**NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN HƯỚNG DẪN**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………….………………………………………………...…………………………………………………………………………………………………………………………………………………………………………………………

*Hưng Yên, ngày…… tháng…... năm........*

***Giáo viên hướng dẫn***

**NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN PHẢN BIỆN 1**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

*Hưng Yên, ngày…… tháng…... năm .........*

***Giáo viên phản biện***

**NHẬN XÉT VÀ ĐÁNH GIÁ CỦA GIÁO VIÊN PHẢN BIỆN 2**

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

*Hưng Yên, ngày…… tháng…... năm .........*

***Giáo viên phản biện***

# MỤC LỤC

[MỤC LỤC 4](#_Toc313009891)

[DANH MỤC CÁC BẢNG 7](#_Toc313009892)

[DANH MỤC CÁC HÌNH VẼ 8](#_Toc313009893)

[LỜI CẢM ƠN 9](#_Toc313009894)

[PHẦN I: MỞ ĐẦU 10](#_Toc313009895)

[1.1. Lý do chọn đề tài 10](#_Toc313009896)

[1.2. Mục đích nghiên cứu 10](#_Toc313009897)

[1.3. Khách thể và đối tượng nghiên cứu 11](#_Toc313009898)

[1.4. Nhiệm vụ nghiên cứu 11](#_Toc313009899)

[1.5. Phạm vi nghiên cứu 11](#_Toc313009900)

[1.6. Phương pháp nghiên cứu 11](#_Toc313009901)

[1.7. Ý nghĩa lý luận và thực tiễn của đề tài 12](#_Toc313009902)

[PHẦN II: NỘI DUNG 13](#_Toc313009903)

[Chương 1: Giới thiệu tổng quan 13](#_Toc313009904)

[1.1. Khái niệm kiểm thử phần mềm: 13](#_Toc313009905)

[1.2. Khái niệm kiểm thử tự động 13](#_Toc313009906)

[1.3. Mục tiêu của kiểm thử tự động 14](#_Toc313009907)

[1.4. Quy trình kiểm thử phần mềm tự động 16](#_Toc313009908)

[1.5. Các công cụ kiểm thử tự động trên thị trường 24](#_Toc313009909)

[1.5.1. QuickTest Professinal: 24](#_Toc313009910)

[1.5.2. DevPartner Studio của công ty Compuware: 29](#_Toc313009911)

[1.5.3. Mercury Interactive 30](#_Toc313009912)

[1.5.4. Phần mềm Rational từ IBM 30](#_Toc313009913)

[1.5.5. Một số công cụ kiểm thử mã nguồn mở: 31](#_Toc313009914)

[Chương 2: Namespace System.Reflection 32](#_Toc313009915)

[2.1. Các khái niệm cơ bản: 32](#_Toc313009916)

[2.1.1. Reflection là gì? 32](#_Toc313009917)

[2.1.2. Lớp System.Type 33](#_Toc313009918)

[2.1.3. Không gian tên System.Reflection 37](#_Toc313009919)

[Chương 3: Namespace System.CodeDom 40](#_Toc313009920)

[3.1. Các khái niệm cơ bản: 40](#_Toc313009921)

[3.1.1. CodeDom là gì? 40](#_Toc313009922)

[3.1.2. Mô hình và các khái niệm cơ bản được sử dụng trong CodeDom: 40](#_Toc313009923)

[3.2. Các lớp trong namespace System.CodeDom 41](#_Toc313009924)

[3.3. Cách sử dụng CodeDom để tạo một chương trình c# 44](#_Toc313009925)

[Chương 4: Xây dựng ứng dụng kiểm thử tự động 47](#_Toc313009926)

[4.1. Đề xuất công cụ kiểm thử phần mềm 47](#_Toc313009927)

[4.2. Không gian tên của .Net và các lớp phục vụ cho phát triển công cụ 48](#_Toc313009928)

[4.3. Lưu trữ dữ liệu trên bảng tính worksheet 51](#_Toc313009929)

[4.4. Phân tích thiết kế hệ thống: 53](#_Toc313009930)

[4.4.1.Đặc tả yêu cầu: 53](#_Toc313009931)

[4.4.2.Phân tích – thiết kế hệ thống: 57](#_Toc313009932)

[4.4.3.Thiết kế giao diện 61](#_Toc313009933)

[PHẦN III: KẾT LUẬN VÀ KHUYẾN NGHỊ 65](#_Toc313009934)

[3.1. Kết quả đạt được 65](#_Toc313009935)

[3.2. Hạn chế 65](#_Toc313009936)

[3.3. Hướng phát triển 65](#_Toc313009937)

[3.4. Đề xuất ý kiến 66](#_Toc313009938)

[TÀI LIỆU THAM KHẢO 67](#_Toc313009939)

# DANH MỤC CÁC BẢNG

[Bảng 1.1: Các chức năng trên giao diện chính của QTP. 27](#_Toc313009207)

[Bảng 2.1: Bảng liệt kê một số item quan trọng của Namespace System.Reflction. 39](#_Toc313009208)

[Bảng 3.1: Thành phần chính trong Namespace System.CodeDom. 43](#_Toc313009209)

[Bảng 3.2: Các kiểu của CodeDom để khai báo kiểu cấu trúc 43](#_Toc313009210)

[Bảng 3.3: Các kiểu của CodeDom để xây dựng các thành phần. 45](#_Toc313009211)

[Bảng 4.1: Các không gian tên được sử dụng phát triển ứng dụng. 52](#_Toc313009212)

[Bảng 4.2: Dòng sự kiện Usecase Tạo Test case. 56](#_Toc313009213)

[Bảng 4.3: Dòng sự kiện chính Usecase Tạo Test script. 57](#_Toc313009214)

[Bảng 4.4: Danh sách các lớp đối tượng. 58](#_Toc313009215)

[Bảng 4.5: Danh sách các thuộc tính của lớp test case . 59](#_Toc313009216)

[Bảng 4.6: Danh sách các phương thức của lớp test case . 59](#_Toc313009217)

[Bảng 4.7: Danh sách các thuộc tính của lớp test script . 60](#_Toc313009218)

[Bảng 4.8: Danh sách các phương thức của lớp test script . 60](#_Toc313009219)

[Bảng 4.9: Danh sách chức năng của module. 63](#_Toc313009220)

[Bảng 4.10: Bảng đặc tả giao diện ứng dụng. 65](#_Toc313009221)

# DANH MỤC CÁC HÌNH VẼ

[Hình 1.1: Quy trình của kiểm thử tự động[3]. 17](#_Toc313007820)

[Hình 1.2:  Bản kế hoạch chính và các bản kế hoạch chi tiết[3]. 18](#_Toc313007821)

[Hình 1.3. Thời điểm phù hợp để thiết lập các kế hoạch kiểm tra[3]. 20](#_Toc313007822)

[Hình 1.4: Quy trình thực hiện kiểm thử của công cụ tự động. 24](#_Toc313007823)

[Hình 1.5: Giao diện QTP 26](#_Toc313007824)

[Hình 1.6: Giao diện viết test script tự động trong QTP 29](#_Toc313007825)

[Hình 3.1: Mô hình các khái niệm cơ bản trong CodeDom. 41](#_Toc313007826)

[Hình 4.1: Kết quả chạy test script lưu trên file MS excel. 54](#_Toc313007827)

[Hình 4.2: Usecase tổng quát. 54](#_Toc313007828)

[Hình 4.3: Usecase Tạo Test case. 55](#_Toc313007829)

[Hình 4.4: Usecase Tạo Test script. 57](#_Toc313007830)

[Hình 4.5: Biểu đồ lớp. 60](#_Toc313007831)

[Hình 4.6: Biểu đồ tuần tự sinh test case. 61](#_Toc313007832)

[Hình 4.7: Biểu đồ tuần tự sinh test script. 62](#_Toc313007833)

[Hình 4.8: Giao diện chương trình. 64](#_Toc313007834)

# LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn thầy cô trong khoa Công nghệ thông tin đã tận tình giảng dạy, bổ sung cho chúng em những kiến thức hiểu biết vô cùng quý báu trong suốt thời gian học tập, cũng như các thầy cô đã tạo điều kiện cho chúng em hoàn thành tốt đồ án 5 trong thời gian thực hiện.

Sau một thời gian nghiên cứu, tìm hiểu chúng em đã hoàn thiện đồ án 5 của mình. Đồ án được thực hiện và hoàn thành tại trường Đại học Sư phạm kỹ thuật Hưng Yên dưới sự hướng dẫn thầy giáo Đào Anh Hiển. Trong thời gian thực hiện nhóm đã gặp rất nhiều khó khăn về mặt kiến thức và nguồn tài liệu nhưng thầy hướng dẫn đã tận tình chỉ bảo, động viên, giúp đỡ, tạo mọi điều kiện hỗ trợ tốt nhất cho chúng em để chúng em hoàn thành được đồ án của mình. Nhóm chúng em xin bày tỏ sự kính trọng và lòng biết ơn đến thầy!

Mặc dù đã rất nỗ lực cố gắng nhưng chắc hẳn đề tài vẫn còn nhiều thiếu sót. Chúng em rất mong nhận được sự góp ý, phê bình của thầy cô để hoàn thiện hơn và có thể phát triển lên làm đồ án tốt nghiệp cho kỳ tới. Chúng em xin cảm ơn!

*Hưng Yên, ngày 10 tháng 8 năm 2011*

Nhóm thực hiện:

*Nguyễn Thị Thi*

*Nguyễn Thị Thảo*

# PHẦN I: MỞ ĐẦU

## Lý do chọn đề tài

Ngành công nghiệp phần mềm Việt Nam đang dần phát triển với nhiều hứa hẹn trong tương lai. Kiểm thử là một phần rất quan trọng và không thể thiếu trong quy trình phát triển phần mềm. Với những đòi hỏi ngày càng cao về chất lượng và việc rút ngắn tối đa thời gian phát triển, kiểm thử ngày càng trở nên quan trọng và thậm chí nhiều khi là vấn đề sống còn để một dự án phát triển phần mềm.

Với nhận định của **Công ty kiểm thử phần mềm của Mỹ Logigea thì “ Việt Nam sẽ là con hổ trong ngành kiểm thử phần mềm Châu Á” và với yêu cầu phát triển như vậy, với nguồn nhân lực về Kiểm thử viên ở Việt Nam vẫn còn khá ít so với Lập trình viên thì những công cụ hỗ trợ Kiểm thử viên là rất cần thiết.**

**Các công cụ hỗ trợ kiểm thử tự động trên thị trường cũng có rất nhiều nhưng cũng có những hạn chế nhất định nào đó chưa thực sự tự động hoàn toàn trong quá trình kiểm thử mà vẫn cần rất nhiều công sức của kiểm thử viên. Vì vậy nhóm chúng em đã chọn đề tài “Tìm hiểu và xây dựng một ứng dụng kiểm thử phần mềm tự động” với mong muốn xây dựng một ứng dụng có thể đáp ứng và hỗ trợ cho kiểm thử viên thực hiện công việc một cách dễ dàng, chính xác và nhanh chóng hơn.**

## Mục đích nghiên cứu

* Đưa ra các khái niệm cơ bản của kiểm thử tự động.
* Đưa ra lợi ích của việc xây dựng một ứng dụng kiểm thử tự động.
* Nghiên cứu các kỹ thuật của .NET hỗ trợ xây dựng ứng dụng kiểm thử tự động.
* Thiết kế ứng dụng kiểm thử tự động phần mềm dựa vào kiến thức đã nghiên cứu.

## Khách thể và đối tượng nghiên cứu

* *Khách thể nghiên cứu:* Kiến thức về kiểm thử phần mềm tự động, quá trình xây dựn ứng dụng kiểm thử tự động.
* *Đối tượng nghiên cứu:* Tìm hiểu và xây dựng ứng dụng kiểm thử tự động phần mềm.

## Nhiệm vụ nghiên cứu

* Trình bày lý thuyết về kỹ thuật xây dựng ứng dụng kiểm thử tự động phần mềm.
* Phân tích – thiết kế hệ thống.
* Xây dựng ứng dụng kiểm thử tự động phần mềm.

## Phạm vi nghiên cứu

Nghiên cứu các nội dung bao gồm:

* Khái niệm kiểm thử phần mềm.
* Khái niệm kiểm thử tự động.
* Lợi ích của kiểm thử tự động và xây dựng ứng dụng kiểm thử tự động phần mềm.
* Kỹ thuật Reflection trong .NET.
* Kỹ thuật CodeDom trong .NET.
* Lưu trữ dữ liệu với XML và MS Excel.

