ĐẠI HỌC QUỐC GIA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN

NGUYỄN THỊ HÒA

NGHIÊN CỨU VÀ ỨNG DỤNG GIẢI PHÁP KIỂM THỬ TỰ ĐỘNG PHẦN MỀM

Ngành: Công nghệ thông tin
Chuyên ngành: Quản lý Hệ thống thông tin
Mã số: Chuyên nghành đào tạo thí điểm

LUẬN VĂN THẠC SĨ

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. ĐINH VĂN DŨNG

HÀ NỘI - 2014

LỜI CAM ĐOAN

Tôi xin cam đoan rằng, đây là kết quả nghiên cứu của tôi trong đó có sự giúp đỡ rất lớn của thầy hướng dẫn và các đồng nghiệp ở cơ quan. Các nội dung nghiên cứu và kết quả trong đề tài này hoàn toàn trung thực.

Trong luận văn, tôi có tham khảo đến một số tài liệu của một số tác giả đã được liệt kê tại phần "Tài liệu tham khảo" ở cuối luận văn.

Tác giả luận văn

Nguyễn Thị Hòa

LỜI CẢM ƠN

Em xin chân thành cảm ơn đến Viện Công Nghệ thông tin, Đại học công nghệ, Đại học Quốc gia Hà Nội đã tạo điều kiện cho em học tập và thực hiện luận văn này.

Em xin gửi lời cảm ơn chân thành đến Tiến Sĩ Đinh Văn Dũng, người đã tận tình hướng dẫn em trong quá trình làm luận văn này.

Em xin cảm ơn quý Thầy Cô đã nhiệt tình giảng dạy cho chúng em trong những năm học vừa qua.

Cuối cùng, em xin được gửi lời cảm ơn chân thành đến gia đình cũng như bạn bè đã luôn ủng hộ, động viên em để em có thể có điều kiện tốt nhất để học tập và nghiên cứu.

Hà Nội, tháng 12/2014

Nguyễn Thị Hòa – Lớp CIO 03

Viện Công nghệ thông tin – Đại học Quốc gia Hà Nội

MỤC LỤC

BẢNG CÁC TỪ VIẾT TẮT

#	Thuật ngữ	Ý nghĩa
1	GUI	Graphical user interface
2	HTKTTĐ	Hệ thống kiểm thử tự động
3	TSV	Tab-Separated Values
4	CSV	Comma-Separated Values
5	HTML	HyperText Markup Language
6	SAP	System Applications Products

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

MỞ ĐẦU

Sự cần thiết của đề tài

Chúng ta đã và đang chứng kiến sự tăng trưởng đáng kinh ngạc của nghành công nghiệp phần mềm trong vài thập kỉ qua. 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. Từ các ứng dụng nhỏ để điều khiển các thiệt 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. Có thể nói, công nghiệp phần mềm đã len lỏi đến từng ngóc nghách nhỏ nhất của đời sống con người, đòi hỏi chất lượng phần mềm ngày một nâng cao hơn. Đồng nghĩa với việc cần phải kiểm thử phần mềm chặt chẽ để có thể đảm bảo chất lượng của phần mềm.

Kiểm thử phần mềm là khâu sống còn 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. Nhất là đối với các phần mềm lớn, chi phí này càng tăng lên gấp bội mỗi khi có sự thay đổi, nâng cấp các chức năng của phần mềm. Mà điều này thì không thể tránh khỏi, phần mềm luôn cần được thay đổi để đáp ứng yêu cầu ngày một cao hơn của người sử dụng. Khi có sự thay đổi của phần mềm, đồng nghĩa ngoài việc kiểm thử chức năng mới, các chức năng cũ cũng cần được kiểm tra kỹ càng để đảm bảo chúng vẫn hoạt động tốt . Đó chính là hoạt động kiểm thử hồi qui.

Hiện tại, kiểm thử hồi qui tại các công ty nhỏ và vừa ở trong nước chủ yếu được thực hiện bởi kiểm thử thủ công. Nhiều khi chức năng thay đổi nhỏ nhưng phần cần thực hiện kiểm thử lại rất lớn, bên cạnh việc tốn kém chi phí, nhân lực, cũng có khả năng có thể chậm tiến độ, bị lọt lỗi khi bàn giao sản phẩm. Do đó, luận văn mong muốn đưa ra giải pháp tự động hóa kiểm thử nhằm giảm thiểu chi phí

kiểm thử, cả về thời gian, tiền bạc, con người, và giảm sự nhàm chán cho kiểm thử viên mà vẫn đảm bảo được chất lượng của sản phẩm.

Với sự phát triển mạnh mẽ của phát triển phần mềm cũng như là kiểm thử phần mềm hiện nay, có rất nhiều công cụ hỗ trợ cho kiểm thử tự động, mỗi công cụ có thể có một số phương pháp luận khác nhau. Nhưng điều đó không đồng nghĩa với việc lựa chọn công cụ bất kỳ nào cũng tốt, hoặc cứ áp dụng kiểm thử tự động là có thể tiết kiệm chi phí và đảm bảo dự án sẽ thành công. Vì vậy luận văn mong muốn đưa ra một cái nhìn tổng quan nhất trong việc nghiên cứu áp dụng tự động hóa trong kiểm thử phần mềm hiện nay, các thế hệ công cụ kiểm thử tự động cũng như nghiên cứu hai phương pháp luận tự động hóa kiểm thử hướng dữ liệu và hướng từ khóa. Cuối cùng, luận văn sẽ áp dụng kiểm thử tự động trong kiểm thử chức năng của phần mềm Ads Editor với công cụ Robot Framework.

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 và kiểm thử tự động - một giải pháp góp phần nâng cao năng suất, chất lượng hoạt động kiểm thử phần mềm.

– Luận văn mô tả phương pháp kiểm thử hướng dữ liệu và phương pháp kiểm thử hướng từ khóa. Nền tảng lý thuyết này sẽ được thử nghiệm trong luận văn này.

