những đường thi hành cần có nhưng không (chưa) được hiện thực :

Vd: if (a>0) doIsGreater(); if (a==0) dolsEqual();

=> thiếu việc xửlý trường hợp a < 0 - if (a<0) dolsLess();

1. **Đồ thị lưu hành**

* Là một trong các phương pháp mô tả thuật giải Mô tả các cấu trúc điều khiển phổ dụng câu lệnh
* If…then
* If…then…else
* While…do
* Do…while
* For…
* Switch…

1. **Đồ thị dòng**

* Đồ thị dòng (flow graph) được Tom McCabe đề xuất năm 1976
* Đồ thị dòng được xây dựng từ đồ thị dòng điều khiển của chương trình bằng cách:
* gộp các lệnh tuần tự liên tiếp thành một nút
* thay các lệnh rẽ nhánh (độc lập) thành một nút
* thay điểm hợp nhất của các lệnh rẽ nhánh bằng một nút.
* Cấu trúc của một đồ thị dòng bao gồm:
* Mỗi nút (hình tròn) biểu thị một vài (có thể là 0) câu lệnh thủ tục.
* Mỗi cạnh biểu diễn dòng điều khiển nối hai nút với nhau.
* Miền là phần mặt phẳng giới hạn bởi các cung và nút của đồ thị.
* Những nút biểu thị sự phân nhánh hoặc hội nhập của các cung được gọi là nút vị từ.
* Cấu trúc cơ bản của đồ thị dòng:

If If-then-else for or while Do-while switch

* Độ phức tạp chu trình:
* Là số đo sự phức tạp logic của chương trình.
* Là số các đường đi độc lập cơ bản trong tập các con đường độc lập của một chương trình.
* Là số đường độc lập nhỏ nhất phủ hết các cạnh của đồ thị luồng.
* Số đo này là giới hạn trên của số ca kiểm thử cần phải tiến hành để đảm bảo rằng, tất cả các câu lệnh trong chương trình đều được thực hiện ít nhất một lần.

1. **Black box testing**
2. **Tổng quan về kiểm thử hộp đen**

* Phương pháp kiểm thử hộp đen: coi hệ thống là một hộp đen, không thể thấy được cấu trúc logic bên trong. Người làm kiểm thử tập trung vào các yêu cầu chức năng của phần mềm dựa trên các dữ liệu lấy từ đặc tả
* Đặc trưng:
* Nhằm thuyết minh: các chức năng phần mềm đủ & vận hành đúng
* Thực hiện các phép thử qua giao diện
* Kiểm thử hộp đen nhằm tìm ra các loại sai:
* Chức năng thiếu hoặc không đúng đắn.
* Sai về giao diện.
* Sai trong cấu trúc hoặc trong truy cập dữ liệu ngoài.
* Sai thực thi chức năng.
* Sai khởi đầu hoặc kết thúc mô đun.

1. **Các phương pháp kiểm thử hộp đen**

**a. Phân hoạch tương đương (Equivalence Partitioning)**

* Ý tưởng: Chia miền vào chương trình thành các lớp dữ liệu. Xác định đầu vào hợp lệ và không hợp lệ để lập các ca kiểm thử theo các lớp đó
* Mỗi lớp dùng để kiểm thử một chức năng, gọi là lớp tương đương.
* Thay vì kiểm tra tất cả các giá trị đầu vào, có thể lựa chọn từ đầu vào cho riêng từng lớp.
* Nguyên tắc xác định lớp tương đương:
* Nếu điều kiện đầu vào định rõ giới hạn của một mảng, hoặc một giá trị xác định thì chia vùng tương đương thành:
* Một lớp tương đương hợp lệ
* Hai lớp không hợp lệ
* Một lớp đặc biệt (nếu có)
* Nếu điều kiện đầu vào chỉ định là một tập giá trị, hoặc xác định là một kiểu đúng sai thì chia vùng tương đương thành :
* Một lớp tương đương hợp lệ.
* Một lớp tương đương không hợp lệ.
* Một lớp đặc biệt (nếu có)