## Phương pháp nghiên cứu

* Tìm kiếm, tham khảo các tài liệu liên quan đến kiểm thử phần mềm, kiểm thử tự động và các kỹ thuật xây dựng ứng dụng tự động; các trang forum, blog về kiểm thử phần mềm như *testingvn.com, yinyangit.wordpress.com , testervn.com* …
* Tìm hiểu cách lấy về thông tin của một assembly một cách tự động.
* Tìm hiểu cách xây dựng và thực thi tự động chương trình c#.

## Ý nghĩa lý luận và thực tiễn của đề tài

Đề tài được hoàn thành sẽ là nguồn tài liệu tham khảo cho những ai muốn quan tâm về kiểm thử và muốn xây dựng một ứng dụng kiểm thử với những chức năng như mong muốn. Về mặt ứng dụng sẽ cung cấp một ứng dụng kiểm thử tự động phần mềm đảm bảo yêu cầu test về mặt nào đó của phần mềm.

# PHẦN II: NỘI DUNG

# Chương 1: Giới thiệu tổng quan

## Khái niệm kiểm thử phần mềm:

Kiểm thử phần mềm là khâu mấu chốt để đảm bảo chất lượng phần mềm, là đánh giá cuối cùng về các đặc tả, thiết kế và mã hóa.

Kiểm thử phần mềm là quá trình chạy một ứng dụng để phát hiện lỗi và xem nó có thỏa mãn các yêu cầu đặt ra không. Trong quá trình phát triển phần mềm, những người phát triển phần mềm và các kỹ sư kiểm thử cùng làm việc để phát hiện lỗi và đảm bảo chất lượng sản phẩm. Một sản phẩm phần mềm được phân phối phải có đầy đủ các chức năng yêu cầu và tương thích với phần cứng của khách hàng.

Mục tiêu đầu tiên của kiểm thử là ngăn ngừa lỗi, ngăn ngừa lỗi còn tốt hơn là sửa lỗi vì ngăn ngừa được lỗi thì sẽ là tốt hơn và không phải sửa mã, giải quyết được vấn đề ngay từ đầu sẽ làm giảm bớt chi phí về thời gian và công sức sửa chữa hơn.

## Khái niệm kiểm thử tự động

Kiểm thử tự động là quá trình thực hiện một cách tự động các bước trong một testcase. Nó sử dụng một công cụ kiểm thử tự động nào đó để rút ngắn thời gian kiểm thử. Kiểm thử tự động hỗ trợ các kiểm thử viên rất nhiều tùy vào công cụ và các nội dung kiểm thử có thể thực hiện bằng tay hay không. Đối với những nhiệm vụ kiểm tra khó mà thực hiện bằng tay hoặc yêu cầu chi phí về nhân công là quá lớn thì sử dụng tool hỗ trợ là điều hết sức cần thiết.

## Mục tiêu của kiểm thử tự động

Phần mềm có khiếm khuyết là thông thường và gây ra thiệt hại về kinh tế theo thời gian. Chính vì vậy các tổ chức về phần mềm dành nhiều thời gian và nguồn lực để phân tích và kiểm thử phần mềm.

Ngày nay ứng dụng tự động hóa vào các ngành đa dạng, trong đó có ngành kiểm thử. Trước đây kiểm thử viên kiểm thử bằng tay thực hiện và ghi lại kết quả trên giấy, nhưng với ứng dụng công nghệ thông tin thì các công cụ kiểm thử cũng phát triển từ rất sớm hỗ trợ kiểm thử viên rất nhiều và đặc biệt là các trường hơpk kiểm thử đặc biệt mà kiểm thử bằng tay không thể thực hiện được hoặc rất khó khăn để thực hiện. Các tổ chức càng quan tâm đến chất lượng phần mềm thì càng có nhiều chức năng và nội dung được kiểm tra đòi hỏi kiểm thử viên phải thực hiện nhiều công việc hơn vì vậy kiểm thử với sự hỗ trợ của công cụ là rất cần thiết.

Để giúp các kiểm thử viên có thể kiểm thử tự động đó là các TestTool, tuy nhiên không phải trong bất cứ trường hợp nào cũng có thể kiểm thử tự động. Vậy kiểm thử thử tự động khi nào?

* Kiểm thử tự động trong các tình huống sau:
* Không đủ tài nguyên:

Khi số lượng tình huống kiểm tra (test case) quá nhiều mà các kiểm thử viên không thể hoàn tất bằng tay trong thời gian cụ thể nào đó.

Có thể lấy một dẫn chứng là khi thực hiện kiểm tra chức năng của một website. Website này sẽ được kiểm tra với 6 môi trường gồm 3 trình duyệt và 2 hệ điều hành. Tình huống này đòi hỏi số lần kiểm tra tăng lên và lặp lại 6 lần so với việc kiểm tra cho một môi trường cụ thể.

* Kiểm tra hồi quy:

Trong quá trình PTPM, nhóm lập trình thường đưa ra nhiều phiên bản phần mềm liên tiếp để kiểm tra. Thực tế cho thấy việc đưa ra các phiên bản phần mềm có thể là hàng ngày, mỗi phiên bản bao gồm những tính năng mới, hoặc tính năng cũ được sửa lỗi hay nâng cấp. Việc bổ sung hoặc sửa lỗi code cho những tính năng ở phiên bản mới có thể làm cho những tính năng khác đã kiểm tra tốt chạy sai mặc dù phần code của nó không hề chỉnh sửa. Để khắc phục điều này, đối với từng phiên bản, kiểm thử viên không chỉ kiểm tra chức năng mới hoặc được sửa, mà phải kiểm tra lại tất cả những tính năng đã kiểm tra tốt trước đó. Điều này khó khả thi về mặt thời gian nếu kiểm tra thủ công.

* Kiểm tra vận hành phần mềm trong môi trường đặc biệt:

Đây là kiểm tra nhằm đánh giá xem vận hành của phần mềm có thỏa mãn yêu cầu đặt ra hay không. Thông qua đó kiểm thử viên có thể xác định được các yếu tố về phần cứng, phần mềm ảnh hưởng đến khả năng vận hành của phần mềm. Có thể liệt kê một số tình huống kiểm tra tiêu biểu thuộc loại này như sau:

Đo tốc độ trung bình xử lý một yêu cầu của web server.

Thiết lập 1000 yêu cầu, đồng thời gửi đến web server, kiểm tra tình huống 1000 người dùng truy xuất web cùng lúc.

Xác định số yêu cầu tối đa được xử lý bởi web server hoặc xác định cấu hình máy thấp nhất mà tốc độ xử lý của phần mềm vẫn có thể hoạt động ở mức cho phép.

Việc kiểm tra thủ công cho những tình huống trên là cực khó, thậm chí "vô phương". Vì vậy kiểm thử tự động tăng độ tin cậy, chính xác, giảm bớt chi phí thực hiện kiểm thử cho các kiểm thử viên về thời gian, công sức; bên cạnh đó việc thiết kế test case cho mỗi lần kiểm thử giúp kiểm thử viên rèn luyện kỹ năng lập trình (viết test script), giảm sự nhàm chán khi cứ mãi thực hiện các kiểm thử thủ công.

Hoạt động kiểm thử tự động nhằm mục đích kiểm tra, phát hiện những lỗi của phần mềm trong những trường hợp đoán trước. Nghĩa là nó thường được thực hiện sau khi đã thiết kế xong các tình huống (test case). Tuy nhiên, như đã nói, không phải mọi trường hợp kiểm tra đều có thể hoặc cần thiết phải tự động hóa, trong tất cả test case thì kiểm thử viên phải đánh giá và chọn ra những test case nào phù hợp hoặc cần thiết để áp dụng kiểm thử tự động dựa trên những tiêu chí đã đề cập bên trên.

* Mục tiêu của kiểm thử tự động :
* Giảm bớt công sức và thời gian thực hiện
* Tăng độ tin cậy
* Giảm sự nhàm chán
* Giảm chi phí cho tổng quá trình kiểm thử
* Ưu điểm của kiểm thử tự động:
* KTPM không cần can thiệp của KTV.
* Giảm chi phí khi thực hiện kiểm tra số lượng lớn test case hoặc test case lặp lại nhiều lần.
* Giả lập tình huống khó có thể thực hiện bằng tay.

## Quy trình kiểm thử phần mềm tự động

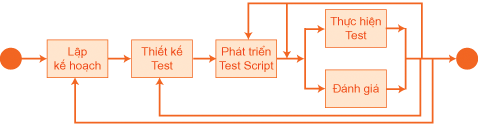
Một công cụ kiểm thử phần mềm tự động yêu cầu phải làm được những công việc sau:

* Hiểu các mã assembly được kiểm tra một cách tự động.
* Tiến hành các nhiệm vụ đơn giản và lặp đi lặp lại một cách tự động.
* Tạo ra các kịch bản và chạy các kịch bản trong những lô lệnh theo một lịch trình đã vạch ra.
* Kiểm tra giao diện của các đối tượng COM và các thành phần phần mềm khác với tập dữ liệu được thiết lập sẵn.
* Truy cập vào dữ liệu để xác minh lại các kết quả.
* Truy cập vào Regestry để xác minh lại các kết quả.
* Quá trình thực hiện kiểm thử thông thường được thực hiện bằng tay:

Sau khi lập kế hoạch, kiểm thử viên thiết kế các test case gồm dữ liệu đầu vào, dữ liệu đầu ra mong chờ và kết quả thực hiện (điền sau khi test). Tùy theo yêu cầu và phương pháp được chọn kiểm thử viên thực thi test bằng tay và ghi lại kết quả trên giấy cuối cùng đánh giá kết quả đó với kết quả mong chờ đã chuẩn bị trước đó. Với phương pháp kiểm thử bằng tay này chỉ sử dụng cho một số nội dung kiểm thử như kiểm thử giao diện, tài liệu hoặc test các class, phương thức đơn giản… còn với test về hiệu năng, khả năng chịu tải (stress/volume test), kiểm thử cấu hình… thì phương pháp này khó mà thực hiện được. Do vậy cần có công cụ kiểm thử tự động hỗ trợ thực hiện.

* Quy trình của kiểm thử tự động:

Quy trình kiểm thử tự động phần mềm cũng giống như quy trình thực hiện bằng tay chỉ khác ở chỗ kiểm thử tự động có hỗ trợ của công cụ ít hoặc nhiều như tạo script (có thể bằng tay hoặc công cụ), công cụ hỗ trợ về ghi lại kết quả và lưu trữ kết quả trong máy tính. Quy trình này cũng gần tương tự với quy trình phát triển phần mềm, được thực hiện qua nhiều bước, được tiến hành rất sớm trong quy trình phát triển phần mềm và đội kiểm thử tiến hành thư song song cùng đội phát triển phần mềm.

****

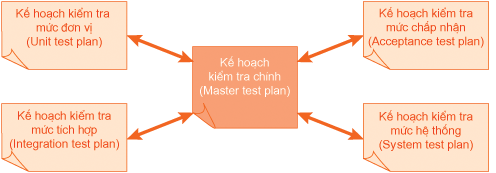
Hình 1.1: Quy trình của kiểm thử tự động[3].

* Lập kế hoạch kiểm tra:

Mục đích: Nhằm chỉ định và mô tả các loại kiểm tra sẽ được triển khai và thực hiện. Kết quả của bước lập kế hoạch là bản tài liệu kế hoạch KTPM, bao gồm nhiều chi tiết từ các loại kiểm tra, chiến lược kiểm tra, cho đến thời gian và phân định lực lượng kiểm tra viên.

Bản kế hoạch kiểm tra đầu tiên được phát triển rất sớm trong chu trình phát triển phần mềm (PTPM), ngay từ khi các yêu cầu đã tương đối đầy đủ, các chức năng và luồng dữ liệu chính đã được mô tả. Bản kế hoạch này có thể được coi là bản kế hoạch chính (master test plan), trong đó tất cả các kế hoạch chi tiết cho các mức kiểm tra và loại kiểm tra khác nhau đều được đề cập.