– Luận văn đã mô tả từng bước quá trình áp dụng kiểm thử hướng dữ liệu và hướng từ khóa vào kiểm thử một hệ thống trong thực tế, góp phần giảm chi phí việc kiểm thử một số sản phẩm phần mềm.

Cấu trúc của luận văn

Với mục tiêu xây dựng giải pháp tự động hóa cho kiểm thử hồi qui, luận văn được chia làm bốn chương:

Chương I: Tổng quan kiểm thử tự động

Chương này giới thiệu về khái niệm kiểm thử, kiểm thử tự động, vai trò và lợi ích khi ứng dụng kiểm thử tự động trong hoạt động kiểm thử phần mềm. Chương này cũng trình bày các bước để tiếp cận kiểm thử tự động cũng như các vấn đề có

thể gặp phải trong quá trình áp dụng kiểm thử tự động.Ngoài ra cũng tổng hợp về tình hình thị trường của kiểm thử tự động, tình hình nghiên cứu áp dụng kiểm thử tự động hiện nay.

Chương II: Giải pháp kiểm thử tự động hướng dữ liệu và hướng từ khóa

Từ những nghiên cứu ở Chương I, chương này giới thiệu hai giải pháp kiểm thử tự động hướng dữ liệu và hướng từ khóa.

Chương III: Thử nghiệm kiểm thử hướng dữ liệu và từ khóa

Chương này giới thiệu sơ lược với bạn đọc về phần mềm quản lý quảng cáo trực tuyến Ads Editors. Lý do cần thiết phải xây dựng hệ thống kiểm thử tự động để kiểm thử cho hệ thống Ads Editors. Đưa ra các bước xây dựng kiểm thử hướng dữ liệu và hướng từ khóa trong việc áp dụng kiểm thử tự động chức năng "Post changes/ Get changes".

Chương IV: Kết luận và khuyến nghị

Trong chương này, chúng tôi sẽ tổng kết lại các kết quả và đóng góp mà việc thực hiện đề tài đem lại. Ngoài ra, chúng tôi cũng đề xuất các phương hướng nghiên cứu tiếp theo, nhằm giúp cho đề tài trở nên hoàn thiện hơn.

CHƯƠNG 1. TỔNG QUAN KIỂM THỬ TỰ ĐỘNG PHẦN MỀM

1.1 Giới thiệu

Khái niệm

Kiểm thử phần mềm là qui trình thực hiện một chương trình hay hệ thống với mục đích tìm lỗi, như Myer định nghĩa [1]. Có kiểm thử, chúng ta có thể đánh giá được các yêu cầu về chức năng cũng như yêu cầu phi chức năng (tính tin cậy, tính khả dụng, tính hiệu quả, khả năng bảo trì, bảo mật, tính di động) của phần mềm. Lỗi phần mềm càng được tìm ra muộn, nhất là sau khi sản phẩm đã 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.

Hiện nay, các công cụ hỗ trợ lập trình đã giúp tăng cải thiện năng suất làm việc của các lập trình viên lên rất nhiều. Điều này dẫn đến tăng áp lực lên các kiểm thử viên, những người thường đứng ở vị trí nút cổ chai trong việc bàn giao sản phẩm phần mềm. Đòi hỏi kiểm thử viên phải kiểm thử nhiều hơn trong khoảng thời gian ít hơn. Nên việc cần làm với các kiểm thử viên đó là tìm ra cách để vừa đảm bảo chất lượng của phần mềm, đồng thời phải rút ngắn thời gian kiểm thử, đồng nghĩa với việc tăng năng suất kiểm thử. Cũng giống như rất nhiều nghành nghề khác, khi nghĩ đến việc tăng năng suất lao động, người ta nghĩ đến tự động hóa. Vậy kiểm thử tự động là gì?

Kiểm thử tự động là quá trình thực hiện một cách tự động các bước trong một kịch bản kiểm thử. Mục đích của kiểm thử tự động là giảm thiểu thời gian, công sức và kinh phí, tăng độ tin cậy, tăng tính hiệu quả và giảm sự nhàm chán cho kiểm thử viên trong quá trình kiểm thử phần mềm. [2]

Kiểm thử phần mềm hay kiểm thử kiểm thử phần mềm tự động có thể được phân chia thành kiểm thử tĩnh và kiểm thử động, trong đó kiểm thử động bao gồm kiểm thử chức năng và kiểm thử phi chức năng. Mỗi loại kiểm thử đều đóng vai trò quan trọng trong đảm bảo chất lượng phần mềm, như kiểm thử tĩnh là thực hiện kiểm thử ở giai đoạn sớm của quá trình phát triển phần mềm, phát hiện các lỗi trên

các tài liệu thiết kế, mã nguồn... Kiểm thử động được thực hiện khi mã nguồn được thực thi, nhằm phát hiện ra các lỗi về chức năng như phần mềm có hoạt động như thiết kế không, hoặc các lỗi phi chức năng như phần mềm có hoạt động như mong muốn của người dùng không. Do có rất nhiều thuật ngữ liên quan đến kiểm thử và kiểm thử tự động không thể giới thiệu hết trong phạm vi của luận văn. Nên luận văn chỉ đề cập đến các khái niệm kiểm thử chức năng và kiểm thử hồi qui, tập trung vào áp dụng kiểm thử tự động các chức năng ở trong giai đoạn kiểm thử hồi qui.