**b. Phân tích giá trị biên (Boundary Value Analysis)**

* Cơ sở : Tập trung phân tích các giá trị biên của miền dữ liệu để xây dựng dữ liệu kiểm thử
* Nguyên tắc : đối với một biến, kiểm thử các dữ liệu vào gồm:
* Giá trị nhỏ nhất: min
* Giá trị gần kề lớn hơn giá trị nhỏ nhất: min+1
* Giá trị gần kề nhỏ hơn giá trị nhỏ nhất: min -1
* Giá trị lớn nhất : max
* Giá trị gần kề lớn hơn giá trị lớn nhất: max+1
* Giá trị gần kề nhỏ hơn giá trị lớn nhất: max-1

**c. Bảng hỗ trợ quyết định**

1. **Manual testing**
2. **Khái niệm**

* Manual Testing là một trong những công việc theo dạng kiểm thử phần mềm, hoặc là một chương trình được thực hiện bằng tay bởi các tester mà không thông qua bất kỳ công cụ hỗ trợ nào.
* Nó hoạt động dựa vào mục đích phát hiện các lỗi bug từ nhỏ cho đến lớn trong phần mềm.

1. **Ưu và nhược điểm**

**a. Ưu điểm**

* Hầu hết, các tester sẽ dễ dàng hơn trong việc test giao diện giúp mang lại những phản hồi nhanh chóng và trực quan nhất về giao diện ứng dụng của phần mềm cần test.
* Nếu như chương trình có thay đổi nhỏ, thì Manual Testing sẽ không phải mất quá nhiều thời gian cho việc kiểm tra đối với những trường hợp kiểm thử.
* Các tester sẽ có nhiều hơn cơ hội cho việc khám phá kiếm thử. Từ đó, hiểu thêm về những lỗi phát sinh khó phát hiện để có thể nâng cao được tay nghề cho tester.

**b. Nhược điểm**

* MT sẽ tiêu tốn nhiều thời gian cũng như công sức của tester hơn trong việc phát hiện ra các lỗi bug.
* Kết quả tìm thấy thường ít được tin cậy hơn bởi có thể do những sai sót của con người khi thực hiện.
* Đối với những task công việc lớn có dạng như: performance testing và stress testing; thì công việc đòi hỏi phải có những tools thực sự chuyên nghiệp để có thể hỗ trợ kỹ thuật manual testing nên việc thực hiện sẽ rất khó.
* Việc sử dụng Manual Testing sẽ giúp bạn phát hiện lỗi thường ít hơn so với kỹ thuật tự động test.
* Chi phí dành cho dự án có thể phải tăng lên do những đòi hỏi trong việc phải thuê nguồn nhân lực MT để tiến hành kiểm tra.
* Quá trình sử dụng tools tự động để test sẽ cho bạn kết quả nhanh hơn cũng như chính xác hơn so với MT.

1. **Automation testing**
2. **Tổng quan kiểm thử tự động**

* Kiểm thử tự động: áp dụng các công cụ giúp thực hiện việc kiểm thử phần mềm.
* Nên sử dụng công cụ tự động khi:
* Không đủ tài nguyên
* Kiểm thử hồi quy
* Kiểm tra khả năng vận hành của phần mềm trong môi trường đặc biệt.
* Test script: nhóm mã lệnh đặc tả kịch bản dùng để tự động hóa một trình tự kiểm thử.
* Test scipt: có thể tạo thủ công hoặc tạo tự động dùng công cụ kiểm thử tự động.

1. **Quy trình kiểm thử tự động**
   1. **Tạo test script**

* Giai đoạn này ta dùng test tool để ghi lại các thao tác lên PM cần kiểm tra và tự động sinh ra test script
  1. **Chỉnh sửa test script**
* Chỉnh sửa lại test script thực hiện kiểm tra theo đúng yêu cầu đặt ra, cụ thể là làm theo test case cần thực hiện
  1. **Chạy test script để kiểm thử tự động**
* Giám sát hoạt động kiểm tra phần mềm của test script
  1. **Đánh giá kết quả**
* Kiểm tra kết quả thông báo sau khi thực hiện kiểm thử tự động. Sau đó bổ sung, chỉnh sửa những sai sót

