

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

NGUYỄN THỊ THU HÀ

**NGHIÊN CỨU MỘT SỐ GIẢI PHÁP KIỂM THỬ
GIAO DIỆN TỰ ĐỘNG SỬ DỤNG RANOREX**

Ngành: Công nghệ thông tin

Chuyên ngành: Kỹ thuật phần mềm

Mã Số: 8480103.01

TÓM TẮT LUẬN VĂN THẠC SĨ

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS. TS. PHẠM NGỌC HÙNG

Hà Nội - 11/2018

MỤC LỤC

Chương 1: Đặt vấn đề.....	2
1.1. Sự cần thiết của đề tài	2
1.2. Nội dung của luận văn	3
1.3. Cấu trúc của luận văn.....	4
Chương 2: Tổng quan về kiểm thử giao diện người dùng tự động	5
2.1. Kiểm thử phần mềm.....	5
2.2. Kiểm thử tự động phần mềm	5
2.3. Kiểm thử giao diện người dùng	6
2.4. Một số phương pháp kiểm thử giao diện người dùng.....	6
Chương 3: Kiểm thử giao diện người dùng sử dụng Ranorex	13
3.1. Giới thiệu về Ranorex	13
3.2. Tính năng nổi bật	13
Chương 4: Ứng dụng và thực nghiệm	15
4.1. Giới thiệu về phần mềm Keepass	15
4.2 Thực nghiệm	15
Chương 5. Kết luận.....	19
TÀI LIỆU THAM KHẢO	21

Chương 1: Đặt vấn đề

1.1. Sự cần thiết của đề tài

Trong vài thập kỉ qua, ngành công nghiệp phần mềm đã có những bước phát triển lớn cả về quy mô và chất lượng, đóng vai trò quan trọng vào sự phát triển của hầu hết mọi mặt của các nước. Nếu như trước đây, phần mềm máy tính chỉ được sử dụng để tính toán khoa học kỹ thuật và xử lý dữ liệu, thì ngày nay, nó đã được ứng dụng vào mọi mặt của đời sống hàng ngày của con người. Các ứng dụng của phần mềm trong cuộc sống rất đa dạng từ các ứng dụng nhỏ để điều khiển các thiết bị gia dụng như điện thoại, máy giặt, ti vi, tủ lạnh đến các ứng dụng lớn hơn cho rất nhiều người dùng cùng sử dụng như hệ thống quản lý doanh nghiệp, các hệ thống hướng dẫn giao thông, hệ thống quản lý việc khám chữa bệnh, v.v. Điều này đòi hỏi chất lượng phần mềm ngày càng phải được nâng cao để đáp ứng nhu cầu của người sử dụng.

Tuy nhiên, quá trình tạo ra một sản phẩm phần mềm có thể sử dụng tốt không thể tránh khỏi những lỗi phần mềm. Chúng ta dù cố gắng đến mức nào thì thực tế là ngay cả những lập trình viên xuất sắc nhất cũng không thể lúc nào cũng viết được những đoạn mã không có lỗi. Tính trung bình, ngay cả một lập trình viên loại tốt thì cũng có từ một đến ba lỗi trên một trăm dòng lệnh. Người ta ước lượng rằng việc kiểm tra để tìm ra các lỗi này chiếm phân nửa khối lượng công việc phải làm để có một phần mềm hoạt động được [1].

Do vậy, kiểm thử phần mềm là khâu rất quan trọng của sản phẩm trước khi đưa vào sử dụng, góp phần quyết định sự thành công của dự án phần mềm. Tuy nhiên, kiểm thử là một công việc tiêu tốn rất nhiều thời gian, tiền bạc, công sức. Chi phí kiểm thử phần mềm thường chiếm tới bốn mươi phần trăm tổng chi phí cho một dự án phát triển phần mềm. Đối với các phần mềm lớn, chi phí này còn tăng lên gấp bội mỗi khi có sự thay đổi, nâng cấp các chức năng của phần mềm, điều này là không thể tránh khỏi đối với mọi phần mềm.

Một sản phẩm tuy được thiết kế tốt nhưng cũng không thể tránh khỏi các sai sót. Kiểm thử hiệu quả sẽ phát hiện ra được các sai sót này, tránh các lỗi

trước khi phát hành sản phẩm. Kiểm thử đứng dưới vai trò của người sử dụng, sẽ giúp cho sản phẩm có sự thích ứng phù hợp hơn với thị hiếu và nhu cầu ngày càng cao của người dùng. Trên thị trường hiện nay có rất nhiều công cụ kiểm thử tự động được sử dụng như Ranorex, QTP, Selenium, v.v. Đề tài này tìm hiểu về các công cụ hỗ trợ kiểm thử tương tác giao diện cho các ứng dụng và đi sâu nghiên cứu công cụ Ranorex vì nó có rất nhiều ưu điểm như hỗ trợ đa nền tảng, hỗ trợ nhiều ứng dụng trên Web, Desktop, Mobile. Ranorex có khả năng xác định chính xác các đối tượng có trong UI hiện nay. Hơn nữa, công cụ này hỗ trợ cơ chế “ghi và chạy lại” kịch bản tương tác UI rất mạnh mẽ. Công cụ này cũng cho phép kiểm thử viên tùy chỉnh kịch bản tương tác UI bằng cách thêm trực tiếp các đoạn mã nhằm tăng tính linh hoạt trong kiểm thử tự động. Giao diện đồ họa người dùng (Graphical user interface – GUI) là những gì người dùng nhìn thấy. Nếu bạn truy cập vào một trang Web, những gì bạn thấy trên trang chủ được gọi là giao diện đồ họa người dùng của trang Web. Người dùng sẽ không nhìn thấy mã nguồn, giao diện người dùng chỉ tập trung vào cấu trúc thiết kế, hình ảnh hiển thị ra ngoài có đúng như lập trình mong đợi hay không [2]. Nếu chúng ta phải làm thử nghiệm GUI, việc đầu tiên cần xác định xem những hình ảnh của trang Web sẽ hiện lên giống nhau trên các trình duyệt khác nhau. Ngoài ra, kiểm thử GUI còn xác nhận các liên kết hoặc các nút hoạt động tốt hay không, nếu người dùng thay đổi kích thước màn hình thì hình ảnh và nội dung không được co lại hoặc cắt đi hay chồng chéo lên nhau [2]. Để đạt được mục tiêu này, luận văn cũng sẽ nghiên cứu về kiểm thử giao diện tự động và các kiến thức liên quan. Cuối cùng luận văn sẽ áp dụng trực tiếp kiểm thử tự động giao diện sử dụng công cụ Ranorex vào để kiểm thử phần mềm bảo mật password Keepass nhằm phát hiện một số lỗi tương tác giao diện cho ứng dụng này.