Kiểm thử chức năng

Giống như tên gọi của nó, kiểm thử chức năng đảm bảo rằng các chức năng của phần mềm sẽ hoạt động đúng như yêu cầu trong đặc tả phần mềm, Chức năng của phần mềm là những gì mà nó được xây dựng để có thể làm được. Vậy kiểm thử chức năng sẽ là kiểm tra xem phần mềm có thể làm được các yêu cầu có trong đặc tả hay ca sử dụng (use cases). Có thể có một số chức năng mà cũng được giả sử sẽ thỏa mãn mặc dù không được nhắc đến trong tài liệu nhưng được ngầm hiểu bởi kiểm thử viên. Kiểm thử chức năng có thể được thực hiện ở tất cả các mức kiểm thử: kiểm thử đơn vị, kiểm thử tích hợp, kiểm thử hệ thống, kiểm thử chấp nhận…

Dựa theo tiêu chuẩn ISO 9126, kiểm thử chức năng có thể tập trung vào tính phù hợp (suitability), tính tương tác (interoperability), tính bảo mật (security), tính chính xác (accuracy) và tính tuân thủ (compliance).

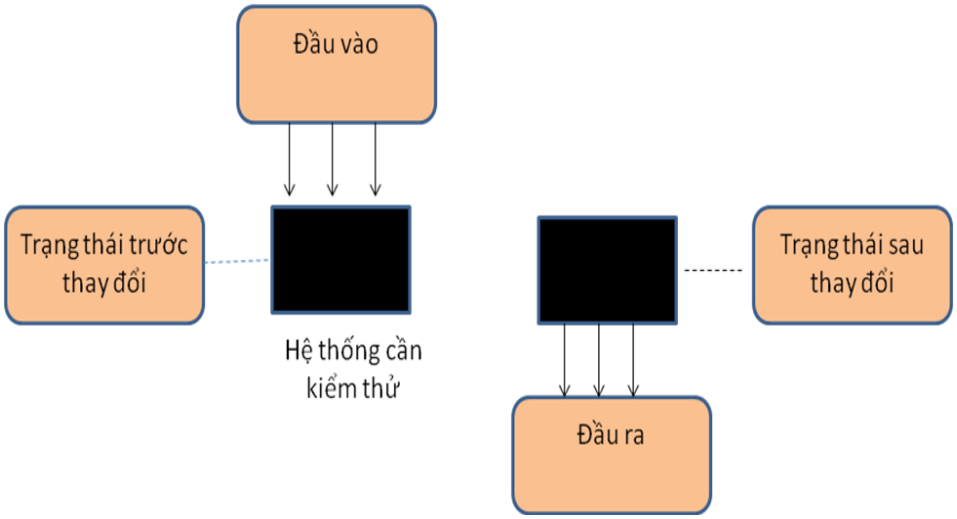
Trong đó tính phù hợp đó là khả năng phần mềm có thể cung cấp các tập hợp các chức năng phù hợp cho một tác vụ cụ thể và mục tiêu người dùng. Tính tương tác đó là khả năng phần mềm có thể tương tác với một hoặc nhiều thành phần cụ thể hoặc hệ thống. Tính bảo mật là khả năng phần mềm có thể ngăn chặn truy cập trái phép, dù vô tình hay cố ý các chương trình và dữ liệu. Tính chính xác là khả năng của phần mềm có thể cung cấp các quyền hay những đồng thuận về kết quả hoặc ảnh hưởng với mức độ cần thiết của độ chính xác.

Kiểm thử chức năng có thể thực hiện theo hai xu hướng: Dựa trên yêu cầu hoặc dựa trên qui trình nghiệp vụ.

Kiểm thử chức năng dựa trên yêu cầu sử dụng đặc tả của hệ thống là tài liệu cơ bản để thiết kế các bài kiểm thử. Một cách bắt đầu tốt đó là sử dụng bảng mục lục của tài liệu đặc tả để liệt kê ra danh sách các đầu mục cần thực hiện kiểm thử (hoặc không kiểm thử). Chúng ta cần đánh độ ưu tiên cho các bài kiểm thử để đảm bảo rằng những phần yêu cầu quan trọng nhất chắc chắn được thực hiện kiểm thử.

Kiểm thử dựa trên qui trình nghiệp vụ sử dụng kiến thức liên quan đến qui trình nghiệp vụ. Qui trình nghiệp vụ miêu tả kịch bản sử dụng chức năng hàng ngày của hệ thống. Sử dụng tài liệu ca sử dụng rất phù hợp cho việc tạo các bài kiểm thử theo hướng nghiệp vụ. [3]

Một cách tổng quát, kiểm thử chức năng có thể được hoàn thành trong hai bước như minh họa ở Hình 1. Đầu tiên hệ thống cần kiểm thử được nhập giá trị đầu vào. Sau đó kết quả đầu ra của phần mềm và sự thay đổi trong trạng thái của phần mềm được so sánh với kết quả mong muốn mà đã được định nghĩa từ trước. Trong hầu hết trường hợp, các bước này đều có thể được tự động hóa.[4]



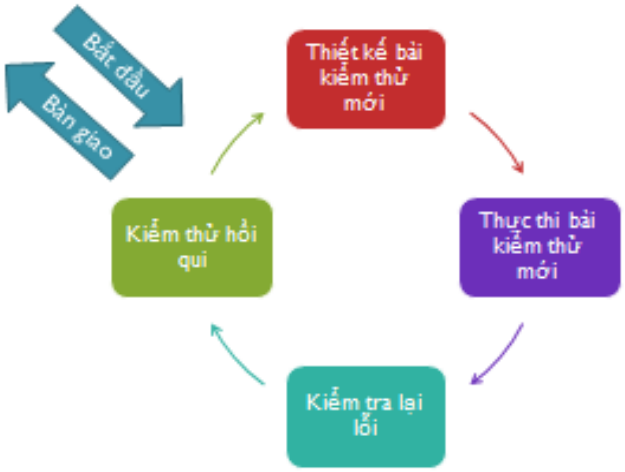
Hình 1: Kiểm thử chức năng ở góc nhìn tổng quan (High level view)

Kiểm thử hồi qui

Như đã nói ở trên, khi xây dựng một phiên bản mới của hệ thống như sửa lỗi, thêm chức năng, chúng ta có thể vô tình gây ra các lỗi mới. Nhiều khi có thay đổi rất nhỏ nhưng cũng gây ra vấn đề rất lớn ngoài sức tưởng tượng của đội phát triển.