Sau khi bản kế hoạch chính được phát triển, các bản kế hoạch chi tiết lần lượt được thiết kế theo trình tự thời gian phát triển của dự án. (Hình 06 minh hoạ thời điểm phù hợp để thiết lập các kế hoạch kiểm tra, gắn liền với quá trình phát triển của dự án. Quá trình phát triển các kế hoạch kiểm tra không dừng lại tại một thời điểm, mà liên tục được cập nhật chỉnh sửa cho phù hợp đến tận cuối dự án.).

  
          
      Hình 1.2:  Bản kế hoạch chính và các bản kế hoạch chi tiết[3].

* Các bước lập kế hoạch:

Xác định yêu cầu kiểm tra: chỉ định bộ phận, thành phần của PM sẽ được kiểm tra, phạm vi hoặc giới hạn của việc kiểm tra. Yêu cầu kiểm tra cũng được dùng để xác định nhu cầu nhân lực.

Khảo sát rủi ro: Các rủi ro có khả năng xảy ra làm chậm hoặc cản trở quá trình cũng như chất lượng kiểm tra. Ví dụ: kỹ năng và kinh nghiệm của kiểm tra viên quá yếu, không hiểu rõ yêu cầu.

Xác định chiến lược kiểm tra: chỉ định phương pháp tiếp cận để thực hiện việc kiểm tra trên PM, chỉ định các kỹ thuật và công cụ hỗ trợ kiểm tra, chỉ định các phương pháp dùng để đánh giá chất lượng kiểm tra cũng như điều kiện để xác định thời gian kiểm tra.

Xác định nhân lực,vật lực: kỹ năng, kinh nghiệm của kiểm tra viên; phần cứng, phần mềm, công cụ, thiết bị giả lập… cần thiết cho việc kiểm tra.

Lập kế hoạch chi tiết: ước lượng thời gian, khối lượng công việc, xác định chi tiết các phần công việc, người thực hiện, thời gian tất cả các điểm mốc của quá trình kiểm tra.

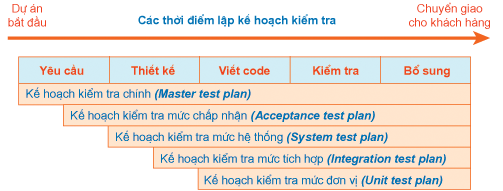
* Tổng hợp và tạo các bản kế hoạch kiểm tra: kế hoạch chung và kế hoạch chi tiết.

Xem xét các kế hoạch kiểm tra: phải có sự tham gia của tất cả những người có liên quan, kể cả trưởng dự án và có thể cả khách hàng. Việc xem xét nhằm bảo đảm các kế hoạch là khả thi, cũng như để phát hiện (và sữa chữa sau đó) các sai sót trong các bản kế hoạch.

* Thiết kế Test:

Mục đích: Nhằm chỉ định các Test Case và các bước kiểm tra chi tiết cho mỗi phiên bản PM. Giai đoạn thiết kế test là hết sức quan trọng, nó bảo đảm tất cả các tình huống kiểm tra “quét” hết tất cả yêu cầu cần kiểm tra.

Hình dưới cho thấy việc thiết kế test không phải chỉ làm một lần, nó sẽ được sửa chữa, cập nhật, thêm hoặc bớt xuyên suốt chu kỳ PTPM, vào bất cứ lúc nào có sự thay đổi yêu cầu, hoặc sau khi phân tích thấy cần được sửa chữa hoặc bổ sung.

  
         
                Hình 1.3. Thời điểm phù hợp để thiết lập các kế hoạch kiểm tra[3].

* Các bước thiết kế test bao gồm:

Xác định và mô tả Test Case: xác định các điều kiện cần thiết lập trước và trong lúc kiểm tra. Mô tả đối tượng hoặc dữ liệu đầu vào, mô tả các kết quả mong chờ sau khi kiểm tra.

Mô tả các bước chi tiết để kiểm tra: các bước này mô tả chi tiết để hoàn thành một Test Case khi thực hiện kiểm tra. Các Test Case như đã nói ở trên thường chỉ mô tả đầu vào, đầu ra, còn cách thức tiến hành như thế nào thì không được định nghĩa. Thao tác này nhằm chi tiết hóa các bước của một Test Case, cũng như chỉ định các loại dữ liệu nào cần có để thực thi các Test Case, chúng bao gồm các loại dữ liệu trực tiếp, gián tiếp, trung gian, hệ thống…

Xem xét và khảo sát độ bao phủ của việc kiểm tra: mô tả các chỉ số và cách thức xác định việc kiểm tra đã hoàn thành hay chưa? bao nhiêu phần trăm PM đã được kiểm tra? Để xác định điều này có hai phương pháp: căn cứ trên yêu cầu của phần mềm hoặc căn cứ trên số lượng code đã viết.

Xem xét Test Case và các bước kiểm tra: Việc xem xét cần có sự tham gia của tất cả những người có liên quan, kể cả trưởng dự án nhằm bảo đảm các Test Case và dữ liệu yêu cầu là đủ và phản ánh đúng các yêu cầu cần kiểm tra, độ bao phủ đạt yêu cầu, cũng như để phát hiện (và sữa chữa) các sai sót.

* Phát triển Test Script

Mục đích: Bước này thường không bắt buộc trong các loại và mức kiểm tra, chỉ yêu cầu trong những trường hợp đặc thù cần thiết kế, tạo ra các Test Script có khả năng chạy trên máy tính giúp tự động hóa việc thực thi các bước kiểm tra đã định nghĩa ở bước thiết kế test.

* Các bước phát triển Test Script bao gồm:

Tạo Test Script: thủ công hoặc dùng công cụ hỗ trợ để phát sinh script một cách tự động (tuy nhiên trong hầu hết mọi trường hợp, ta vẫn phải chỉnh sửa ít hoặc nhiều trên các script được sinh tự động). Thông thường, mỗi bước kiểm tra được thiết kế trong phần thiết kế test, đòi hỏi ít nhất một Test Script. Các Test Script có khả năng tái sử dụng càng nhiều càng tốt để tối ưu hóa công việc.

Kiểm tra Test script: xem có “chạy” tốt không nhằm bảo đảm các Test Script hoạt động đúng yêu cầu, thể hiện đúng ý đồ của các bước kiểm tra.

Thành lập các bộ dữ liệu ngoài dành cho các Test Script: bộ dữ liệu này sẽ được các Test Script sử dụng khi thực hiện kiểm tra tự động. Gọi là “ngoài” vì chúng được lưu độc lập với các Test Script, tránh trường hợp vì dễ dãi, một số kiểm tra viên “tích hợp” luôn phần dữ liệu vào bên trong code của các script (thuật ngữ chuyên môn gọi là “hard-code”). Việc tách riêng dữ liệu cho phép dễ dàng thay đổi dữ liệu khi kiểm tra, cũng như giúp việc chỉnh sửa hoặc tái sử dụng các script sau này.

Xem xét và khảo sát độ bao phủ của việc kiểm tra: bảo đảm các Test Script được tạo ra bao phủ toàn bộ các bước kiểm tra theo yêu cầu.

* Thực hiện kiểm tra:

Mục đích: Thực hiện các bước kiểm tra đã thiết kế (hoặc thi hành các Test Script nếu tiến hành kiểm tra tự động) và ghi nhận kết quả.

Việc thực hiện kiểm tra cũng được làm rất nhiều lần trong suốt chu trình kiểm tra, cho đến khi kết quả kiểm tra cho thấy đủ điều kiện để dừng hoặc tạm dừng việc thực hiện.

* Quá trình thực hiện kiểm tra thường thông qua các bước sau:

Thực hiện các bước kiểm tra: thủ công hoặc thi hành các Test Script nếu là quy trình kiểm tra tự động. Để thực hiện kiểm tra, thao tác đầu tiên cần làm là xác lập và khởi động môi trường và điều kiện kiểm tra. Việc này nhằm bảo đảm tất cả các bộ phận liên quan (như phần cứng, phần mềm, máy chủ, mạng, dữ liệu…) đã được cài đặt và sẵn sàng, trước khi chính thức bắt đầu thực hiện kiểm tra.

Đánh giá quá trình kiểm tra: giám sát quá trình kiểm tra suôn sẻ đến khi hoàn thành hay bị treo và dừng giữa chừng, có cần bổ sung hay sữa chữa gì không để quá trình kiểm tra được tốt hơn.

- Nếu quá trình diễn ra trơn tru, kiểm tra viên hoàn thành chu kỳ kiểm tra và chuyển qua bước “Thẩm định kết quả kiểm tra”

- Nếu quá trình bị treo hoặc dừng giữa chừng, kiểm tra viên cần phân tích để xác định nguyên nhân lỗi, khắc phục lỗi và lập lại quá trình kiểm tra.

Thẩm định kết quả kiểm tra: sau khi kết thúc, kết quả kiểm tra cần được xem xét để bảo đảm kết quả nhận được là đáng tin cậy, cũng như nhận biết được những lỗi xảy ra không phải do PM mà do dữ liệu dùng để kiểm tra, môi trường kiểm tra hoặc các bước kiểm tra (hoặc Test Script) gây ra. Nếu thực sự lỗi xảy ra do quá trình kiểm tra, cần phải sửa chữa và kiểm tra lại từ đầu.

* Đánh giá quá trình kiểm tra:

Mục đích: Đánh giá toàn bộ quá trình kiểm tra, bao gồm xem xét và đánh giá kết quả kiểm tra, liệt kê lỗi, chỉ định các yêu cầu thay đổi, và tính toán các số liệu liên quan đến quá trình kiểm tra (chẳng hạn số giờ, thời gian kiểm tra, số lượng lỗi, phân loại lỗi…).

Mục đích của việc đánh giá kết quả kiểm tra ở bước này hoàn toàn khác với bước thẩm định kết quả kiểm tra sau khi hoàn tất một vòng kiểm tra. Đánh giá kết quả kiểm tra ở giai đoạn này mang tính toàn cục và nhằm vào bản thân giá trị của các kết quả kiểm tra.

Việc đánh giá quá trình và kết quả kiểm tra được thực hiện song song với bất kỳ lần kiểm tra nào và chỉ chấm dứt khi quá trình kiểm tra đã hoàn tất.

* Đánh giá quá trình kiểm tra thường thông qua các bước sau:

• Phân tích kết quả kiểm tra và đề xuất yêu cầu sửa chữa: Chỉ định và đánh giá sự khác biệt giữa kết quả mong chờ và kết quả kiểm tra thực tế, tổng hợp và gửi thông tin yêu cầu sửa chữa đến những người có trách nhiệm trong dự án, lưu trữ để kiểm tra sau đó.

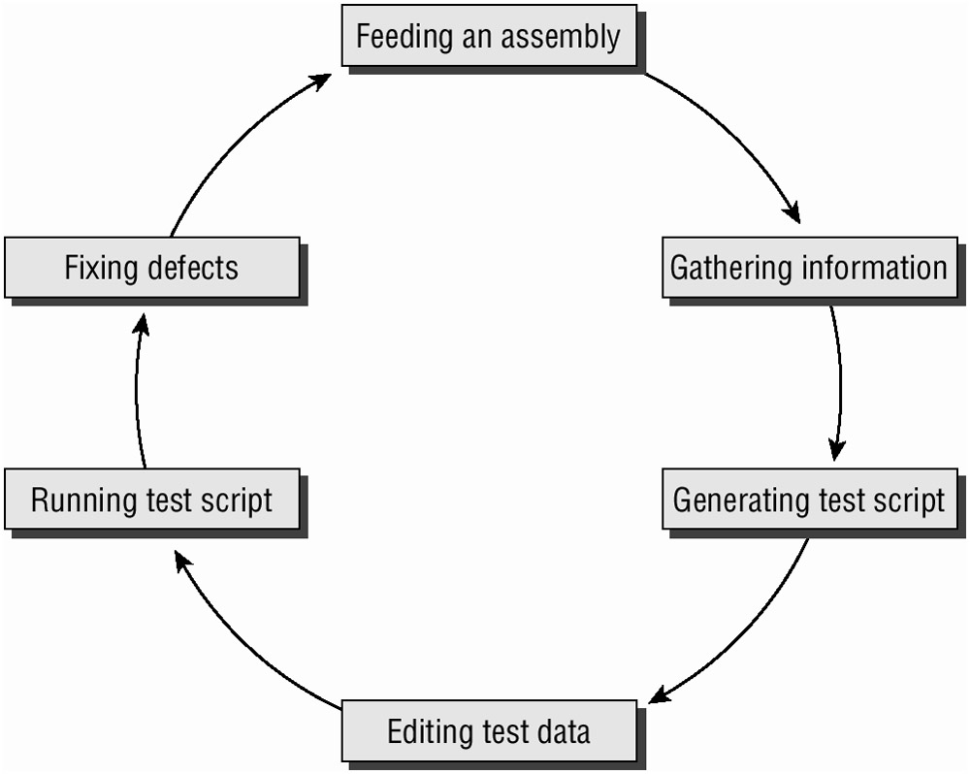
• Đánh giá độ bao phủ: Xác định quá trình kiểm tra có đạt được độ bao phủ yêu cầu hay không, tỷ lệ yêu cầu đã được kiểm tra (tính trên các yêu cầu của PM và số lượng code đã viết).