1.2. Nội dung của luận văn

Với mục đích như trên, luận văn có những nội dung như sau: Luận văn tổng hợp lý thuyết về kiểm thử phần mềm, kiểm thử tự động, kiểm thử giao diện tự động - một giải pháp góp phần nâng cao năng suất, chất lượng hoạt động

kiểm thử phần mềm. Luận văn giới thiệu về một số công cụ hỗ trợ kiểm thử giao diện tự động trong đó sẽ đi tìm hiểu sâu về công cụ Ranorex. Luận văn sẽ mô tả từng bước quá trình áp dụng kiểm thử giao diện tự động với công cụ Ranorex từ đó giúp phần mềm giảm bớt chi phí kiểm thử cũng như tiết kiệm được thời gian và nhân lực kiểm thử của các kiểm thử viên. Tránh được các lỗi khi làm dự án nhất là các dự án lớn đòi hỏi sự chính xác rất cao.

1.3. Cấu trúc của luận văn

Phần còn lại của luận văn được cấu trúc như sau. Chương 2 giới thiệu tổng quan về kiểm thử, kiểm thử giao diện người dùng và các khái niệm cơ bản được sử dụng trong nghiên cứu của luận văn. Chương này chủ yếu giới thiệu về kiểm thử, kiểm thử tự động và kiểm thử giao diện tự động. một số phương pháp hỗ trợ kiểm thử giao diện người dùng. Tiếp đến, kiểm thử giao diện người dùng sử dụng Ranorex sẽ được mô tả trong Chương 3. Trong chương này sẽ giới thiệu chi tiết về công cụ kiểm thử giao diện tự động Ranorex. Cơ chế hoạt động sinh kịch bản, chạy kịch bản và xuất ra kết quả của công cụ, giới thiệu cả những tính năng ứng dụng nổi bật của công cụ. Từ đó, luận văn sẽ tổng kết những tính năng nổi bật khi sử dụng công cụ Ranoex trong những dự án lớn. Chương 4 là việc ứng dụng và thực nghiệm. Công cụ kiểm thử giao diện tự động sẽ được đưa vào ứng dụng thực tế trong chương trình phần mềm bảo mật password Keepass nhằm minh chứng cho khả năng vận dụng các kiến thức tìm hiểu được của học viên. Cuối cùng, tổng kết những kết quả đạt được của luận văn và hướng nghiên cứu tiếp theo sẽ được trình bày trong Chương 5.

Chương 2: Tổng quan về kiểm thử giao diện người dùng tự động

Hiện nay, phần mềm được sử dụng rất rộng rãi trong rất nhiều lĩnh vực như khoa học, kinh tế và xã hội. Vì vậy, việc đảm bảo rằng phần mềm đáp ứng được các mong muốn của người sử dụng là rất quan trọng. Kiểm thử phần mềm lúc này trở thành một trong những hoạt động cơ bản và cần thiết nhằm đảm bảo chất lượng phần mềm.

2.1. Kiểm thử phần mềm

Kiểm thử phần mềm có nhiều cách định nghĩa khác nhau, tuy nhiên chúng cùng bao trùm hai nội dung cơ bản là phát hiện lỗi và đánh giá chất lượng của phần mềm. Định nghĩa của Glenford J. Myers: “*Kiểm thử phần mềm là quá trình thực thi một chương trình với mục đích tìm ra lỗi*” [3] được xem là đơn giản và được sử dụng nhiều nhất.

Mục đích của kiểm thử là phát hiện lỗi vì thực tế phần mềm nào cũng có lỗi. Theo Myers, kiểm thử mà không phát hiện được lỗi được coi là không thành công [4]. Lỗi phần mềm cần tìm ra sớm vì nếu để sản phẩm đến tay người dùng thì càng tốn nhiều thời gian và tiền bạc để sửa lỗi. Do đó, kiểm thử phần mềm được thực hiện ngay trong quá trình phát triển phần mềm trước khi sản phẩm được đến tay người dùng.

2.2. Kiểm thử tự động phần mềm

Phương pháp kiểm thử thủ công vừa tốn nhiều thời gian, công sức để thực hiện, vừa không phải lúc nào cũng có hiệu quả trong việc tìm kiếm các lớp lỗi. Vì vậy, kiểm thử tự động cung cấp một khả năng để thực hiện các loại kiểm thử một cách hiệu quả.

Kiểm thử tự động là quá trình thực hiện một cách tự động các bước trong một kịch bản kiểm thử. Kiểm thử tự động sử dụng phần mềm kiểm thử (khác biệt với kiểm thử bằng tay) để kiểm soát việc thực hiện các bài kiểm tra và so sánh kết quả thực tế với kết quả dự đoán. Mục đích của kiểm thử tự động là tăng

độ tin cậy, tăng tính hiệu quả, giảm thời gian, công sức, kinh phí, giảm sự nhầm lẫn cho kiểm thử viên trong quá trình kiểm thử.

2.3. Kiểm thử giao diện người dùng

Giao diện người dùng (User Interface - UI) là một phương thức giao tiếp giữa con người với các thiết bị, máy móc và chương trình máy tính. Trong lĩnh vực phát triển phần mềm, UI được thiết kế nhằm mục đích giúp con người điều khiển, sử dụng, tương tác với các chương trình phần mềm để đạt được mục đích sử dụng của người dùng. Trong lịch sử phát triển, UI được chia thành nhiều dạng. Trong đó, hai dạng UI được sử dụng phổ biến cho tới ngày nay là giao diện dòng lệnh (Command-line interface) và giao diện đồ họa người dùng (Graphical user interface - GUI).