Để tránh những sự việc đáng tiếc này xảy ra, chúng ta cần kiểm thử lại phần mềm để đảm bảo chức năng đã chạy tốt vẫn tiếp tục chạy tốt.

Khi phiên bản mới của phần mềm không còn hoạt động như trước chúng ta nói là phiên bản mới hồi qui với bản trước đó. Phiên bản mới không hồi qui tức là nó vẫn giữ được các tính năng là một yêu cầu chất lượng cơ bản. Các hoạt động kiểm thử tập trung vào vấn đề hồi qui được gọi là kiểm thử hồi qui. Đúng ra chúng ta cần kiểm tra phiên bản mới không hồi qui về phiên bản cũ, nhưng thuật ngữ kiểm thử hồi quy đã phổ biến nên chúng ta dùng thuật ngữ này. Kiểm thử hồi qui được minh họa như hình dưới đây.



Hình 2: Kiểm thử hồi qui

Phương pháp đơn giản để kiểm thử hồi quy là chạy lại tất cả các ca kiểm thử của phiên bản trước. Tuy nhiên việc kiểm lại toàn bộ này rất tốn kém và không tầm thường. Các bài kiểm thử trước đó có thể không chạy lại được với phiên bản mới của phần mềm mà không sửa đổi gì. Chạy lại toàn bộ các bài kiểm thử là rất tốn kém và nhiều phần trong đó là không cần thiết. Một bộ kiểm thửu chất lượng tốt phải được duy trì xuyên suốt các phiên bản của hệ thống.

Thay đổi trong phiên bản phần mềm mới có thể tác động tới khuôn dạng của đầu vào và đầu ra, và các bài kiểm thử có thể cần các thay đổi tương ứng để chạy được. Ngay cả việc sửa đổi một cách đơn giản của cấu trúc dữ liệu, chẳng hạn như bổ xung các trường có thể làm cho các bài kiểm thử trước đây không chạy được

nữa. Hơn nữa một số bài kiểm thử có bị lỗi thời, khi các tính năng của phần mềm đã thay đổi hoặc bị loại bỏ khỏi phiên bản mới. Các khung hỗ trợ để dịch đặc tả kiểm thử thành bài kiểm thử sẽ giúp giảm ảnh hưởng của thay đổi định dạng của đầu vào và đầu ra. Các đặc tả bài kiểm thử và các mệnh đề về kết quả mong đợi mà phản ánh tính đúng của bài kiểm thử và tránh các chi tiết cụ thể sẽ giảm một phần lớn công sức kiểm thử khi có thay đổi nhỏ. Các bộ kiểm thử chất lượng cao có thể được duy trì qua các phiên bản của phần mềm bằng việc xác định và loại bỏ các bài kiểm thử lỗi thời và phát hiện và đánh dấu thích hợp các bài kiểm thử dư thừa. Ca kiểm thử đi cùng một đường đi trong chương trình là dư thừa với tiêu chuẩn kiểm thử cấu trúc, nhưng bài kiểm thử cùng một phân hoạch lại là dư thừa với kiểm thử dựa trên lớp tương đương. Việc dư thừa là do nhiều người cùng làm hoặc do chương trình bị thay đổi gây ra. Các bài kiểm thử dư thừa không ảnh hưởng đến tính đúng, sai, mà chỉ ảnh hưởng đến chi phí kiểm thử. Các bài kiểm thử lỗi thời cần phải loại bỏ, vì chúng ta không còn cần chúng còn những ca kiểm thử dư thừa chúng ta có thể giữ, vì chúng giúp phần mềm phát triển trong tương lai đảm bảo chất lượng hơn, tuy nhiên nếu chúng gây ra chi phí kiểm thử lớn quá mức cần thiết thì có thể loại bỏ. Một cách tiếp cận khác đó là thực hiện loại bỏ cả những bài kiểm thử mà không gặp lỗi trong một thời gian dài, mặc dù cần phải xem xét cẩn trọng hướng tiếp cận này. [5]

Để góp phần nâng cao hiệu quả kiểm thử hồi qui có thể vừa kết hợp kỹ thuật lựa chọn các bài kiểm thử như đã nói ở đây, cùng với việc thực hiện các bài kiểm thử một cách tự động. Tuy nhiên, trong phạm vi của luận văn này, sẽ chỉ tập trung vào việc thực thực thi các bài kiểm thử trong kiểm thử hồi qui một cách tự động.

1.2 Qui trình kiểm thử tự động

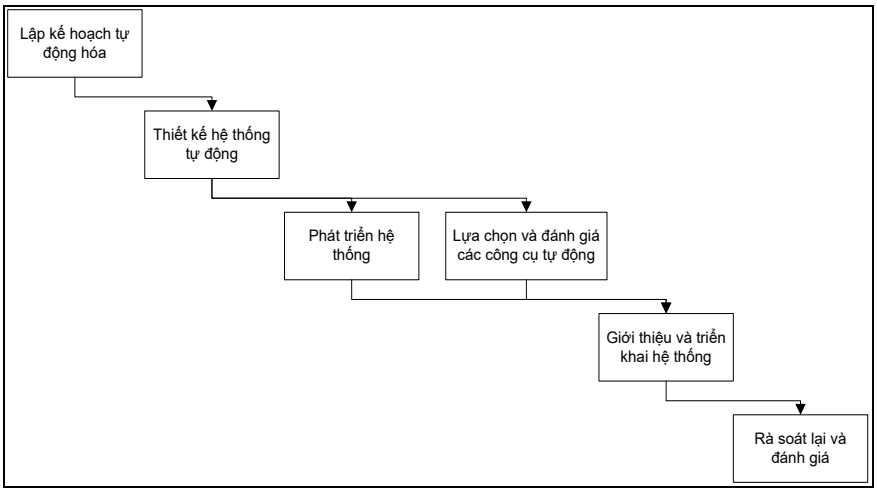
Kiểm thử tự động sẽ được sử dụng khi dự án không đủ tài nguyên (thời gian, nhân lực và chi phí), như sau khi sản phẩm được sửa đổi, nâng cấp cần phải thực hiện kiểm thử hồi qui để chắc chắn rằng việc sửa đổi nâng cấp không làm ảnh hưởng đến các chức năng đã có của sản phẩm. Ngoài ra, kiểm thử tự động cũng được lựa chọn khi phải kiểm tra khả năng vận hành của sản phẩm trong các môi trường đặc biệt mà nếu dùng kiểm thử thủ công sẽ rất khó khăn để thực hiện như là