• Phân tích lỗi: Đưa ra số liệu phục vụ cho việc cải tiến các qui trình phát triển, giảm sai sót cho các chu kỳ phát triển và kiểm tra sau đó. Ví dụ, tính toán tỷ lệ phát sinh lỗi, xu hướng gây ra lỗi, những lỗi “ngoan cố” hoặc thường xuyên tái xuất hiện.

• Xác định quá trình kiểm tra có đạt yêu cầu hay không: Phân tích đánh giá để xem các Test Case và chiến lược kiểm tra đã thiết kế có bao phủ hết những điểm cần kiểm tra hay không? Kiểm tra có đạt yêu cầu dự án không? Từ những kết quả này, kiểm tra viên có thể sẽ phải thay đổi chiến lược hoặc cách thức kiểm tra.

• Báo cáo tổng hợp: Tổng hợp kết quả các bước ở trên và phải được gửi cho tất cả những người có liên quan.

* Quy trình thực hiện kiểm thử tự động của một Tool tự động:



Hình 1.4: Quy trình thực hiện kiểm thử của công cụ tự động.

* Quy trình được thực hiện theo vòng tròn, lặp đi lặp lại, các bước thực hiện:
* Cung cấp mã assembly.
* Thu gom thông tin.
* Sinh ra các test script.
* Chỉnh sửa các dữ liệu test.
* Chạy test script.
* Sửa lỗi.

## Các công cụ kiểm thử tự động trên thị trường

### QuickTest Professinal:

QuickTest Professional (QTP) phiên bản 8.2 của hãng Mercury khá tốt và mạnh, bao gồm nhiều chức năng điển hình của một công cụ kiểm tra tự động. QTP 8.2 đã có một cái tên mới hơn là Mercury Functional Testing 8.2. QTP là TT dùng để kiểm tra chức năng (functional test) và cho phép thực hiện kiểm tra hồi qui (regression test) một cách tự động. Đây cũng là công cụ áp dụng phương pháp Keyword-Driven, một kỹ thuật scripting (lập trình trong KTTĐ) hiện đại, cho phép KTV bổ sung test case bằng cách tạo file mô tả cho nó mà không cần phải chỉnh sửa hay bổ sung bất cứ script nào cả. Nó cũng phù hợp trong tình huống chuyển giao công việc mà người mới tiếp nhận chưa có thời gian hoặc không hiểu script vẫn có thể thực hiện kiểm tra PM theo đúng yêu cầu.

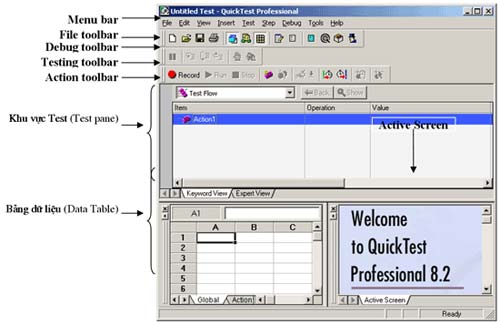
* **Loại phần mềm hỗ trợ**

QTP giúp chúng ta KTPM theo hướng chức năng trên rất nhiều loại chương trình phần mềm khác nhau. Tuy nhiên Mercury chỉ hỗ trợ sẵn một số loại chương trình thông dụng như:

• Ứng dụng Windows chuẩn/Win32.   
• Ứng dụng web theo chuẩn HTML, XML chạy trong trình duyệt Internet Explorer, Netscape hoặc AOL.   
• Visual Basic.   
• ActiveX.   
• QTP hỗ trợ Unicode (UTF-8, UTF-16).

* **Đặc điểm**

• Dễ sử dụng, bảo trì, tạo test script nhanh. Cung cấp dữ liệu kiểm tra rõ ràng và dễ hiểu.   
• Kiểm tra phiên bản mới của ứng dụng với rất ít sự thay đổi. Ví dụ khi ứng dụng thay đổi nút tên "Login" thành "Đăng nhập", thì chỉ cần cập nhật lại Object Repository (OR - được giải thích ở phần sau) để QTP nhận ra sự thay đổi đó mà không cần thay đổi bất cứ test script nào.   
• Hỗ trợ làm việc theo nhóm thông qua sự chia sẻ thư viện, thống nhất quản lý Object Repository.   
• Thực tế cho thấy, QTP thực hiện KTTĐ trên nhiều trình duyệt cùng lúc tốt hơn những TT khác.   
• Với chức năng Recovery Scenarios, QTP cho phép xử lý những sự kiện hoặc lỗi không thể đoán trước có thể làm script bị dừng trong khi đang chạy.   
• QTP có khả năng hiểu test script của Mercury Winrunner (một công cụ kiểm tra khác của Mercury).



Hình 1.5: Giao diện QTP

|  |  |
| --- | --- |
| **Khu vực** | **Chức năng** |
| Menu bar | Cấu hình thao tác với QTP và script |
| File toolbar | Hỗ trợ quản lý script |
| Debug toolbar | Hỗ trợ kiểm tra lỗi trong test script (debug) |
| Testing toolbar | Hỗ trợ quá trình tạo test script hoặc thực hiện KTTĐ |
| Action toolbar | Xem một Action (thủ tục, hàm) hoặc toàn bộ chu trình của test script. |
| Test pane | Soạn thảo script ở một trong 2 chế độ Keyword View hoặc Expert View. |
| Data Table | Nơi lưu trữ dữ liệu cho test script |
| Active Screen | Xem lại giao diện PM được kiểm tra |

Bảng 1.1: Các chức năng trên giao diện chính của QTP.

* **Các thành phần quan trọng trong QTP**

1. **Action**:

Giống như thủ tục hay hàm trong các ngôn ngữ lập trình khác, Action ghi lại các bước thực hiện KTTĐ và nó có thể được sử dụng lại nhiều lần. Trong một test script có thể có nhiều Action.  
**b. DataTable**:

Nơi lưu dữ liệu phục vụ cho KTTĐ. Một test script sẽ có một DataTable được dùng chung cho tất cả các Action. Bên cạnh đó mỗi Action cũng có một DataTable cho riêng mình.

**c. Object Repository (OR)**:

Cấu trúc theo dạng cây, mô tả các đối tượng trong PM được kiểm tra. Đây được xem là cầu nối để test script tương tác với PM được kiểm tra.

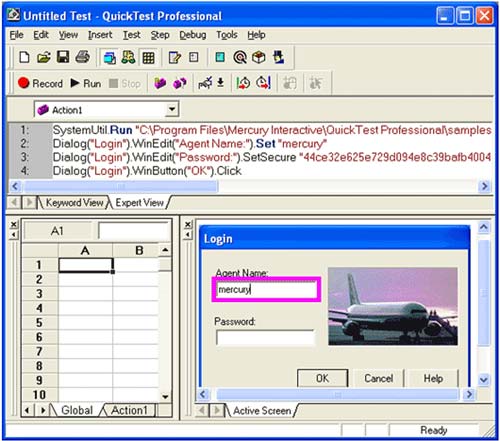
Khi ra lệnh cho QTP ghi lại thao tác người dùng lên PM thì trong OR sẽ tự động phát sinh thành phần đại diện cho những đối tượng trên PM vừa được thao tác. OR có thể tổ chức thành 2 loại, một loại dùng chung trong nhiều test script, loại khác dùng theo từng Action. Để xem OR, chọn menu Tools > Object Repository.   
**d. Checkpoint**:

Có thể hiểu là nơi kiểm tra trong test script, khi chạy nó sẽ thực hiện so sánh kết quả thực tế khi kiểm tra PM với kết quả mong đợi. Sau khi tiến hành so sánh QTP sẽ tự động ghi lại kết quả vào Test Results (nơi lưu kết quả khi chạy test script).

* **Ngôn ngữ sử dụng viết script**

QTP sử dụng ngôn ngữ VBScript để viết test script. Đây là ngôn ngữ dễ học; rất giống ngôn ngữ VBA. Chế độ Expert View của QTP là chế độ soạn thảo dành cho VBScript. Ngoài việc dùng VBScript để tương tác với PM được kiểm tra, QTP còn có khả năng cấu hình hệ thống bằng ngôn ngữ Windows Script.

Chi tiết về ngôn ngữ VBScript, người đọc có thể dễ dàng tìm trong các sách hiện có trên thị trường, thậm chí ngay chính trong phần help của QTP.



Hình 1.6: Giao diện viết test script tự động trong QTP

QTP hỗ trợ việc sử dụng các cấu trúc lớp và hàm để quản lý các Test Case

*Class NameClass*

*----------------------*

*Public sub Run()*

*End sub*

---------------------- Constructor

*Private Sub Class\_Initialize*

*End sub*

---------------------- Constructor

*Private Sub Class\_Terminate*

*End sub*

*----------------------*

*End class*

Sử dụng RegisterUserFunc để đăng ký hàm với QTP, tạo ra các thư viện hàm để có thể sử dụng lại trong các dự án khác.

### DevPartner Studio của công ty Compuware:

Compuware phát triển công cụ kiểm thử tự động phát hiện, chuẩn đoán và tạo điều kiện thuận lợi để giải quyết các lỗi phần mềm liên quan đến hiệu suất của phần mềm. NuMega DevPartner Studio 6.1 bao gồm các thành phần:

BoundsChecker: kiểm thử cho việc phát hiện rò rỉ bộ nhớ và tài nguyên của ứng dụng Visual C++

JCheck: sử dụng để phân tích tuyến và sự kiện trong các ứng dụng Java.

TrueTime: cung cấp các phân tích hiệu suất của các ứng dụng

* Ngoài ra còn có các thành phần như: SoftIce, CodeReview, CodeReview, SmartCheck…

### Mercury Interactive

Mercury Interactive cung cấp các sản phẩm phần mềm để kiểm tra các ứng dụng Internet. Bộ công cụ bao gồm các thành phần:

TestDirector: công cụ này cho phép lập kế hoạch kiểm tra, thực hiện và theo dõi các chức năng có khiếm khuyết. Nó có thể sử dụng với một loạt các hệ thống quản lý cơ sở dữ liệu như: Oracle, Sybase, MS SQL Server, and MS Access. TestDiector tích hợp với công cụ chụp/xem lại của Mercury là WinRunner và công cụ kiểm tra khả năng chịu đưng áp lực về mặt hiệu là LoadRunner. Đây là một công cụ dựa trên nền Windows.

WinRunner, XRunner.

LoadRunner: là công cụ kiểm thử về stress/volume cho cả 2 môi trường Windows và Unix, tích hợp với WinRunner, TestDirector và XRunner. Công cụ này có thể mô phỏng hàng ngàn người sử dụng và có thể tạo ra các kịch bản tiêu chuẩn hiệu suất của các thành phần của các nhà cung cấp khác nhau chẳng hạn như máy chủ, cơ sở dữ liệu và các thành phần mạng. Nó cũng cung cấp báo cáo chi tiết xác định vị trí tắc nghẽn hệ thống.

### Phần mềm Rational từ IBM

Gần đây, Rational Software đã trở thành một chi nhánh của IBM. Nó là một công cụ vòng đời đầy đủ cái mà cung cấp công cụ hỗ trợ yêu cầu quản lý, mô hình hóa hình ảnh, thử nghiệm, cấu hình, và kiểm soát thay đổi. Hãng đã mua được sản phẩm của các nhà cung cấp khác nhau trong những năm qua, chẳng hạn như VisualTest, SQA, Pure Atria, và nâng cao nhận thức Hiệu suất.