2.4. Một số phương pháp kiểm thử giao diện người dùng

Kiểm thử giao diện người dùng thủ công: Dựa trên tri thức, hiểu biết về miền của kiểm thử viên. Kiểm thử viên tập trung kiểm thử các chức năng quan trọng của hệ thống hay các kịch bản có thể gây ra lỗi. Tuy nhiên, rất nhiều kịch bản quan trọng có thể bị bỏ sót khi kiểm thử thủ công. Trong thực tế, thường có một nhóm các chuyên gia tập trung giải quyết vấn đề bằng cách đánh giá qua kinh nghiệm, tìm giải pháp qua thử nghiệm và rút bớt khuyết điểm. Một kỹ thuật khác được sử dụng là đi qua có nhận thức (Cognitive walkthrough). Kiểm thử viên lần lượt sử dụng các chức năng của chương trình, các hành động và phản hồi được ghi lại để đối chiếu với mục tiêu kỳ vọng. Cuối cùng, các điểm cần cải tiến, sửa chữa được ghi lại. Ngoài ra, việc kiểm thử tính dễ sử dụng của chương trình cũng được kiểm thử viên tiến hành thủ công. Kiểm thử UI thủ công có thể phát hiện được một số lỗi mà kiểm thử tự động không thể phát hiện được. Tuy nhiên, phương pháp này yêu cầu nhiều chi phí về thời gian và nhân lực. Đồng thời, độ phủ đạt được là không cao. Đối với các hệ thống có UI phức tạp, kiểm thử thủ công là giải pháp không khả thi.

Kiểm thử giao diện người dùng dựa trên mô hình: Mô hình là một mô tả đồ họa về hành vi của hệ thống. Nó giúp chúng ta hiểu và dự đoán hành vi của hệ thống. Quá trình kiểm thử UI tự động dựa trên mô hình bao gồm ba giai đoạn chính như sau:

- Mô hình hóa hệ thống từ các bản đặc tả và thiết kế của hệ thống: Hệ thống được mô hình hóa sử dụng công cụ mô hình hóa cùng với các bản đặc tả, phân tích thiết kế từ khách hàng. Việc này giúp hiểu rõ hơn hệ thống cần kiểm thử và UI của hệ thống đó.
- Sinh kịch bản tương tác UI từ mô hình của hệ thống: Tùy thuộc vào tiêu chí và thuật toán sinh kịch bản tương tác UI mà số lượng kịch bản sinh ra là khác nhau. Nhìn chung, kỹ thuật sinh kịch bản tương tác UI từ mô hình cho số lượng kịch bản và độ phủ lớn hơn nhiều so với các kỹ thuật kiểm thử UI khác. Đây là một ưu điểm đáng chú ý của kỹ thuật này so với các kỹ thuật còn lại.
- Chạy các kịch bản tương tác UI trên chương trình cần kiểm thử để thu được kết quả. Các dự đoán kết quả kiểm thử, giá trị đầu ra mong muốn được đưa vào trước đó để so sánh với giá trị kết quả thực tế sau khi chạy nhằm xác định kịch bản tương tác UI nào thành công. Pha này có thể chạy tự động sử dụng các công cụ hỗ trợ nhằm giảm thiểu chi phí.

Trong những năm gần đây đã có nhiều nghiên cứu về việc sử dụng các mô hình phù hợp cho kiểm thử UI tự động. Trong đó, mô hình đồ thị là mô hình được sử dụng phổ biến để mô hình hóa UI của chương trình cần được kiểm thử. Mô hình này mô tả toàn bộ các chuỗi thao tác người dùng có thể thực hiện với các đối tượng có trên UI. Một số dạng đồ thị được sử dụng để xây dựng mô hình như đồ thị dòng sự kiện, đồ thị tương tác sự kiện, v.v.

Kiểm thử giao diện người dùng bằng cách ghi và chạy lại kịch bản tương tác: Kiểm tra GUI có thể được thực hiện bằng các công cụ tự động hóa. Việc này được thực hiện thành hai quá trình. Trong quá trình ghi lại, các bước kiểm tra được bắt bởi công cụ tự động hóa. Trong quá trình phát lại, các bước kiểm tra được ghi lại được thực hiện trên ứng dụng đang chạy thử.

2.5. Một số công cụ kiểm thử giao diện người dùng tự động

Selenium

Selenium¹ là một công cụ kiểm tra phần mềm được sử dụng để kiểm tra hồi quy (Regression Testing). Selenium được dùng để kiểm thử các ứng dụng trên nền Web. Năm 2004, Selenium được phát triển bởi ThoughtWorks với tên ban đầu là JavaScriptTestRunner. Đến năm 2007, tác giả Jason Huggins rời ThoughtWorks và gia nhập Selenium Team (thuộc Google), từ đó tiếp tục phát triển Selenium như hiện nay. Đây là một công cụ kiểm tra mã nguồn mở cung cấp chức năng phát lại và thu âm để kiểm tra hồi quy.

Bên cạnh mã nguồn mở, Selenium hỗ trợ một loạt các ngôn ngữ bao gồm Java, Python, PHP, C#, Ruby, thậm chí cả Java Script thuần túy. Selenium là công cụ mã nguồn mở mạnh mẽ nhất có sẵn và nó dựa trên Java script trong một giới hạn lớn. Nó phù hợp hơn cho phương pháp phát triển nhanh của việc phát triển và kiểm thử. Tính năng ghi lại của Selenium được thực hiện như là một phần thêm vào trình duyệt Firefox, và cho phép ghi lại, chỉnh sửa và gỡ rối các kịch bản kiểm thử.

HP Quick Test Pro (QTP) hoặc HPE Unified Functional Testing (UFT)

Quick Test Professional² (QTP) là một công cụ kiểm thử tự động được thiết kế bởi Mercury Interactive và sau đó được mua lại bởi HP. QTP giúp người kiểm thử (tester) tiến hành các kiểm tra một cách tự động để xác định lỗi khác với kết quả mong muốn của ứng dụng, phần mềm hay chức năng v.v. mà ta đang kiểm tra [10]. QTP được sử dụng rộng rãi để kiểm tra chức năng (Functional Testing) và tiến hành các hoạt động kiểm thử hồi quy (Regression Testing), giải quyết các ứng dụng phần mềm. Để đơn giản hóa việc tạo và bảo trì thử nghiệm, nó sử dụng khái niệm kiểm tra từ khóa.