đo tốc độ xử lý, xác định khả năng chịu tải tối đa, kiểm tra cơ chế an ninh, an toàn, lặp đi lặp lại một số thao tác trong thời gian dài…

Việc kiểm thử tự động thì tốn kém hơn kiểm thử thủ công rất nhiều nếu chỉ thực hiện một lần. Để đạt được lợi ích, kiểm thử tự động cần phải được lựa chọn và thực thi một cách cẩn thận theo một tiến trình cụ thể.

Trước khi thực hiện kiểm thử tự động, kỹ sư kiểm thử và người quản lý phải có hiểu biết rõ ràng về kiểm thử tự động, bao gồm nhu cầu, mục tiêu, lợi ích, các vấn đề và thách thức.

Theo Douglas Hoffman, một tiến trình hiệu quả để thực hiện tự động hóa kiểm thử bao gồm các bước như ở hình dưới đây [6]



Hình 3 Tiến trình tự động hóa

– Bước 1: Lập kế hoạch tự động hóa. Đây là bước khởi tạo. Mục đích chính của bước này là lập kế hoạch xác định các đối tượng cần phải tự động, mục đích, chiến lược, yêu cầu, lịch trình, kinh phí. Trong thực tế, kế hoạch tự động hóa thường được tạo cho một sản phẩm, hoặc một dòng sản phẩm ngay ở giai đoạn đầu của tiến trình phát triển phầm mềm.

– Bước 2: Thiết kế kiểm thử tự động. Mục tiêu chính của bước này là tạo ra một giải pháp tự động hóa chi tiết, phù hợp với yêu cầu và mục tiêu trong kế hoạch đã vạch ra. Bao gồm 2 nhiệm vụ chính. Thứ nhất là xác định và lựa chọn các công cụ có sẵn (các sản phẩm thương mại hoặc tự làm) để hỗ trợ tiến trình tự động hóa.

Để thực hiện nhiệm vụ này, người tự động hóa cần phải có thông tin hướng dẫn và đánh giá về công cụ được lựa chọn. Thứ hai là thiết kế giải pháp tự động hóa.

– Bước 3: Phát triển công cụ kiểm thử tự động. Ở bước này, sẽ thực hiện phát triển công cụ dựa theo giải pháp tự động hóa đã đưa ra. Điểm mấu chốt ở bước này là phải chắc chắn rằng công cụ đã được phát triển là đáng tin cậy và dễ sử dụng, có tài liệu hướng dẫn tốt. Rất nhiều dự án tự động kiểm thử đã thất bại bởi vì chất lượng kém và tài liệu không tốt.

– Bước 4: Triển khai công cụ tự động. Giống như một sản phẩm thương mại, công cụ tự động và các lợi ích của nó cần phải được giới thiệu và triển khai sử dụng cho một sản phẩm nào đó. Ở bước này, việc đào tạo cho người dùng và hỗ trợ họ là thiết yếu.

– Bước 5: Rà soát lại (review) và đánh giá. Bất cứ khi nào một công cụ mới được triển khai, thì đều phải tiến hành rà soát lại để xác định các vấn đề và hạn chế của công cụ, đánh giá các tính năng mà nó cung cấp. Kết quả của việc rà soát này là bài học cho các lần triển khai về sau.

1.3 Lợi ích và thách thức của kiểm thử tự động

1.3.1 Lợi ích

Dưới đây là một số lợi ích của kiểm thử tự động 69[7]:

– Kiểm thử hồi qui cho một phiên bản mới của chương trình. Điều này là hiển nhiên, đặc biệt là trong điều kiện các chương trình thường xuyên bị thay đổi. Giả thiết rằng các bài kiểm thử đã tồn tại và đã được chạy tự động ở một phiên bản trước đó, thì ở các phiên bản tiếp theo, chỉ cần lựa chọn các bài kiểm thử phù hợp và một chút chi phí cho việc hướng dẫn sử dụng công cụ là có thể thực hiện được việc kiểm thử.

– Chạy được nhiều bài kiểm thử và thường xuyên hơn. Với việc kiểm thử tự động, sẽ có nhiều bài kiểm thử được thực hiện trong khoảng thời gian ít hơn, và do đó các bài kiểm thử cũng được thực hiện thường xuyên hơn. Điều này sẽ làm tăng cường tính tin cậy của hệ thống.

– Cho phép hoàn thành kiểm thử đối với các bài kiểm thử mà rất khó hoặc không thể khi thực hiện bằng tay. Ví dụ: việc cố gắng hoàn thành đúng như thực tế một bài kiểm thử của hệ thống với 200 người dùng cùng trực tuyến (Online) có thể không thực hiện được nếu thực hiện kiểm thử bằng tay. Nhưng 200 người dùng này có thể được giả lập bằng các công cụ kiểm thử tự động. Khi kiểm thử bằng tay, kết quả mong muốn thường là những nội dung rõ ràng mà người kiểm thử có thể quan sát. Tuy nhiên, có những thuộc tính rất khó để có thể xác nhận theo cách bình thường. Ví dụ đối tượng GUI (Graphical User Interface) có thể gây ra một sự kiện nào đó, mà ảnh hưởng của nó không được thể hiện ra ngay lập tức. Nhưng nếu sử dụng công cụ kiểm thử thì ta có thể kiểm tra được sự kiện như vậy.