Rational Software đã phát triển một gói các công cụ kiểm tra của mình với một tên gọi chung của SQA Suite. Gói bao gồm các thành phần : Rational Test RealTime, SQA Robot, SQA Repository, SQA WebEntry, SQA Manager, SQA LoadTest, Visual Test…

### Một số công cụ kiểm thử mã nguồn mở:

Ngoài các phần mềm nêu trên còn có các công cụ mã nguồn mở dành cho các nhà phát triển trên ngôn ngữ Java như JUnit, Ant, JProbe Suite, HttpUnit… cũng được sử dụng rộng rãi đối với những phần mềm phù hợp với chúng.

# Chương 2: Namespace System.Reflection

## 2.1. Các khái niệm cơ bản:

### 2.1.1. Reflection là gì?

Reflection được hiểu là một chức năng trong .Net cho phép đọc thông tin từ các siêu dữ liệu (metadata) của assembly để tạo ra một đối tượng (có kiểu là Type) bao gói các thông tin đó lại. Với reflection, bạn có thể trích xuất để gọi và tạo ra các phương thức, truy cập và thay đổi các thuộc tính của đối tượng một cách linh động trong quá trình runtime.

Không gian tên .NET Reflection được dùng để khám phá ra kiểu thực thi, nó giống như một lăng kính phản chiếu ánh sáng mặt trời trong một quang phổ ánh sáng. Nó cung cấp các lớp và phương thức để phân biệt một assembly thành các kiểu và các thành viên. Vì vậy nó được dùng cho dự án xây dựng công cụ kiểm thử tự động. Reflection cho phép đọc thông tin từ các siêu dữ liệu (metadata) của assembly để tạo ra một đối tượng (có kiểu là Type) bao gói các thông tin đó lại. Với reflection, bạn có thể trích xuất để gọi và tạo ra các phương thức, truy cập và thay đổi các thuộc tính của đối tượng một cách linh động trong quá trình runtime.

Không gian tên này trợ giúp để tải một assembly tại lúc chạy và khám phá ra các thông tin trong một file .exe. Sau đó nó có được một danh sách của tất cả các loại kiểu chứa trong một module nào đó bao gồm các phương thức, trường, thuộc tính và sự kiện được định nghĩa. Nó cũng có thể lập trình để khám phá ra một tập các giao diện được hỗ trợ bởi lớp (hoặc cấu trúc), các tham số của phương thức cũng như các chi tiết liên quan khác ( như lớp cơ sở, thông tin tên miền, …).

Để có được thông tin từ một assembly, không gian tên System.Reflection thường sử dụng cùng với các lớp của System.Type. Lớp này có chứa một số phương thức có khả năng lấy về thông tin của kiểu mà bạn đang test , nó cũng chứa nhiều kiểu liên quan để tạo điều kiện để tải ràng buộc cuối và động của mã assembly.

### 2.1.2. Lớp System.Type

Lớp [System.Type](http://msdn.microsoft.com/en-us/library/system.type.aspx) là một abstract class đại diện cho các kiểu dữ liệu (kiểu lớp, interface, mảng, giá trị,…), đây là lớp chính để thực hiện các cơ chế Reflection đại diện cho các kiểu dữ liệu trong .Net. Bằng cách sử dụng lớp này, bạn có thể lấy về tất cả thông tin về các kiểu dữ liệu, phương thức, thuộc tính, sự kiện,… Ngoài ra ta còn có thể tạo ra các instance của kiểu dữ liệu và thực thi các phương thức của chúng, kĩ thuật này còn được gọi với thuật ngữ Late Binding.

Có rất nhiều mục được định nghĩa trong System.Reflection sử dụng lớp trừu tượng System.Type gồm một số phương thức được dùng để lấy về kiểu, thông tin của một đối tượng. Mục đích của việc lấy về này là để gom thông tin cần thiết cho việc kiểm thử. Ví dụ như test unit là một phương thức nào đó, có đối số, ta dùng tên miền này để thao tác lấy về tên phương thức, đối số, kiểu trả về nếu có. Từ đó phụ vụ cho quá trình thực thi test script sau này trong việc lấy thông tin để kiểm tra và so sánh với test case đã chuẩn bị trước đó. Trong System.Type một tập rất nhiều các thành phần, dưới đây là một số thuộc tính và phương thức sử dụng thường xuyên nhất:

Sau đây là các thuộc tính của lớp System.Type để tìm ra số các thuộc tính cơ bản về kiểu được chú ý (ví dụ: mỗi thuộc tính trong số chúng trả ra một giá trị True hoặc false dựa trên loại là một lớp trừu tượng, một mảng, một lớp …):

IsAbstract

IsArray

IsClass

IsCOMObject

IsEnum

IsInterface

IsPrimitive

IsNestedPublic

IsNestedPrivate

* Phương thức GetType() và toán tử typeof

Bạn có thể dùng phương thức GetType() của lớp Object để trả về đối tượng kiểu Type mô tả kiểu dữ liệu của đối tượng. Có được đối tượng Type này rồi, ta sẽ lấy thông tin của kiểu dữ liệu qua các thuộc tính của lớp Type.

Sau đây là một ví dụ đơn giản minh họa cách in ra tên kiểu dữ liệu của các biến qua thuộc tính FullName của lớp Type:

[*static*](http://social.msdn.microsoft.com/Search/en-US?query=static&ref=congdongcviet.com)[*void*](http://social.msdn.microsoft.com/Search/en-US?query=void&ref=congdongcviet.com)*Main(*[*string*](http://social.msdn.microsoft.com/Search/en-US?query=string&ref=congdongcviet.com)*[] args)  
{*[*var*](http://social.msdn.microsoft.com/Search/en-US?query=var&ref=congdongcviet.com)*number = 100;*[*var*](http://social.msdn.microsoft.com/Search/en-US?query=var&ref=congdongcviet.com)*text = "abc";  
    Console.WriteLine(number.GetType().FullName);  
    Console.WriteLine(text.GetType().FullName);  
    Console.Read();  
}*

Phương thức GetType() chỉ có thể lấy được thông tin từ các biến đối tượng. Trong trường hợp muốn lấy thông tin của lớp thông qua tên lớp, bạn phải sử dụng phương thức tĩnh Type.GetType(), tuy nhiên tham số truyền vào cần phải ghi đầy đủ cả namespace.

[*static*](http://social.msdn.microsoft.com/Search/en-US?query=static&ref=congdongcviet.com)[*void*](http://social.msdn.microsoft.com/Search/en-US?query=void&ref=congdongcviet.com)*Main()  
{  
    // Không được ghi tham số là string, int hoặc Int32  
    Type mType1 = Type.GetType("System.Int32");  
    Type mType2 = Type.GetType("System.String");  
  
    Console.WriteLine(mType1.FullName);  
    Console.WriteLine(mType2.FullName);  
    Console.Read();  
}*

Sử dụng toán tử typeof, bạn có thể lấy về đối tượng kiểu System.Type của bất kì kiểu nào với cú pháp typeof(type).

[*static*](http://social.msdn.microsoft.com/Search/en-US?query=static&ref=congdongcviet.com)[*void*](http://social.msdn.microsoft.com/Search/en-US?query=void&ref=congdongcviet.com)*Main(*[*string*](http://social.msdn.microsoft.com/Search/en-US?query=string&ref=congdongcviet.com)*[] args)  
{  
     
    Console.WriteLine(*[*typeof*](http://social.msdn.microsoft.com/Search/en-US?query=typeof&ref=congdongcviet.com)*(*[*Int32*](http://social.msdn.microsoft.com/Search/en-US?query=int32&ref=congdongcviet.com)*).FullName);  
    Console.WriteLine(*[*typeof*](http://social.msdn.microsoft.com/Search/en-US?query=typeof&ref=congdongcviet.com)*(*[*String*](http://social.msdn.microsoft.com/Search/en-US?query=string&ref=congdongcviet.com)*).FullName);  
    Console.Read();  
}*

Cả ba ví dụ trên đều cho ra cùng kết quả khi được thực thi:

System.Int32

System.String

* Lấy các thông tin từ Type

Hầu hết cá phương thức của System.Type được sử dụng để chứa chi tiết các thành viên của kiểu dữ liệu tương ứng – hàm, thuộc tính, phương thức, sự kiện… có nhiều phương thức nhưng tất cả chúng đều theo nền chung. Ví dụ, có hai phương thức mà nhận chi tiết phương thức của kiểu dữ liệu: getmethod() và getmethods() trả về 1 tham chiếu đến đối tượng System.Reflection.MethodInfo mà chứa chi tiết của 1 phương thức. Getmethods() trả về 1 mảng tham chiếu trong khi getmethod() trả về chi tiết của 1 phương thức được chỉ định trong danh sách thông số.

Sau đây là các phương thức trả về một mảng đại diện cho các mục (như giao diện, phương thức, thuộc tính,…) đáng quan tâm:

GetConstructors()

GetEvents()

GetFields()

GetInterfaces()

GetMethods()

GetMembers()

GetNestedTypes()

GetProperties()

Mỗi phương thức này trả về một mảng liên quan ( ví dụ: GetFields() trả về một mảng FieldInfo, GetMethods() trả về một mảng MethodInfo,…) . Các phương thức này có hình thức số ít như GetField(), GetMethod(), GetProperty() để lấy về một mục cụ thể bằng tên hơn là một mảng của tất cả các mục liên quan.

* Cuối cùng là:

FindMembers() trả ra một mảng các kiểu MemberInfo đã được lọc dựa trên cơ sở tìm kiếm nào đó.

GetType() trả ra một đối tượng kiểu cho một tên chuỗi

InvokeMember() cho phép ràng buộc cho một mục được đưa ra.

### 2.1.3. Không gian tên System.Reflection

Như đã nói không gian tên System.Reflection sử dụng để thu thập thông tin trong một assembly, nó định nghĩa một số các lớp kiểu liên quan. Tương tự như một số không gian tên khác, nó gồm nhiều item nhưng một số đáng chú ý hơn là:

Bảng sau liệt kê một số item quan trọng:

|  |  |
| --- | --- |
| **System.Reflection** | **Ý nghĩa** |
| Assembly | Lớp này (ngoài ra còn nhiều kiểu liên quan) có chứa một số phương thức cho phép bạn tải, tìm hiểu và thao tác với một assembly. |
| AssemblyName | Lớp này cho phép bạn tìm ra nhiều chi tiết đằng sau một đặc tính của assembly (thông tin phiên bản, thông tin mở rộng,…) |
| EventInfo | Lớp này nắm bắt thông tin một sự kiện được đưa ra. |
| FieldInfo | Lớp này nắm bắt thông tin một sự trường được đưa ra. |
| MemberInfo | Đây là lớp cơ sở trừu trượng xác định hành vi thông thường của EventInfo, FieldInfo, MethodInfo và PropertyInfo types. |
| MethodInfo | Lớp này chứa thông tin cho một phương thức được đưa ra. |
| Module | Lớp này cho phép bạn truy cập một module trong một assembly đa tệp. |
| ParameterInfo | Lớp này nắm giữ thông tin của một tham số được đưa ra. |
| PropertyInfo | Lớp này nắm giữ thông tin của một thuộc tính được đưa ra. |

Bảng 2.1: Bảng liệt kê một số item quan trọng của Namespace System.Reflction.

* Lớp System.Reflection.Assembly

.Net Assembly có thể được hiểu là kết quả của quá trình biên dịch từ mã nguồn sang tập tin nhị phân dựa trên .Net framework. Là thành phần cơ bản nhất .Net framework, assemly là thành phần không thể thiếu trong bất kì ứng dụng .Net nào và được thể hiện dưới hai dạng tập tin là EXE (process assembly) và DLL (library assembly). Assembly có thể được lưu trữ dưới dạng single-file hoặc multi-file, tùy theo kiểu dự án mà bạn làm việc.