¹ <https://www.seleniumhq.org/>

² <http://shivaquicktestpro.blogspot.com/>

Ranorex

Ranorex³ cung cấp một loạt các công cụ tự động hóa máy tính để bàn, Web và di động được sử dụng để nâng cao chất lượng phần mềm của công ty. Nó có thể thực thi kiểm thử chức năng và phi chức năng trên môi trường Win và Web, hỗ trợ tất cả các trình duyệt hiện hành. Kiểm thử Mobile trên các hệ điều hành Android, IOS. Ranorex không có ngôn ngữ lập trình riêng của chính nó, thay vào đó, nó dùng ngôn ngữ lập trình như C# và VB.Net. Công cụ hỗ trợ một số dạng kiểm thử dưới đây:

- của người dùng lên các phần tử đó.

Bảng 2.1 liệt kê những điểm nổi bật của Ranorex so với Selenium và QTP. Về vấn đề cài đặt thì cài đặt Selenium phức tạp hơn việc cài đặt công cụ QTP và Ranorex. Đối với những người chưa có kiến thức về lập trình thì sẽ cảm thấy khó khăn khi cài đặt và thiết lập môi trường của Selenium. Còn với Ranorex và QTP thì chúng ta chỉ cần tải về và làm theo các bước cài đặt thông thường là có thể sử dụng được.

Bảng 2.1. So sánh mức độ thân thiện khi sử dụng công cụ QTP, Selenium và Ranorex

	Selenium	QTP	Ranorex
Cài đặt và cấu hình			
Dễ dàng cài đặt cho người không phải lập trình viên	✗	✓	✓
Khả năng bảo trì			
Dễ dàng bảo trì	✗	✓	✓
Dễ dàng thực thi cho mọi			

³ <https://www.ranorex.com/>

	Selenium	QTP	Ranorex
người			
Dễ dàng thực thi với việc chạy tệp .exe	✗	✓	✓
Dịch vụ hỗ trợ chuyên nghiệp			
Tăng cường hỗ trợ và đào tạo trong việc thực hiện dự án	✗	✓	✓
Hỗ trợ từng cá nhân	✗	✓	✓
Giấy phép			
Miễn phí	✓	✗	✗

Ranorex dễ dàng có thể chạy cho cả các tệp .exe bởi khả năng nhận diện đối tượng mạnh của nó. Dịch vụ hỗ trợ khi sử dụng thì QTP và Ranorex đều cung cấp hỗ trợ khách hàng chuyên nghiệp chúng có hỗ trợ đào tạo trong việc thực hiện dự án và có thể hỗ trợ từng cá nhân khi làm việc, còn Selenium thì không hỗ trợ người dùng chính thức nào khi đang được cung cấp dịch vụ.

Nhưng về mặt giấy phép sử dụng thì Selenium là công cụ mã nguồn mở nên sử dụng hoàn toàn miễn phí, với QTP muốn sử dụng phải mua giấy phép với chi phí khoảng 8000 USD còn với Ranorex thì chi phí khoảng 3500 USD/năm sử dụng. Đây chính là hạn chế của Ranorex vì thế nên Ranorex thường chỉ được sử dụng ở các dự án lớn với nhiều kinh phí.

Bảng 2.2. So sánh tính năng sử dụng của Selenium, QTP và Ranorex

	Selenium	QTP	Ranorex
Công nghệ hỗ trợ			
Ứng dụng desktop	✗	✓	✓

Ứng dụng web	✓	✗	✓
Ứng dụng mobile	✗	✓	✓
Chụp và chạy lại			
Ghi lại hành động	✓	✓	✓
Bảng hành động cho việc chỉnh sửa các bước sau khi ghi	✗	✗	✓
Khả năng mở rộng			
Thực thi kiểm tra song song	✓	✗	✓
Công cụ tự động hóa thử nghiệm			
Thiết lập và cấu hình dễ dàng	✗		✓
Tạo các bài kiểm tra mà không cần lập trình	✗		✓
Đã bao gồm đầy đủ IDE	✗		✓
Cấu trúc thử nghiệm mô-đun và tái sử dụng	✗		✓
So sánh dựa trên hình ảnh	✗		✓
Báo cáo được tích hợp sẵn	✗		✓
Xác định và quản lý phần tử UI			
Nhận dạng đối tượng mạnh mẽ	✗		✓
Kho lưu trữ đối tượng có thể chia sẻ	✗		✓
Trình chỉnh sửa bản đồ đối tượng UI	✗		✓
Hỗ trợ cho ID động	✗		✓
Đồng bộ hóa đối tượng UI tự động	✗		✓

Bảng 2.3. Một số tính năng chuyên sâu của Selenium, QTP và Ranorex

	Selenium	QTP	Ranorex
Xác định đối tượng			

	Selenium	QTP	Ranorex
Sử dụng Xpath ⁴	✓	✗	✓
Chỉnh sửa Xpath trên màn hình giao diện	✗	✗	✓
Đồng bộ hóa đối tượng giao diện tự động	✗	✗	✓
Xác định và quản lý phần tử UI			
Nhận dạng đối tượng mạnh mẽ	✗		✓
Kho lưu trữ đối tượng có thể chia sẻ	✗		✓
Trình chỉnh sửa bản đồ đối tượng UI	✗		✓
Hỗ trợ cho ID động	✗		✓
Đồng bộ hóa đối tượng UI tự động	✗		✓
Hội nhập			
Tập thử nghiệm thực thi để tích hợp đơn giản với máy chủ CI ⁵	✗		✓
Tích hợp với Visual Studio	✓		✓
Tự động tạo các báo cáo tương thích JUnit	✗		✓

⁴ XPath được định nghĩa là đường dẫn XML. Nó là một cú pháp hoặc ngôn ngữ để tìm kiếm bất kỳ phần tử nào trên trang web bằng cách sử dụng biểu thức XML path. XPath được sử dụng để tìm vị trí của bất kỳ phần tử nào trên trang web bằng cách sử dụng cấu trúc DOM HTML

⁵ Continuous Integration là một thực hành của việc liên tục tích hợp những thay đổi tạo ra với project và test lại nó hàng ngày hoặc thường xuyên hơn.