– Sử dụng tài nguyên (Resources) tốt hơn. Tự động hóa giúp nâng cao độ chính sác và giải tỏa tinh thần cho người thực hiện kiểm thử. Giúp kiểm thử viên có có nhiều thời gian hơn cho việc lập kế hoạch, thiết kế bài kiểm thử…Mặt khác, với kiểm thử tự động thì máy kiểm thử có thể được sử dụng để chạy kiểm thử vào những lúc rảnh rỗi.

– Đảm bảo tính nhất quán của kiểm thử. Kiểm thử tự động sẽ được thực hiện lặp đi lặp lại một cách chính xác ở tất cả các lần kiểm thử (ít nhất là dữ liệu đầu vào sẽ không bị thay đổi, kết quả đầu ra thì có thể khác nhau do thời gian thực hiện). Điều này tạo ra sự nhất quán giữa các lần kiểm thử, điều rất khó đạt được nếu thực hiện bằng tay. Các bài kiểm thử giống nhau có thể được thực hiện trên các phần cứng, hệ điều hành hoặc cơ sở dữ liệu khác nhau. Điều này tạo nên sự nhất quán về chất lượng trên các nền tảng khác nhau của sản phẩm. Điều rất khó đạt được nếu thực hiện kiểm thử bằng tay (kiểm thử thủ công). Việc áp dụng chế độ kiểm thử tốt có thể đảm bảo các tiêu chuẩn phù hợp cho kiểm thử và phát triển. Ví dụ, công cụ kiểm thử có thể được sử dụng để kiểm tra cùng một loại tính năng đã được thực hiện theo cùng một cách ở tất cả các ứng dụng hoặc chương trình khác nhau.

– Tái sử dụng các bài kiểm thử. Có thể thực hiện kiểm thử nhiều lần mà không mất chi phí để quyết định cái gì sẽ được kiểm thử, thiết kế bài kiểm thử, xây dựng các bài kiểm thử hay đảm bảo tính chính xác của kiểm thử.

– Rút ngắn thời gian phát triển sản phẩm. Khi mà các bài kiểm thử được kiểm thử tự động, nó có thể được thực hiện lặp đi lặp lại một cách nhanh chóng, vì vậy sẽ rút ngắn được thời gian kiểm thử. Và qua đó cũng có thể rút ngắn được thời gian phát triển sản phẩm và đưa sản phẩm ra thị trường (điều này còn tùy thuộc vào việc khắc phục lỗi, tính khả dụng của chương trình)

– Tăng tính tin cậy. Khi một số lượng lớn các bài kiểm thử tự động được thực hiện thành công, nó sẽ tăng cường mức độ đảm bảo rằng hệ thống sẽ được phát hành mà không có vấn đề gì.

Tóm lại, với kiểm thử tự động, chúng ta có thể thực hiện kiểm thử với chi phí ít hơn, và chất lượng, năng suất cao hơn.

1.3.2 Thách thức của kiểm thử tự động

Bên cạnh những lợi ích của kiểm thử tự động, cũng có rất nhiều thách thức và điểm yếu. Hầu hết các dự án tự động bị thất bại do các mong muốn thiếu thực tế, thiếu thực hành, các giả thiết đặt ra không chuẩn hoặc các vấn đề về kỹ thuật.

Một vấn đề về kỹ thuật đối với một kỹ sư kiểm thử đó là phải xác định nên tự động hóa cái gì. Các thất bại phổ biến nhất thường là cố gắng thực hiện tự động hóa quá nhiều. Không thể thực hiện tự động hết tất cả các hoạt động kiểm thử hay các bài kiểm thử. Luôn luôn có một số hoạt động kiểm thử mà thực hiện thủ công dễ hơn rất nhiều so với kiểm thử tự động, hoặc hoạt động đó rất khó để có thể tự động hóa. Các kiểm thử mà được tiến hành rất ít, hoặc các kiểm thử mà kết quả của nó rất dễ được xác nhận khi kiểm thử thủ công, nhưng lại rất khó khi tự động, hoặc phần mềm được kiểm thử không ổn định, hay khi thực hiện kiểm thử tính khả dụng.

Thực hiện tự động hóa kiểm thử quá sớm không phải là một ý kiến hay, bởi vì đặc tả của phần mềm thường có xu hướng thay đổi nhiều lần trong giai đoạn phát triển. Trong trường hợp xấu nhất, kết quả sẽ là có rất nhiều các kiểm thử tự động mà rất khó để bảo trì.

Chỉ đơn giản mua một công cụ tự động không đảm bảo đạt được các lợi ích, giống như chỉ mua một máy tập thể hình sẽ không đảm bảo rằng bạn sẽ giảm cân và

có thân hình như ý muốn. Mỗi tổ chức cần đầu tư công sức và thời gian để có thể đạt được các lợi ích từ kiểm thử tự động.

Có rất nhiều rủi ro khi các công cụ hỗ trợ kiểm thử được sử dụng, không kể đó là công cụ nào, các rủi ro bao gồm:

- Đặt ra mong muốn sử dụng công cụ không thực tế
- Đánh giá quá thấp thời gian, giá thành và công sức phải bỏ ra khi bắt đầu áp dụng công cụ tự động
- Đánh giá thấp các công sức cần thiết để bảo trì các tài sản kiểm thử mà được sinh tự động từ công cụ
- Phụ thuộc quá mức vào công cụ.

Kỳ vọng không thực tế có lẽ là một trong những rủi ro lớn nhất khi sử dụng công cụ tự động. Các công cụ cũng chỉ là phần mềm và chúng ta đều biết rằng bất kỳ phần mềm nào cũng có rất nhiều vấn đề. Định hướng rõ ràng về mục tiêu sử dụng của công cụ và các mục tiêu đó phải thực tế là vô cùng quan trọng để đạt được thành công khi tự động hóa kiểm thử.