1. **Ưu, nhược điểm của kiểm thử tự động**
   1. **Ưu điểm**

* Kiểm thử phần mềm không cần can thiệp của tester
* Giảm chi phí thực hiện kiểm tra số lượng lớn các test case hoặc test case lặp lại nhiều lần
* Giả lập tình huống khó có thể thực hiện bằng tay
  1. **Nhược điểm**
* Mất chi phí tạo các script để thực hiện kiểm thử tự động
* Tốn chi phí dành cho bảo trì các script
* Đòi hỏi tester phải có kỹ năng tạo và thay đổi script cho phù hợp test case
* Không áp dụng tìm được các lỗi mới cho phần mềm

1. **Selenium**

* Selenium là bộ kiểm thử tự động miễn phí (mã nguồn mở) dành cho các ứng dụng web trên các trình duyệt và nền tảng khác nhau. Nó khá là giống với HP Quick Test Pro (QTP bây giờ là UFT) chỉ khác là Selenium thì tập trung vào việc tự động hoá các ứng dụng dựa trên nền tảng web. Kiểm thử được thực hiện bằng cách sử dụng công cụ Selenium thường được gọi là Kiểm thử Selenium. Selenium không chỉ là 1 công cụ độc lập mà là 1 bộ công cụ của phần mềm, mỗi bộ đều đáp ứng được nhu cầu kiểm thử khác nhau của 1 tổ chức. Nó có 4 thành phần:
* Selenium Integrated Development Environment (IDE)
* Selenium Remote Control (RC)
* WebDriver
* Selenium Grid
* **Giới thiệu qua về Selenium IDE**
* Môi trường phát triển tích hợp Selenium (IDE) là khuôn khổ đơn giản nhất trong bộ Selenium và là cách đơn giản nhất để học. Đây là một plugin của Firefox mà bạn có thể cài đặt dễ dàng như các plugin khác. Tuy nhiên, vì tính đơn giản của nó, Selenium IDE chỉ nên được sử dụng như là một công cụ tạo mẫu. Nếu bạn muốn tạo các trường hợp thử nghiệm nâng cao hơn, bạn sẽ cần phải sử dụng Selenium RC hoặc WebDriver.
* **Giới thiệu qua về Selenium Remote Control**
* Selenium RC là framework thử nghiệm hàng đầu của toàn bộ dự án Selenium trong một thời gian dài. Đây là công cụ kiểm tra web tự động đầu tiên cho phép người dùng sử dụng ngôn ngữ lập trình mà họ thích. Tính đến phiên bản 2.25.0, RC có thể hỗ trợ các ngôn ngữ lập trình sau:
  + Java
  + C#
  + PHP
  + Python
  + Perl
  + Ruby
* **Giới thiệu qua về WebDriver**
* WebDriver chứng tỏ bản thân nó tốt hơn cả Selenium IDE và Selenium RC về nhiều mặt. Nó thực hiện tiếp cận hiện đại và ổn định hơn trong tự động hoá các hành động của trình duyệt. WebDriver, không giống như Selenium RC, không phụ thuộc vào JavaScript cho Tự động hóa. Nó điều khiển trình duyệt bằng cách liên lạc trực tiếp với nó. Các ngôn ngữ được hỗ trợ giống với Selenium RC.
  + Java
  + C#
  + PHP
  + Python
  + Perl
  + Ruby
* **Selenium Grid**
* Selenium Grid là một công cụ được sử dụng cùng với Selenium RC để chạy thử nghiệm song song trên các máy khác nhau và các trình duyệt khác nhau cùng một lúc. Thực hiện song song có nghĩa là chạy nhiều kiểm thử cùng một lúc. Tính năng:
* Cho phép **chạy đồng thời các thử nghiệm**trong**nhiều trình duyệt và môi trường**.
* **Tiết kiệm thời gia**n to lớn.
* Sử dụng khái niệm hub-and-nodes. Hub hoạt động như một nguồn chính của Selenium lệnh cho mỗi nút kết nối với nó.