Chương 3: Kiểm thử giao diện người dùng sử dụng Ranorex

Trong kiểm thử giới thiệu những khái niệm chung, kiểm thử tự động và kiểm thử giao diện tự động trong đó nhấn mạnh hơn về kiểm thử giao diện tự động. Khung kiểm thử giao diện tự động là một tập hợp các giả định, các khái niệm và công cụ được cung cấp để hỗ trợ cho quy trình kiểm thử tự động. Nó là một hệ thống tích hợp thiết lập các qui tắc tự động hóa một sản phẩm cụ thể chủ yếu là về giao diện của sản phẩm. Vì giao diện của mỗi sản phẩm được coi là phần quan trọng của sản phẩm đó, và cũng là phần gần như là chính nhất mà khách hàng quan tâm và sử dụng. Với mục tiêu áp dụng kiểm thử giao diện tự động trong chương này sẽ trình bày về công cụ kiểm thử tự động Ranorex. Và chương này sẽ mô tả các tính năng nổi bật của Ranorex.

3.1. Giới thiệu về Ranorex

Nói về kiểm tra UI, tất cả những gì mà bạn phải làm là viết kịch bản thử nghiệm, một vài kịch bản cho mỗi môi trường nhưng nó hơi tẻ nhạt. Và bạn không muốn thử nghiệm của mình thất bại. Vì vậy, việc bạn phải sửa lại kịch bản của mình hoặc viết kịch bản mới mỗi khi có bản nâng cấp hoặc một nút thay đổi, một hộp thoại, văn bản thay đổi, di chuyển hay bất cứ điều gì thay đổi trong sản phẩm. Những thay đổi chỉ mất một vài giờ, mỗi thay đổi mỗi tập lệnh có thể không phải là vấn đề nhưng tất nhiên khi sự thay đổi là nhiều thì lúc đó việc kiểm thử bằng tay sẽ khiến bạn phải đau đầu. Khi đó việc sử dụng các công cụ tự động sẽ hơn rất nhiều so với việc dùng thử nghiệm bằng tay.

3.2. Tính năng nổi bật

Dễ dàng sử dụng cho người mới bắt đầu, ứng dụng mạnh mẽ cho các chuyên gia. Với áp lực thời gian của kiểm thử hồi quy bằng tay và đa nền tảng, một công cụ kiểm thử tự động là thứ không thể thiếu. Nhiều công cụ kiểm thử tự động đòi hỏi kỹ năng mã hóa nâng cao, chỉ giới hạn ở một số nền tảng nhất định hoặc yêu cầu tích hợp phức tạp để tự động hóa giao diện người dùng. Ranorex

Studio giải quyết những thách thức này với các công cụ để sử dụng để tự động hóa thử nghiệm không cần sử dụng mã.

Tính năng nhận dạng đối tượng mạnh mẽ nhận dạng và phân tích ngay lập tức các yếu tố giao diện người dùng của Windows, Web hoặc ứng dụng di động.

Chỉnh sửa và ghi hành động dễ dàng tạo các dự án tự động hóa thử nghiệm mà không cần mã hóa. Dễ dàng ghi lại và phát lại quy trình tự động hóa thử nghiệm bằng trình ghi Ranorex.

Chỉnh sửa mã tạo các kịch bản tự động hóa thử nghiệm linh hoạt bằng các ngôn ngữ lập trình chuẩn. Vì thư viện Ranorex dựa trên Microsoft.NET framework, khi làm việc với Ranorex không yêu cầu phải học một ngôn ngữ kịch bản độc quyền, đây là một lợi thế cho các kiểm thử viên (tester).

Trực tiếp tích hợp Ranorex vào nhiều hệ thống kiểm soát nguồn Ranorex tích hợp trực tiếp với các hệ thống điều khiển phiên bản TFS, Git và SVN.

Tích hợp Ranorex vào môi trường phát triển kiểm thử tự động là điều cần thiết cho cả nhà phát triển và người kiểm thử. Ranorex Studio cung cấp tất cả các điều kiện cần thiết để đảm bảo các nhà phát triển và người kiểm thử có thể làm việc liền mạch với nhau trên các dự án kiểm thử tự động.

Kiểm tra trình duyệt chéo (tích hợp Ranorex với Selenium) Selenium WebDriver được tích hợp vào API lõi của Ranorex để bạn có thể tạo các thử nghiệm qua trình duyệt bằng cách sử dụng các công cụ không mã hóa của Ranorex Studio hoặc các ngôn ngữ lập trình chuẩn C# và VB.NET.

Báo cáo và phân tích lỗi phân tích kỹ lưỡng các lần chạy thử nghiệm với báo cáo thử nghiệm dựa trên XML cung cấp một cái nhìn tổng quan toàn diện về toàn bộ luồng thực hiện kiểm tra.

Chương 4: Ứng dụng và thực nghiệm

Trong chương này sẽ trình bày ứng dụng thực tế một phần mềm được sử dụng công cụ Ranorex để test đó là phần mềm Keepass một phần mềm bảo vệ mật khẩu.

4.1. Giới thiệu về phần mềm Keepass

Trong thời đại công nghệ phát triển như hiện nay việc sở hữu tài khoản cá nhân không còn đơn thuần nữa. Mỗi cá nhân sử dụng rất nhiều tài khoản trên các trang khác nhau vì yêu cầu công việc và giải trí. Trường hợp bạn sử dụng cùng một mật khẩu cho tất cả các trang nếu bị lấy cắp mật khẩu thì bạn mất hết tài khoản. Trường hợp bạn lưu mỗi trang một tài khoản nên sử dụng thêm phần mềm hỗ trợ quản lý mật khẩu. KeePass Password Safe⁶ là chương trình mã nguồn mở nhỏ gọn để quản lý Password trong Windows. Keepass là một phần mềm miễn phí sử dụng mã nguồn mở (chứng nhận của OSI). Chương trình này dùng cơ sở dữ liệu được mã hóa cho phép dùng một khóa quan trọng (key file) hay một mã Password duy nhất thay thế cho rất nhiều Password khác nhau của người dùng trong những ứng dụng, cửa sổ khác nhau [5].