Rõ ràng, công cụ không phải là phép màu. Chúng có thể hoàn thành tốt công việc mà chúng được thiết kế để làm, hoặc ít nhất là có thể hoàn thành, nhưng chúng không làm được tất cả. Một công cụ có thể giúp đỡ, nhưng không thể thay thế được sự thông minh cần thiết để biết cách sử dụng công cụ một cách tốt nhất, và làm sao để đánh giá việc sử dụng hiện tại cũng như trong tương lai của công cụ. Ví dụ, công cụ thực thi kiểm thử (Test excecution tool) không thể thay thế được công việc thiết kế kịch bản kiểm thử và không thể thay thế được toàn bộ các hoạt động kiểm thử - Một số hoạt động mà kiểm thử thủ công sẽ tốt hơn kiểm thử tự động, như là những bài kiểm thử mà mất rất nhiều thời gian để tự động nhưng lại không phải tiến hành thường xuyên.[3]

1.4 Thị trường kiểm thử tự động

Trên thế giới

Một nghiên cứu thị trường được thực hiện trong tháng 11 & tháng 12 năm 2012 ở một số nước ở Bắc Mỹ và châu Âu về xu hướng kiểm thử tự động cho hệ

thống doanh nghiệp trong năm 2013 đã chỉ ra rằng kiểm thử tự động ngày càng nhận được sự quan tâm và đầu tư của doanh nghiệp. Có 204 doanh nghiệp trên tổng số 594 doanh nghiệp xác nhận lên kế hoạch tăng cường đầu tư vào kiểm thử tự động và đảm bảo chất lượng phần mềm trong 12 tháng tới. Chỉ có 24.6% nhận định là không lên kế hoạch trong lĩnh vực này.

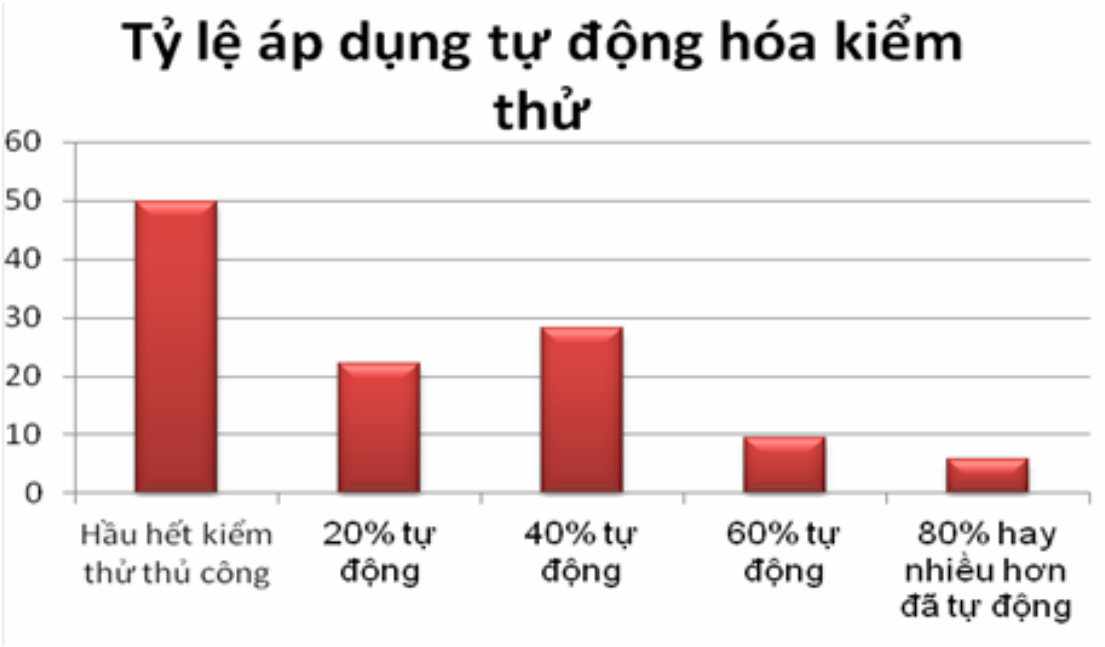
Khi được hỏi liệu họ có tin rằng hầu hết các phần mềm có liên quan đến qui trình SAP có thể được kiểm thử hiệu quả với kiểm thử tự động? Thì 68.1% tin tưởng rằng các phần mềm kiểm thử chức năng có thể cung cấp kiểm thử tự động mức độ cao (bao quát trên 50% các qui trình nghiệp vụ quan trọng). Điều này thể hiện sự trưởng thành đáng kể của thị trường cho kiểm thử tự động phần mềm, và giúp giải thích việc áp dụng nhanh chóng của công nghệ này hiện nay.

Kiểm thử tự động mang đến lợi ích trong nhiều lĩnh vực cho hầu hết các công ty. Hơn 4/5 các doanh nghiệp sử dụng kiểm thử tự động xác định các lợi ích kinh doanh của kiểm thử tự động trong rất nhiều lĩnh vực (86%), với hầu hết những người được hỏi xác định được từ 3 đến 6 lĩnh vực.Trong đó nhóm 5 các lĩnh vực đó là:

- Tiết kiệm đội ngũ nhân lực và thời gian
- Xác định sớm các lỗi trước khi ảnh hưởng đến người sử dụng nghiệp vụ
- Chất lượng cao hơn trong qui trình nghiệp vụ
- Bắt được nhiều lỗi với độ chính xác cao hơn
- Phát triển sang tạo và các tính năng mới nhanh hơn