Lớp System.Reflection.Assembly có thể được coi là một kiểu mô phỏng chi tiết về các assembly. Lớp Assembly chứa đầy đủ các thông tin cho phép chúng ta truy xuất thông tin và thực thi các phương thức lấy được từ các assembly. Có thể hiểu một cách tương tự: Lớp Type đại diện cho các kiểu dữ liệu, lớp Assembly đại diện cho các assembly. Lớp Assembly cung cấp các phương thức tĩnh để nạp một assembly thông qua [AssemblyName](http://msdn.microsoft.com/en-us/library/system.reflection.assemblyname.aspx), đường dẫn tập tin hoặc từ tiến trình đang chạy. Bạn cũng có thể lấy Assembly dễ dàng từ property của lớp Type, ví dụ typeof(int).Assembly.  
Từ Assemly lấy được, bạn có thể dùng phương thức GetTypes() lấy về các kiểu dữ liệu và thực hiện các kĩ thuật giống như trong phần về lớp System.Type tôi đã nói tới.  
Sau đây là một ví dụ đơn giản sử dụng phương thức Assembly.GetExecutingAssembly() để lấy về assembly hiện tại và in ra màn hình các kiểu dữ liệu của nó:

[*class*](http://social.msdn.microsoft.com/Search/en-US?query=class&ref=congdongcviet.com)*Program  
{*[*class*](http://social.msdn.microsoft.com/Search/en-US?query=class&ref=congdongcviet.com)*MyClass { }*

[*static*](http://social.msdn.microsoft.com/Search/en-US?query=static&ref=congdongcviet.com)[*void*](http://social.msdn.microsoft.com/Search/en-US?query=void&ref=congdongcviet.com)*Main()  
    {  
        Assembly ass = Assembly.GetExecutingAssembly();  
        Type[] mTypes = ass.GetTypes();  
        Array.*[*ForEach*](http://social.msdn.microsoft.com/Search/en-US?query=foreach&ref=congdongcviet.com)*(mTypes,type => Console.WriteLine(type.Name));  
        Console.Read();  
    }  
}*

Kết quả xuất ra:

Program

MyClass

# Chương 3: Namespace System.CodeDom

## 3.1. Các khái niệm cơ bản:

### 3.1.1. CodeDom là gì?

CodeDom nằm trong Model Object Document Code, nó có thể tự động sinh ra code một cách tự động cho người sử dụng công cụ. Do vậy, nó có thể cung cấp cho kỹ sư kiểm thử một công cụ mới có thể được dùng để sinh ra test script tự động. Không gian tên này là cốt lỗi để xây dựng một ứng dụng kiểm thử tự động, cung cấp rất nhiều kiểu giúp thực hiện tự động sinh ra test script cho công cụ.

### 3.1.2. Mô hình và các khái niệm cơ bản được sử dụng trong CodeDom:

CompileUnit

Namespace

NamespaceImport

Type

Property

Method

Field

Statement

Get/Set Statement

Hình 3.1: Mô hình các khái niệm cơ bản trong CodeDom.

Trong đó:

1. **CompileUnit**: Đơn vị chứa toàn bộ CodeDom với một collection các Namespace.
2. **Namespace**: Tên của namespace
3. **NamespaceImport**: Các namespace được sử dụng (bằng chỉ thị using trong C#)
4. **Type:** class, structure, interface, enumeration
5. **Property, Method, Field**: các thành viên của lớp
6. **Statement/ Expression**: câu lệnh/ biểu thức trong method hoặc property.

## 3.2. Các lớp trong namespace System.CodeDom

Tên miền này cung cấp đầy đủ các kiểu để lập trình viên có thể xây dựng chương trình có thể sinh ra một chương trình khác hoàn hảo với đầy đủ tên miền, các phương thức, điểm vào để chạy chương trình, các thuộc tính,…

Các kiểu được cung cấp bởi CodeDom để xây dựng namespace cho một

chương trình C#:

Không gian tên CodeDom cũng cung cấp một số kiểu để giúp ta xây dựng các thành phần cơ bản của một chương trình, như các kiểu sau đây để khai báo tên các namespace:

|  |  |
| --- | --- |
| **Tên kiểu** | **Mô tả** |
| CodeNamespace | Xây dựng một khai báo cho một namespace |
| CodeNamespaceCollection | Xây dựng một tập các namespace |
| CodeNamespaceImport | Xây dựng một trạng thái vào của một namespace |
| CodeNamespaceImportCollection | Xây dựng một trạng thái vào của một tập các namespace |

Bảng 3.1: Thành phần chính trong Namespace System.CodeDom.

Các kiểu của CodeDom để khai báo kiểu cấu trúc:

|  |  |
| --- | --- |
| **CodeDom Type để xây dựng các kiểu** | **Mô tả** |
| CodeTypeDeclaration | Khai báo một lớp, cấu trúc, giao diện. Các định nghĩa kiểu cơ bản tương ứng với một trong các thuộc tính như là IsClass, IsInterface, IsStruct, and IsEnum. |
| CodeTypeDeclarationCollection | Khai báo một tập các kiểu |
| CodeTypeDelegate | Khai báo một delegate. |

Bảng 3.2: Các kiểu của CodeDom để khai báo kiểu cấu trúc

Các kiểu của CodeDom để xây dựng các thành phần:

|  |  |
| --- | --- |
| **CodeDom Type để xây dựng thành phần** | **Mô tả** |
| CodeConstructor | Tạo một hàm tạo dựng cho một kiểu. |
| CodeEntryPoint | Tạo một phương thức đi vào của một chương trình với các thuộc tính mặc định (ví dụ, Main()) . |
| CodeMemberEvent | Tạo ra một khai báo sự kiện cho một lớp. |
| CodeMemberField | Khai báo code của một trường trong lớp. |
| CodeMemberMethod | Khai báo code của một phương thức trong lớp. |
| CodeMemberProperty | Khai báo code của một thuộc tính trong lớp. |
| CodeParameterDeclarationExpression | Khai báo code của một tham số của một thành viên. |
| CodeTypeConstructor | Đại diện cho hàm khởi tạo tĩnh cho một lớp. Hàm khởi tạo này được gọi khi kiểu của nó được chạy. |
| CodeTypeMember | Sử dụng một lớp cơ sở trừu tượng để khởi động một thành viên trong một kiểu. |
| CodeTypeMemberCollection | Khai báo một tập thành viên trong một kiểu được định nghĩa. |
| MemberAttributes | Khai báo các thuộc tính với các định dạng được sử dụng bởi CodeTypeMember. |

Bảng 3.3: Các kiểu của CodeDom để xây dựng các thành phần.

## 3.3. Cách sử dụng CodeDom để tạo một chương trình c#

Không gian tên này cung cấp một số kiểu để xây dựng một chương trình c#. với một chương trình C# đầy đủ bao gồm các namespace, các hàm sự kiện, phương thức và thuộc tính, và quan trọng là hàm Main(). Với codeDom có các hàm và kiểu giúp xây dựng một chương trình có thể tự sinh ra một chương trình c# như nó. Với một chương trình xây dựng với tên miền này nhằm tạo ra một chương trình khác thực hiện một công việc cụ thể nào đó mà người lập trình có thể chỉ định trước cho nó. Do vậy với 1 chương trình được lập trình ra để tạo ra một chương trình khác với mục đích đã có từ trước như vậy sẽ tạo ra sự tự động thực hiện một yêu cầu, hoặc thực thi hoặc chạy một test script.

* **Tạo Namespace và import các namespace cần thiết:**

*CodeNamespace cNamespace = new CodeNamespace("CodeDomExample");*

*cNamespace.Imports.Add(new CodeNamespaceImport("System"));*

* **Tạo class và thêm vào namespace**

*CodeTypeDeclaration cClass=new CodeTypeDeclaration("Rectangle");*

*cNamespace.Types.Add(cClass);*

* **Tạo các field:**

*CodeMemberField field1 = new CodeMemberField("Int32", "\_length");*

*field1.Attributes = MemberAttributes.Private;*

*CodeMemberField field2 = new CodeMemberField("Int32", "\_width");*

*field2.Attributes = MemberAttributes.Private;*

*cClass.Members.Add(field1);*

*cClass.Members.Add(field2);*

* **Tạo các property:**

**Với các thuộc tính ta sử dụng lớp CodeMemberProperty,** các property bao gồm các thuộc tính:

-    bool **HasGet**: getter  
-    bool **HasSet**: setter  
-    CodeStatementCollection **GetStatement**: các câu lệnh trong getter  
-    CodeStatementCollection **SetStatement**: các câu lệnh trong setter

Viết như sau:

*CodeMemberProperty property1 = new CodeMemberProperty();*

*property1.Name = "Length";*

*property1.Attributes = MemberAttributes.Public;*

*property1.Type = new CodeTypeReference("Int32");*

*property1.HasGet = true;*

*property1.HasSet = true;*

Để viết lệnh cho getter cho property này, bạn cần dùng lớp thêm một đối tượng kiểu  CodeMethodReturnStatement vào GetStatement để trả về field \_length. Cú pháp như sau:

*GetStatements ( return ( field ( this.\_length ) ) )*

Và thêm chúng vào bằng câu lệnh:

*property1.GetStatements.Add(new CodeMethodReturnStatement(new CodeFieldReferenceExpression(new CodeThisReferenceExpression(),"\_length")));*

*property1.SetStatements.Add(new CodeAssignStatement(new CodeFieldReferenceExpression(new CodeThisReferenceExpression(),"\_length"), new CodeArgumentReferenceExpression("value")));*

* **Tạo Constructor:**

Giả sử Constructor mà ta cần tạo sẽ có hai tham số length và width để khởi tạo giá trị cho hai field tương ứng của class.

*CodeConstructor cConstructor = new CodeConstructor();*

*cConstructor.Attributes = MemberAttributes.Public;*

*cConstructor.Parameters.Add(new CodeParameterDeclarationExpression("Int32", "length"));*

*cConstructor.Parameters.Add(new CodeParameterDeclarationExpression("Int32", "width"));*

* **Tạo phương thức Main()**

Mặc dù cũng là phương thức nhưng CodeDom có một lớp riêng để định nghĩa một phương thức có Entry Point. Cách sử dụng tương tự như bạn tạo phương thức thông thường, tất nhiên là bạn không cần gán tên cho phương thức này:

*CodeEntryPointMethod mainMethod = new CodeEntryPointMethod();*

* **Đóng gói tất cả vào CompileUnit:**

*CodeCompileUnit comUnit = new CodeCompileUnit();*

comUnit.Namespaces.Add(cNamespace);

*return comUnit;*

# Chương 4: Xây dựng ứng dụng kiểm thử tự động

## 4.1. Đề xuất công cụ kiểm thử phần mềm

Một công cụ kiểm thử hoàn thiện có thể giảm chi phí cho việc phát triển và làm tăng chất lượng phần mềm. Một công cụ kiểm thử tự động phải đảm bảo những đặc điểm như:

* Chính xác về mặt chức năng, độ tin cậy, khả năng tương tác.
* Giao diện thân thiện, dễ sử dụng, tìm hiểu và hoạt động.
* Tăng cường khả năng chịu lỗi và thu hồi lỗi tự động.
* Sản phẩm ổn định và có khả năng nâng cấp sau này.
* Cài đặt, gỡ bỏ dễ dàng, tương thích với hệ điều hành yêu cầu, bảo mật.

Như đã được biết, có một nhu cầu rất lớn cho công nghệ kiểm thử tin cậy. Các công cụ kiểm thử thường tụt hậu so với sự phát triển của các ứng dụng phần mềm (ITToolbox 1999). Kiểm thử viên thường xuyên phàn nàn rằng nhiều công cụ trong số các công cụ kiểm tra là khó hiểu và đôi khi gây nhầm lẫn khi sử dụng chúng. Họ thường không thể thực hiện các công việc thử nghiệm các nhà cung cấp công cụ tuyên bố họ có thể làm. Họ đều thích các công cụ kiểm thử phần mềm máy tính thêm vào thay vì các công cụ kiểm tra tự động đầy đủ hơn.

Hiện nay, các công cụ ngày càng hoàn thiện và nhiều chức năng hơn, cách dùng đễ dàng và tiện lợi hơn. Đối với đề tài này của nhóm thực hiện đề xuất ra phương pháp xây dựng một ứng dụng kiểm thử tự động phần mềm viết trên ngôn ngữ C#, thực hiện được các chức năng đơn giản và cơ bản như test về mặt chức năng, test đơn vị cho chương trình, thực hiện ghi nhận kết quả test xuất ra file Excel.

Mục đích xây dựng công cụ hỗ trợ cho kiểm thử, nhóm em phát triển công cụ với mục tiêu đề ra là thực hiện hỗ trợ kiểm thử viên viết test script, nhập dữ liệu các testcase từ các bản excel và thực thi tự động test script sau đó xuất kết quả trả về cho excell, so sánh đánh giá kết quả. Để thực hiện được việc đó, bằng ngôn ngữ lập trình C#, chúng ta quan tâm đến các không gian tên .NET Reflection, .NET Type class, MS Excel API programming và XML để hoàn thành.