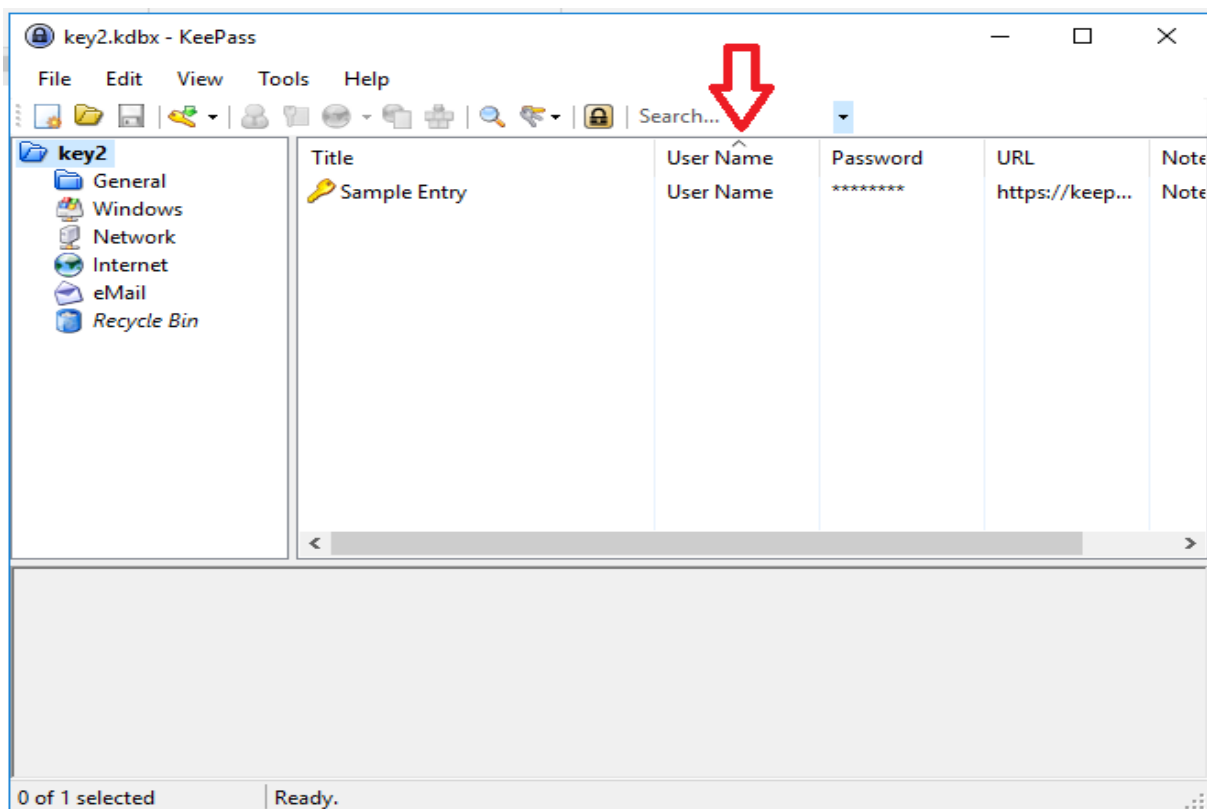
4.2 Thực nghiệm

Đầu tiên trước khi thực nghiệm cần mở và chạy chương trình Keepass. Nhấn Start của sổ tiếp theo sẽ hiển thị. Vì Keepass là một phần mềm mã nguồn mở nên chúng ta có thể chỉnh sửa để có nhiều phiên bản khác nhau trong quá trình kiểm thử.

Thực nghiệm 1: Kiểm tra chức năng di chuyển con trỏ tới phần tử yêu cầu.

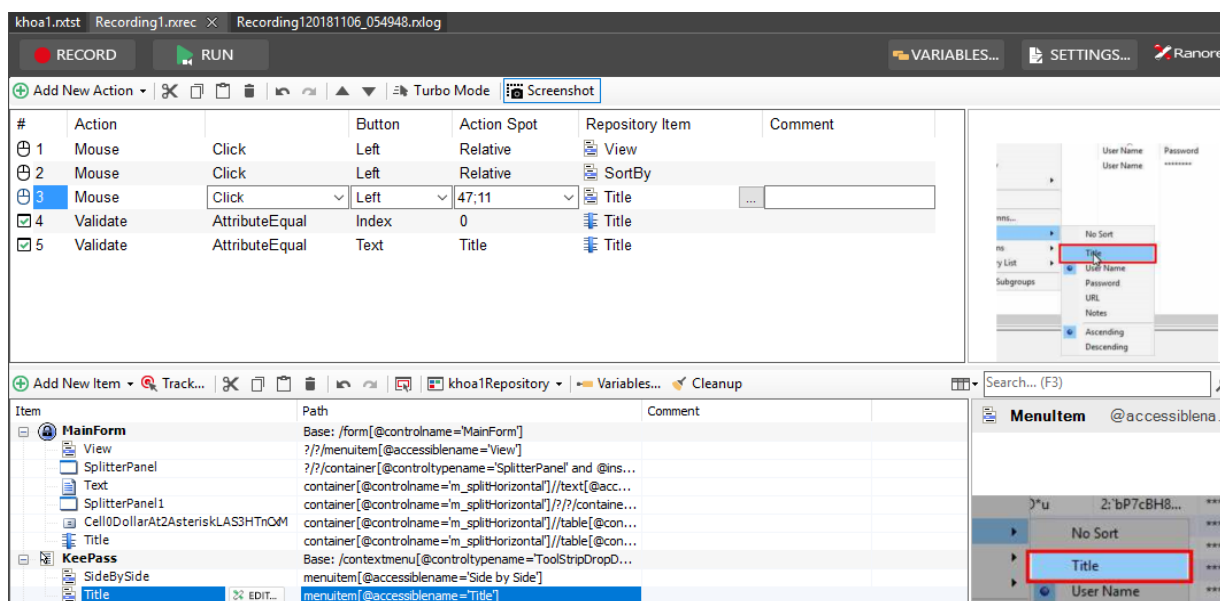
Sau khi mở chương trình thực nghiệm Bước 1: vào View trên thanh công cụ Bước 2: chọn Sort by sau đó Bước 3: click chuột vào mục Title. Mong muốn kết quả đầu ra là trỏ chuột thay đổi vị trí từ vị trí ban đầu ở mục User name chuyển sang mục Title.

⁶ <https://keepass.info/>



Hình 4.1. Giá trị ban đầu khi thực hiện thử nghiệm

Sử dụng Ranorex để kiểm tra (test) chương trình dựa trên thử nghiệm đầu vào và đầu ra như mong muốn nếu kết quả là không có lỗi nghĩa là chương trình đã không thể thay đổi vị trí trỏ chuột như mong muốn của người dùng từ đó cần báo cho lập trình viên sửa lỗi của chương trình.



Hình 4.2. Các hành động được ghi khi thực hiện thử nghiệm

Hình 4.2 khi thực hiện thử nghiệm từng hành động đều được ghi lại một cách rõ ràng bởi Ranorex. Từng bước click chuột hay các giá trị đầu vào của trường chỉnh luôn đi kèm hình ảnh hiển thị.

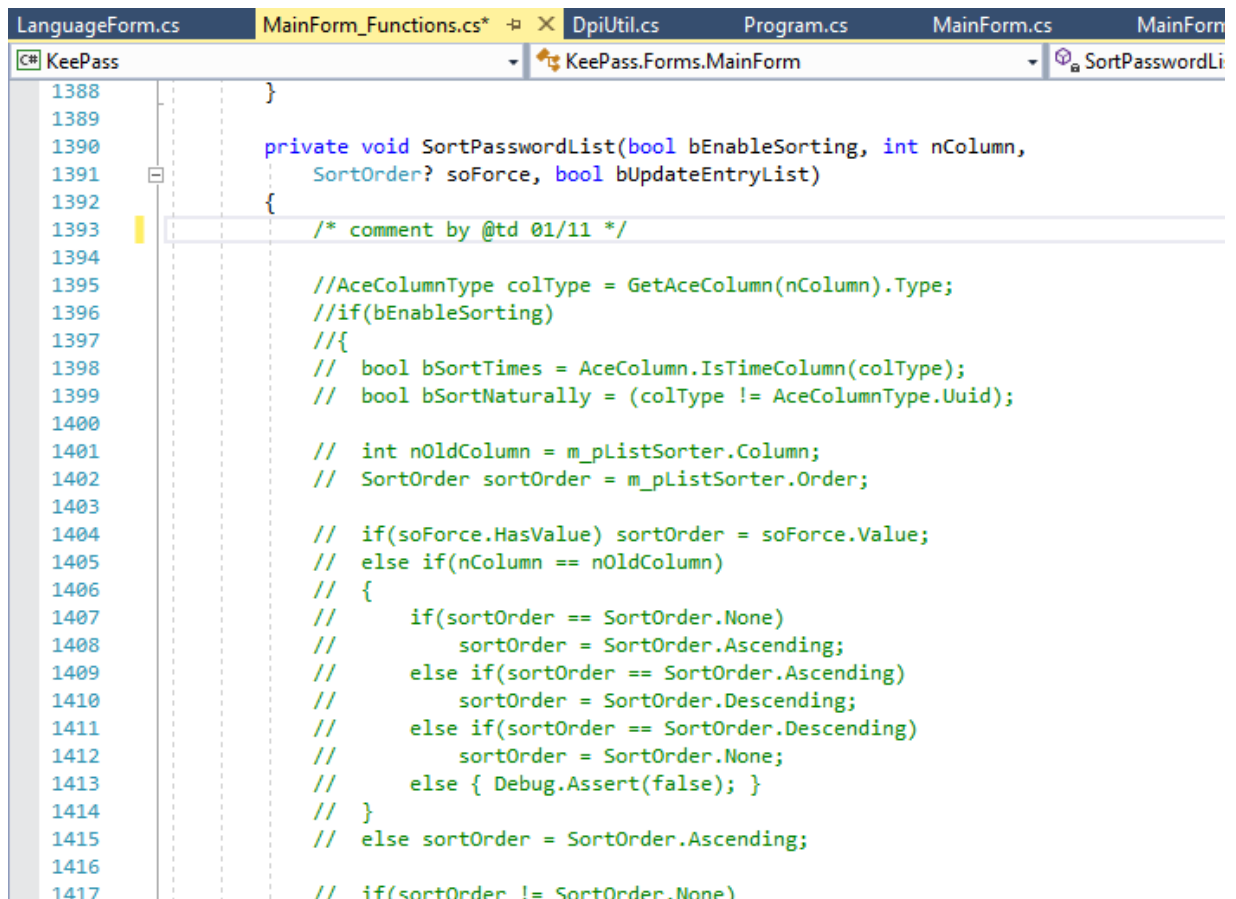
Recording1
Success
Execution time
11/6/2018 5:49:53 AM
Operating system
Windows 10 64bit
OS Language
en-US
Total errors
0
Computer name
ADMIN
Screen dimensions
1366x768
Total warnings
0

Filter: ☒ Info ☒ Warn ☒ Success

Time	Level	Category	Message
00:00.667	Warn	Data	The following module variables are not bound to a data column: NewVariable
00:00.691	Info	Mouse	Mouse Left Click item 'MainForm.View' at 27;11.
00:02.841	Info	Mouse	Mouse Left Click item 'KeePass1.SortBy' at 95;9.
00:03.775	Info	Mouse	Mouse Left Click item 'KeePass.Title' at 47;11.
00:04.793	Info	Validation	Validating AttributeEqual (Index='0') on item 'MainForm.Title'.
00:05.160	Success	Validation	Attribute 'Index' of element for item 'khoa1Repository.MainForm.Title' does match the specified value.
00:05.289	Info	Validation	Validating AttributeEqual (Text='Title') on item 'MainForm.Title'.
00:05.601	Success	Validation	Attribute 'Text' of element for item 'khoa1Repository.MainForm.Title' does match the specified value.