## Không gian tên của .Net và các lớp phục vụ cho phát triển công cụ

Dưới đây là bảng liệt kê các không gian tên cần thiết nhất để thực hiện một công cụ, trong đó không gian tên CodeDom và Reflection là cốt lõi của tự động, System.Reflection thu thập các thông tin để kiểm tra và System.Codom sinh ra các test script.

|  |  |
| --- | --- |
| **Không gian tên (Namespace)** | **Chức năng** |
| System | Chứa các lớp có các chức năng cơ bản như chuyển đổi dữ liệu, các toán tử toán học, chương trình gọi, thu gom rác và quá trìnhquản lý môi trường. Đây là không gian tên lớn nhất được cung cấp bởi .NET. |
| System.CodeDom | Có các lớp bao gồm các phần tử của dữ liệu mã nguồn. Nó được dung để sinh ra các test script một cách tự động. |
| System.Collections | Chứa các lớp xác định các tập của đối tượng giống như các list, queue, sorted list, mảng (array), bảng băm và từ điển. Công cụ kiểm thử sử dụng không gian tên này để lưu trữ dữ liệu tạm thời. |
| System.ComponentModel | Chứa các lớp tạo ra các thành phần và điều khiển thời gian chạy và thời gian thiết kế. |
| System.Data | Chứa các lớp định nghĩa kiến trúc truy cập dữ liệu ADO.NET. Các kiến trúc ADO.NET cho phép xây dựng các thành phần có thể quản lý dữ liệu từ nhiều nguồn dữ liệu khác nhau trong cơ chế kết nối hoặc phi kết nối. |
| System.Diagnostics | Có các lớp có thể được dùng để gỡ lỗi các ứng dụng .NET, để theo dỗi việc thực thi các mã code của bạn và để theo dõi hiệu suất bộ đếm và đọc, viết các bản ghi sự kiện, và các quá trình bắt đầu và dừng lại. |
| System.Drawing | Chứa các lớp để thêm vào giao diện thiết bị đồ họa phục vụ vẽ chức năng. |
| System.IO | Contains classes that can read from and write to data streams and disk files with specifications of synchronous and asynchronous input/output.  Chứa các lớp có thể đọc và viết các luồng dữ liệu và các tệp trên đĩa với các thông số kỹ thuật của luồng vào/ ra đồng bộ và không đồng bộ. |
| System.Net | Có các lớp cung cấp một lớp bao xung quanh nhiều giao thức mạng được dùng hiện nay và xử lý các yêu cầu DNS, HTTP và FTP. Nếu bạn quan tâm đến kiểm thử web thì không gian tên này rất hữu ích. |
| System.Reflection | Chứa các lớp cung cấp một cái nhìn về kiểu, phương thức với các tham số, thuộc tính và trường có sẵn trong ứng dụng .NET. Thậm chí có thể lập trình tạo và gọi các kiểu (types) ngay tại thời gian chạy. |
| System.Resources | Cung cấp các lớp giúp người phát triển dự án tạo ra, lưu trữ và quản lý tài nguyên trong ứng dụng nội tại. |
| System.Runtime.InteropServices | Cung cấp chức năng cho mã nguồn không được quản lý. |
| System.Text | Chứa các lớp có thể sử dụng để làm việc với các bảng mã ký tự: ASCII, Unicode, UTF-7 và UTF-8 |
| System.Threading | Chứa các lớp cho phép đa luồng hệ thống vận hành cùng lúc trong ứng dụng .NET, do đó dùng để tạo ra một ứng dụng đa luồng thực sự. |
| System.Timers | Có các lớp để bắn ra một sự kiện vào một khoảng thời gian đã được định sẵn hoặc theo một lịch trình thời gian phức tạp hơn. |
| System.Web | Có các lớp để triển khai các giao thức chuyển đổi siêu văn bản (HTTP). |
| System.Windows.Forms | Chứa các lớp để xây dựng một ứng dụng Windows đầy đủ tính năng với các tính năng như hộp thoại, menu và các nút nhấn. |
| System.XML | Có các lớp để xử lý dữ liệu XML. Cung cấp hỗ trợ cho XML 1.0, không gian tên XML, lược đồ XML, XPath, XSL và XSLT, DOM Level 2,và SOAP 1.1. |

Bảng 4.1: Các không gian tên được sử dụng phát triển ứng dụng.

## Lưu trữ dữ liệu trên bảng tính worksheet

Quá trình thực hiện cần một không gian để lưu trữ dữ liệu để kiểm thử các thành phần. Dữ liệu bao gồm tên của các lớp trong một mã assembly, một cấu trúc, tên của những thành phần khác trong lớp hoặc cấu trúc, và giá trị của các đối số của phương thức hoặc cấu trúc lấy về từ các lớp Reflection ở trên. Các kiểm thử viên thường phải soạn thảo các dữ liệu lưu trữ này sử dụng các công cụ thương mại có sẵn hoặc mã nguồn mở.

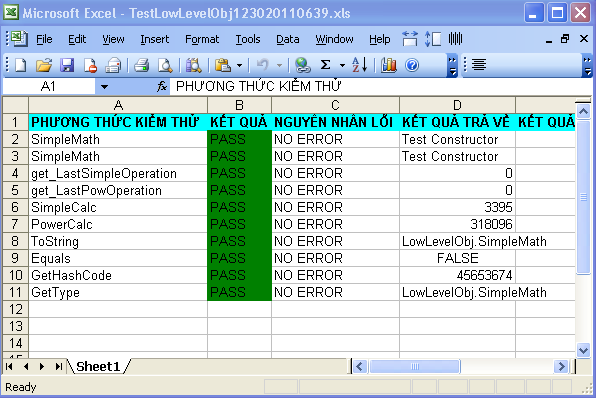
Khi kiểm thử viên thực hiện kiểm thử bằng tay, một kỹ sư thử nghiệm thường lưu trữ dữ liệu trong nhiều định dạng dữ liệu riêng biệt. Một công cụ kiểm thử thương mại cung cấp một phương pháp lưu trữ dữ liệu duy nhất và yêu cầu tuân theo nó. Đối với cả công cụ kiểm tra thương mại và công cụ mã nguồn mở dữ liệu thường được mã hóa cứng trong test script.

Có nhiều cách để lưu trữ dữ liệu thử nghiệm cho công cụ kiểm thử phần mềm này. Như các tài liệu XML đã trở thành một chuẩn cho nhiều tổ chức phần mềm sử dụng cho lưu trữ dữ liệu và trao đổi dữ liệu. Trong Microsoft Visual Studio .NET IDE, C# cho phép nhà phát triển thêm tài liệu hướng dẫn XML, và khi nó tự động sinh ra một tài liệu trợ giúp XML. Nó rất dễ dàng và hiệu quả cho các nhà phát triển để cài đặt một trường hợp kiểm thử (test case) khi viết code một phương thức với những dòng hướng dẫn XML. Công cụ kiểm thử tự động sẽ đọc dữ liệu được lưu trữ trong tài liệu XML. Sử dụng phương pháp này kiểm thử viên sẽ không bao giờ phải lo lắng về việc phỏng đoán mã code khi soạn các dữ liệu kiểm thử.

Ứng dụng mà nhóm xây dựng sẽ áp dụng bảng tính MS Excel và tài liệu XML để phát triển công cụ.

Một đối tượng MS Excel được tổ chức trong mô hình phân cấp đối tượng của chính nó. Các đối tượng ứng dụng là cao nhất trong hệ thông phân cấp của nó. Nó bao gồm tập hợp các bảng tính, biểu đồ, và các cửa sổ khác. Một bảng tính chứa một vài bảng nhỏ. Một bảng nhỏ gồm nhiều range và cell sắp xếp trong cột và hàng.

Dưới đây là kết quả khi chương trình thực hiện lấy về tên của lớp, các phương thức của lớp đó và các đối số của phương thức, tất cả được trình bày trong MS Excel:



Hình 4.1: Kết quả chạy test script lưu trên file MS excel.

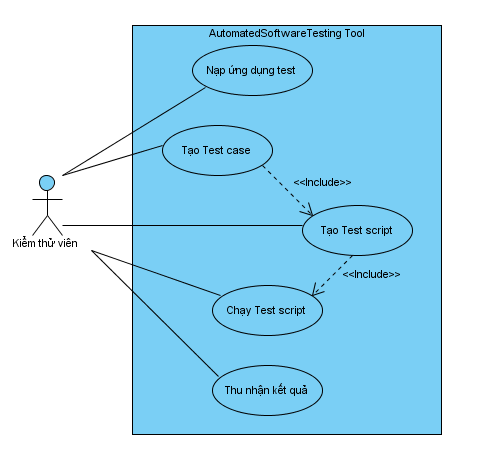
Các bảng tính này còn được sử dụng để lưu các test case phục vụ cho quá trình thực thi test script và sau đó thì lưu trữ kết quả test sau khi chạy test script.

## Phân tích thiết kế hệ thống:

### 4.4.1.Đặc tả yêu cầu:

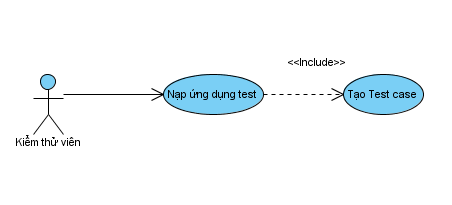
#### 4.4.1.1.Yêu cầu về chức năng:

* Biểu đồ Usecase tổng quát của hệ thống:



Hình 4.2: Usecase tổng quát.

* Chức năng 1(Usecase Tạo Test case):
* Mục đích
* Sinh ra Test case tự động dựa vào các yếu tố đầu vào là ứng dụng test.
* Tác nhân liên quan
* Kiểm thử viên.
* Điều kiện trước
* Sau khi công cụ test đã được bật, kiểm thử viên đã có chương trình chạy để kiểm thử.
* Điều kiện sau
* Sau khi test case được xuất ra, đối với các tham số của phương thức, hàm thì Kiểm thử viên có thể chỉnh sửa các tham số đầu vào này để các trường hợp kiểm thử phù hợp và có lợi nhất.
* Biểu đồ use-case



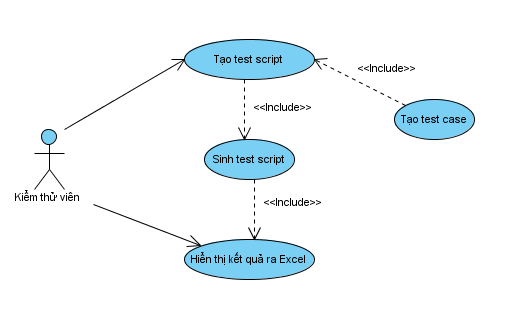
Hình 4.3: Usecase Tạo Test case.

* Dòng sự kiện chính

|  |  |
| --- | --- |
| Hành động của tác nhân | Phản ứng của hệ thống |
| 1. Kiểm thử viên nạp chương trình (file assembly) vào công cụ. | 1. Hệ thống hiển thị thư mục đường dẫn thư mục cho phép nạp chương trình test vào công cụ. |
| 1. Kiểm thử viên chọn chương trình cần test. | 1. Hệ thống xuất các dữ liệu của chương trình test như: lớp, thuộc tính, phương thức, cấu trúc ra file Excel và mặc định tham số đầu vào đối với các phương thức có tham số. |

Bảng 4.2: Dòng sự kiện Usecase Tạo Test case.

* Chức năng 2 (Usecase Tạo Test script):
* Mục đích
* Sinh ra Test script tự động dựa vào các yếu tố đầu vào là ứng dụng test và test case đã tạo ra được trước đó.
* Tác nhân liên quan
* Kiểm thử viên.
* Điều kiện trước
* Sau khi công cụ test sinh ra đc test case, kiểm thử viên đã chỉnh sửa test case và chấp nhận tạo test script.
* Điều kiện sau
* Sau khi test script được tạo thì công cụ tự động chạy test script và cho ra kết quả.
* Biểu đồ use-case



Hình 4.4: Usecase Tạo Test script.

* Dòng sự kiện chính

|  |  |
| --- | --- |
| Hành động của tác nhân | Phản ứng của hệ thống |
| 1. Nhấn vào nút tạo test script. | 1. Hệ thống tự động sinh test script và mở test script lên màn hình. |
| 1. Kiểm thử viên chạy (debug) test script. | 1. Hệ thống thực thi test script và xuất kết quả thực hiện ra file Excel. |

Bảng 4.3: Dòng sự kiện chính Usecase Tạo Test script.