Hình 4.3. Kết quả chạy thử nghiệm từ thử nghiệm

Kết quả chạy thử nghiệm giống như kết quả đầu vào mong muốn điều này chứng tỏ chương trình có lỗi ở vị trí con trỏ mũi tên di chuyển trong cửa sổ hiển thị.



```
1388 }
1389
1390 private void SortPasswordList(bool bEnableSorting, int nColumn,
1391     SortOrder? soForce, bool bUpdateEntryList)
1392 {
1393     /* comment by @td 01/11 */
1394
1395     //AceColumnType colType = GetAceColumn(nColumn).Type;
1396     //if(bEnableSorting)
1397     //{
1398         // bool bSortTimes = AceColumn.IsTimeColumn(colType);
1399         // bool bSortNaturally = (colType != AceColumnType.Uuid);
1400
1401         // int nOldColumn = m_pListSorter.Column;
1402         // SortOrder sortOrder = m_pListSorter.Order;
1403
1404         // if(soForce.HasValue) sortOrder = soForce.Value;
1405         // else if(nColumn == nOldColumn)
1406         // {
1407             // if(sortOrder == SortOrder.None)
1408             //     sortOrder = SortOrder.Ascending;
1409             // else if(sortOrder == SortOrder.Ascending)
1410             //     sortOrder = SortOrder.Descending;
1411             // else if(sortOrder == SortOrder.Descending)
1412             //     sortOrder = SortOrder.None;
1413             // else { Debug.Assert(false); }
1414         // }
1415         // else sortOrder = SortOrder.Ascending;
1416
1417         // if(sortOrder != SortOrder.None)
```

Hình 4.4. Đoạn mã chứa lỗi

Hình 4.4 hiển thị đoạn mã chứa lỗi di chuyển con trỏ như trên đã mô tả. Giờ lập trình viên sửa mã này sẽ có được chương trình thực hiện không còn lỗi di chuyển con trỏ mũi tên nữa. Toàn bộ đoạn mã (code) được sử dụng comment là đoạn mã thể hiện chức năng di chuyển con trỏ trong giao diện phần mềm.

Chương 5. Kết luận

Tự động hóa quá trình kiểm thử nói chung và kiểm thử tương tác giao diện người dùng được xem là giải pháp hiệu quả góp phần giải quyết được hai vấn đề bao gồm đảm bảo chất lượng và giảm chi phí, thời gian trong quá trình phát triển chương trình phần mềm. Đã có nhiều giải pháp và công cụ được đề xuất nhằm thực hiện hóa mục tiêu này như kiểm thử tương tác giao diện người dùng sử dụng công cụ Ranorex hay sử dụng các công cụ ghi và chạy lại các kịch bản tương tác UI. Tuy nhiên, chi phí để sử dụng công cụ Ranorex khá lớn, công cụ như Ranorex chỉ hỗ trợ ghi và chạy mà không hỗ trợ sinh các kịch bản tương tác UI.

Luận văn đã tiến hành nghiên cứu về kiểm thử tự động, kiểm thử tương tác giao diện người dùng nhằm củng cố các kiến thức nền tảng. Luận văn đi sâu tìm hiểu một số công cụ hỗ trợ kiểm thử tương tác giao diện người dùng nhằm chỉ ra những điểm nổi bật và hạn chế của từng công cụ. Từ những khảo sát này, luận văn nhận thấy Ranorex là bộ công cụ có nhiều tính năng nổi bật nhất. Ngoài khả năng xác định các đối tượng UI chính xác, Ranorex còn cung cấp tính năng cho phép kiểm thử viên chỉnh sửa kịch bản sử dụng các đoạn mã giúp cho việc kiểm thử linh hoạt hơn. Ranorex là công cụ kiểm thử giao diện người dùng được cho là nổi trội nhất hiện nay cả về mặt tính năng cũng như mức độ thân thiện với người sử dụng. Luận văn cũng đã áp dụng kiến thức tìm hiểu thực hiện kiểm thử một số phiên bản cho một ứng dụng phần mềm sử dụng công cụ Ranorex nhằm minh chứng cho những hiểu biết đã nghiên cứu.

Mặc dù đã có nhiều cố gắng trong thời gian thực hiện luận văn nhưng với kinh nghiệm và kiến thức về công nghệ còn hạn chế nên luận văn không tránh khỏi những thiếu sót. Sự áp dụng những kiến thức tìm hiểu được mới chỉ dừng lại ở một ứng dụng nhỏ, mà vẫn chưa thử áp dụng cho các bài toán hay ứng dụng lớn. Sự so sánh mới chỉ dừng lại ở ba công cụ Selenium, QTP, Ranorex chứ chưa mở rộng được nhiều công cụ. Trong quá trình áp dụng thử nghiệm, học viên sẽ chủ động cây lỗi cho ứng dụng để có được các phiên bản khác nhau.

Tuy nhiên, các phiên bản áp dụng hiện tại mới chỉ chứa một số lỗi cơ bản, với các tình huống có thể gây lỗi đối với đối tượng giao diện là textbox. Luận văn đang được tiếp tục phát triển để xử lý với các đối tượng khác như button, dateandtime, checkbox, v.v. Ngoài ra, việc cho phép kiểm thử viên tự định nghĩa một số tình huống có thể gây lỗi mới giúp cơ sở dữ liệu đầy đủ hơn. Luận văn cũng sẽ tiếp tục nghiên cứu thêm các công cụ hỗ trợ kiểm thử tự động khác như kế hoạch kiểm thử (Test Plan), kịch bản kiểm thử (Test Case) và tiến hành tích hợp vào Ranorex, tối đa sự thuận tiện cho kiểm thử viên. Luận văn cũng sẽ cố gắng tạo ra một giao diện mở với tài liệu đầy đủ để người dùng viết các tính năng thêm vào (plug-in) của riêng mình.

TÀI LIỆU THAM KHẢO

Tiếng Anh

- [1] Boris Beizer and Van Nostrand Reinhold (1990), Software Testing Techniques, Second Edition.
- [2] <https://www.ranorex.com/>
- [3] Glenford J. Myers, Corey Sandler, and Tom Badgett (2011), *The Art of Software Testing* (3rd ed.). Wiley Publishing.
- [4] Glenford J. Myers (1979), “The Psychology and Economics of Program Testing”, *The art of software testing*, pp. 11.
- [5] <http://thuthuatphanmem.vn/huong-dan-quan-ly-mat-khau-bang-keepass/>
<http://conganbackan.vn/bao-mat-may-tinh/trai-nghiem-keepass-trinh-quan-ly-mat-khau-an-tuong-24792.html>