#### 4.4.4.2.Yêu cầu phi chức năng:

* Yêu cầu về hiệu năng
* Ứng dụng thực hiện tạo test case và test script nhanh, thời gian phản hồi kết quả không quá 3s.
* Các ràng buộc thiết kế
* Ứng dụng phát triển trên nền tảng .NET framework với công cụ Visual Studio với ngôn ngữ lập trình là C# và sử dụng ứng dụng MS Excel để lưu trữ dữ liệu test.
* Ứng dụng được thiết kế có giao diện Window Form sáng sủa, khoa học, dễ thao tác và làm việc.

### 4.4.2.Phân tích – thiết kế hệ thống:

#### 4.4.2.1. Biểu đồ lớp

1. Danh sách các lớp đối tượng

|  |  |  |
| --- | --- | --- |
| STT | Tên lớp | Mô tả |
| **1** | Test case | Trường hợp kiểm thử chức năng cho ứng dụng. |
| **2** | Test script | Testscript được tạo ra để test ứng dụng. |

Bảng 4.4: Danh sách các lớp đối tượng.

1. Chi tiết hóa các lớp đối tượng:

* Lớp Test case:
* Danh sách các thuộc tính:

|  |  |  |
| --- | --- | --- |
| STT | Tên thuộc tính | Mô tả |
| **1** | MaTC | Mã Test case |
| **2** | TenTC | Tên Test case |
| **3** | Class | Tên lớp cần kiểm tra |
| **4** | Method | Phương thức thuộc lớp |
| **5** | Constructor | Cấu trúc thuộc lớp |
| **6** | Property | Thuộc tính của lớp |
| **7** | GTVao | Gía trị đầu vào |
| **8** | GTRa | Giá trị đầu ra |
| **9** | GTMongcho | Giá trị mong chờ của test case |

Bảng 4.5: Danh sách các thuộc tính của lớp test case .

* Danh sách phương thức:

|  |  |  |
| --- | --- | --- |
| STT | Tên phương thức | Mô tả |
| **1** | SinhTestcase | Sinh tự động test script |

Bảng 4.6: Danh sách các phương thức của lớp test case .

* Lớp Test script
* Danh sách các thuộc tính:

|  |  |  |
| --- | --- | --- |
| STT | Tên thuộc tính | Mô tả |
| **1** | MaTS | Mã Test script |
| **2** | TenTS | Tên Test script |
| **3** | TestCase | Test case tương ứng |
| **4** | Action | Hành động thực hiện |

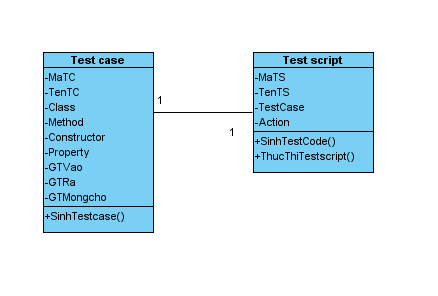
Bảng 4.7: Danh sách các thuộc tính của lớp test script .

* Danh sách phương thức:

|  |  |  |
| --- | --- | --- |
| STT | Tên phương thức | Mô tả |
| **1** | SinhTestScript | Sinh tự động test script |
| **2** | ThucThiTestscript | Thực thi test script để tạo ra kết quả test |

Bảng 4.8: Danh sách các phương thức của lớp test script .

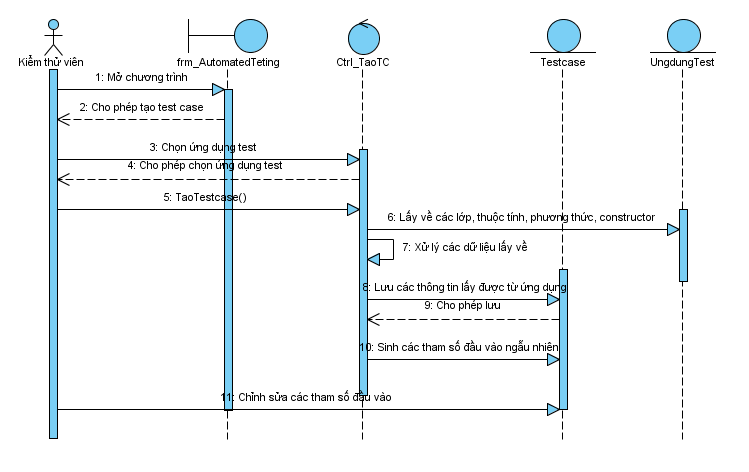
1. Mô hình hóa các lớp đối tượng



Hình 4.5: Biểu đồ lớp.

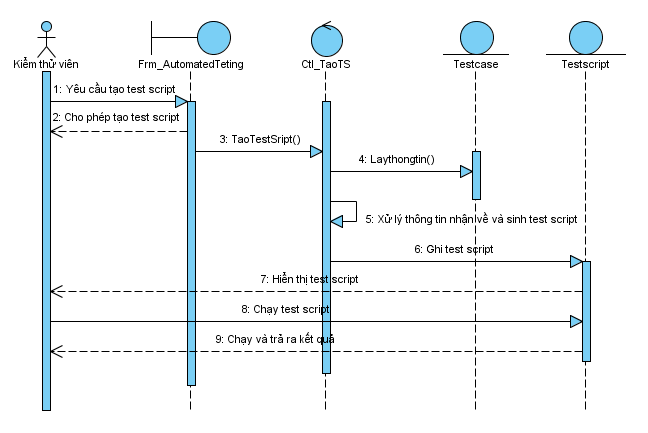
#### 4.4.2.2. Biểu đồ tuần tự

* Biểu đồ tuần tự cho ca sử dụng sinh Test case:



Hình 4.6: Biểu đồ tuần tự sinh test case.

* Biểu đồ tuần tự cho ca sử dụng sinh Test script



Hình 4.7: Biểu đồ tuần tự sinh test script.

### 4.4.3.Thiết kế giao diện

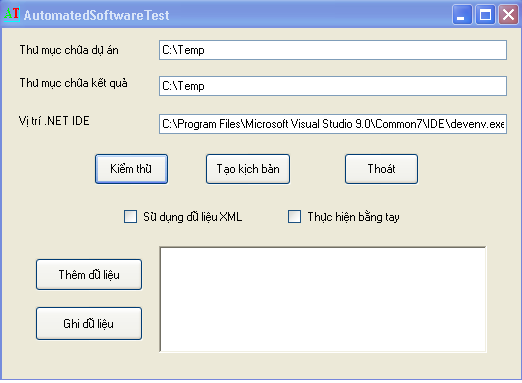
4.4.3.1. Thiết kế giao diện cho module kiểm thử chức năng cho một lớp:

* Danh sách các chức năng của module kiểm thử chức năng cho một lớp:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên chức năng | Tên form | Cách chọn từ chương trình |
| **1** | Kiểm thử chức năng | AutomatedSoftwareTest | Giao diện chính |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Bảng 4.9: Danh sách chức năng của module.

* Chi tiết hóa các giao diện của module kiểm thử chức năng một lớp
* Chức năng kiểm thử chức năng
* Mục đích:
* Kiểm thử về mặt chức năng của một lớp trong ứng dụng thực hiện chức năng đúng hay sai và ghi lại kết quả.
* Phạm vi:
* Kiểm thử về mặt chức năng của ứng dụng cần test.
* Ràng buộc:
* Dữ liệu đầu vào: ứng dụng cần test có khả năng chạy mà không bị lỗi.
* Dữ liệu đầu ra: kết quả kiểm tra sau khi chạy chương trình được ghi ra file Excel gồm đầy đủ các thông về lớp (Phương thức, thuộc tính, đối số…) và kết quả true/false tương ứng với các giá trị yêu cầu của test case.
* Giao diện chương trình:



11

10

9

8

7

6

5

3

1

2

4

Hình 4.8: Giao diện chương trình.

* Đặc tả giao diện:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thành phần** | **Kiểu** | **Mô tả** |
| **1** | txtTMnguon | Textbox | Ghi địa chỉ thư mục chứa dự án. |
| **2** | txtTMdich | Textbox | Ghi địa chỉ thư mục chứa kết quả của dự án. |
| **3** | txtIDE | Textbox | Ghi địa chỉ file assembly của ứng dụng test. |
| **4** | btnKiemthu | Button | Kích vào nút này để tạo tự động test case. |
| **5** | btnTaokichban | Button | Kích vào để tạo test script tự động. |
| **6** | btnThoat | Button | Thoát chương trình. |
| **7** | chkXML | Checkbox | Sử dụng kiểm thử với chương trình là file XML. |
| **8** | chkBangtay | Checkbox | Cho phép kiểm thử bằng tay |
| **9** | btnThemDL | Button | Cho phép kiểm thử nhiều chương trình. |
| **10** | btnGhiDL | Button | Ghi đường dẫn của từng chương trình test vào file text để phục vụ test nhiều chương trình. |
| **11** | rtfDl | Listbox | Thêm file Excel để kiểm thử. |

Bảng 4.10: Bảng đặc tả giao diện ứng dụng.

# PHẦN III: KẾT LUẬN VÀ KHUYẾN NGHỊ

## 3.1. Kết quả đạt được

Sau thời gian nghiên cứu và thực hiện đồ án 5 dưới sự hướng dẫn trực tiếp của thầy Đào Anh Hiển, nhóm chúng em đã đạt được những kết quả sau:

* Trình bày đầy đủ, chính xác các định nghĩa về kiểm thử phần mềm và kiểm thử tự động; trình bày chi tiết về các kỹ thuật Reflection và CodeDom của .NET.
* Từ kỹ thuật đã nghiên cứu và trình bày, đưa ra cách xây dựng một ứng dụng kiểm thử tự động dựa vào các kỹ thuật đã trình bày.
* Áp dụng lý thuyết xây dựng một ứng dụng kiểm thử tự động phần mềm về mặt chức năng của phần mềm đó.
* Kết quả nghiên cứu là tài liệu mang tính khoa học, là nguồn tham khảo cho sinh viên khi nghiên cứu về kiểm thử tự động và ứng dụng nó vào xây dựng công cụ kiểm thử.

## 3.2. Hạn chế

Mặc dù nhóm chúng em đã hết sức cố gắng để hoàn thiện đồ án nhưng do thời gian nghiên cứu và thực hiện đồ án có hạn cũng như kiến thức chuyên môn chưa cao nên nhóm mới xây dựng ứng dụng nhỏ với chức năng còn hạn chế là kiểm thử về mặt chức năng của chương trình, chưa mở rộng kiểm thử được nhiều nội dung khác.

## 3.3. Hướng phát triển

Trong thời gian tới, nhóm chúng em sẽ tiếp tục nghiên cứu sâu hơn về kiểm thử phần mềm, kiểm thử tự động và các kỹ thuật của .NET hỗ trợ tự động để xây dựng và hoàn thiện ứng dụng của mình.

## 3.4. Đề xuất ý kiến

Hiện nay ngành kiểm thử phần mềm rất phát triển và có vị trí chiến lược trong phát triển và gia công phần mềm. Thị trường lao động Việt Nam đang “khát” kỹ sư kiểm thử phần mềm có chuyên môn cao, đáp ứng yêu cầu công việc. Vì vậy, việc đào tạo kỹ sư kiểm thử phần mềm không chỉ có kiến thức kỹ năng về kiểm thử mà còn có khả năng tự xây dựng công cụ test hỗ trợ công việc một cách chuyên nghiệp. Nhóm em xin đề xuất một số ý kiến cho khoa Công nghệ thông tin trường Đại học sư phạm kỹ thuật Hưng Yên như sau:

* Cần có nhiều giảng viên nghiên cứu sâu về kiểm thử phần mềm.
* Khoa nên đưa thêm các môn học về kiểm thử phần mềm vào giảng dạy.
* Khuyến khích hơn nữa giáo viên và sinh viên nghiên cứu về kiểm thử phần mềm.
* Khuyến khích giáo viên và sinh viên nghiên cứu về công cụ hỗ trợ kiểm thử phần mềm.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1].  *Effective Software Test Automation: Developing an Automated Software Testing Tool* by Kanglin Li and Menqi Wu. | |
|  |  |

[2]. YinYang’s Programming Blog, <http://yinyangit.wordpress.com>

[3]. Forum Cộng đồng C Việt, <http://diendan.congdongcviet.com/>

[4]. <http://testervn.com/>

[5]. <http://www.testingvn.com/>