Kết quả của bản điều tra đã chỉ rõ ra các công ty toàn cầu đã nhận thức cao hơn về các lợi ích quan trọng của kiểm thử tự động, dẫn đến cơ hội quan trọng trong việc tăng cường áp dụng kiểm thử tự động: Nhìn chung thâm nhập trị thường của kiểm thử tự động vẫn khá thấp với 49.8% những công ty được hỏi báo cáo rằng hầu hết các kiểm thử SAP của họ vẫn được thực hiện thủ công, và thêm 22.1% báo cáo dưới 20% kiểm thử chức năng của họ được tự động hóa. Khoảng 28.2 % doanh nghiệp đã đạt được mức độ tương đối cao hay rất cao của kiểm thử tự động – trên 40% tự động hóa. Ở những doanh nghiệp đã tự động hóa ở mức độ cao thì 5.8% báo

cáo rằng hơn 80% hoặc hầu như toàn bộ kiểm thử chức năng của họ đã được tự động hóa, như hình minh họa ở Hình 4. (Chiều dọc: Tỉ lệ % số lượng công ty được hỏi. Chiều ngang: % tương ứng với kiểu thực hiện kiểm thử tự động hoặc thủ công)



Hình 4: Tình hình áp dụng tự động hóa kiểm thử

Một trong những điểm thú vị của bản điều tra này đó là chỉs ra tiềm năng áp dụng kiểm thử tự động ở mức cao trong các doanh nghiệp vận hành. Hầu hết các chuyên gia trong nghề - hơn 2/3 – đã nhận ra rằng phần mềm kiểm thử tự động có sẵn ngày nay có thể cung cấp mức độ tự động hóa rất cao. Nhưng thực tế hầu hết các doanh nghiệp hiện nay vẫn dựa chủ yếu trên tiếp cận thủ công. Vì vậy kiểm thử tự động hứa hẹn là một thị trường rất tiềm năng. Dẫn đến nhu cầu rất lớn trong tuyển dụng và đào tạo kỹ sư kiểm thử phần mềm tự động. [8]

Tại Việt Nam

Theo báo cáo dịch vụ phần mềm toàn cầu hàng năm từ Gartner, Việt Nam đứng trong tốp 30 của các nước gia công phần mềm hàng đầu thế giới cho gia công phần mềm và đứng trong tốp 10 của khu vực Châu Á – Thái Bình Dương. Kiểm thử phần mềm là một nghành công nghiệp mới và nắm giữ rất nhiều tiềm năng cho Việt Nam, đặc biệt là trong lĩnh vực gia công phần mềm, nơi mà kiểm thử phần mềm

đang thu hút được sự quan tâm của giới trẻ [9]. Tuy nhiên nguồn nhân lực về kiểm thử phần mềm và đặc biệt là kiểm thử tự động ở Việt Nam đang không đáp ứng được nhu cầu của các nhà tuyển dụng.

Các công ty phần mềm lớn như FPT hay Logigear đã tự xây dựng cho mình các trung tâm đào tạo nhân lực kiểm thử phần mềm và kiểm thử tự động. Rất nhiều dự án lớn của những công ty này đã được áp dụng tự động hóa với tỉ lệ thực hiện tự động cao (>70%).

1.5 Tình hình nghiên cứu kiểm thử tự động

Kiểm thử tự động có thể được áp dụng ở qui mô nhỏ cho một phần của hoạt động kiểm thử, như là kiểm tra hai tệp tin xem có dữ liệu tương đương như nhau không, đến qui mô lớn là là thực hiện tự động toàn bộ hệ thống kiểm thử, từ cài đặt môi trường, thực thi các bài kiểm thử cho đến việc báo cáo kết quả.

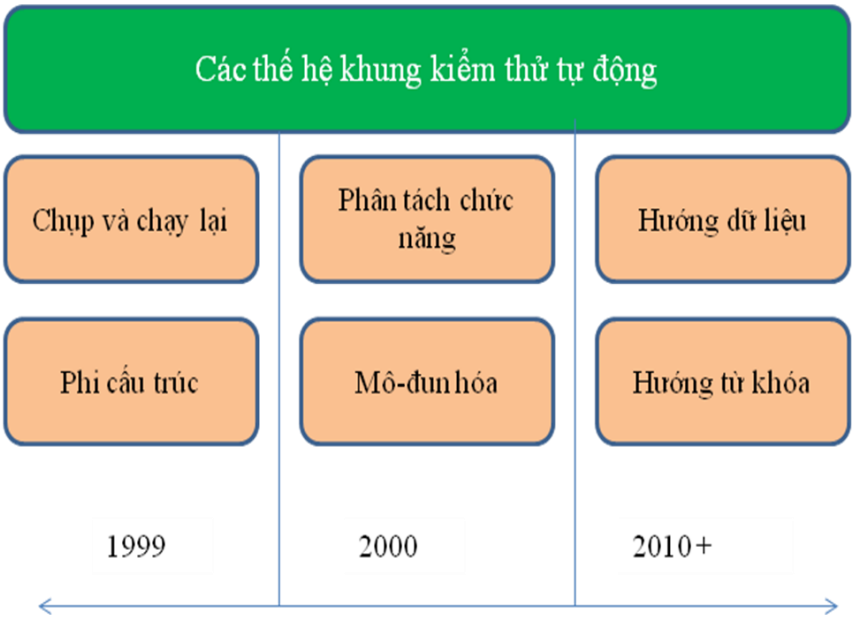
Tự động hóa ở qui mô nhỏ chỉ nhằm hỗ trợ cho kiểm thử thủ công. Các công cụ kiểm thử tự động được sử dụng trong các lĩnh vực mà máy tính làm việc tốt hơn con người và việc cài đặt và sử dụng các công cụ đó là dễ dàng. Ví dụ như việc kiểm tra rằng trình cài đặt đã sao chép toàn bộ các tệp tin (File) cần thiết đến đúng thư mục, kiểm tra hai tệp tin có dữ liệu như nhau không thì tốn nhiều thời gian và công sức của kiểm thử viên nếu thực hiện thủ công, nhưng lại rất dễ để tự động hóa. Cách tiếp cận kiểm thử tự động này rất được ủng hộ bởi Bach bởi tính dễ xây dựng, không đắt đỏ và có thể sửa lỗi dễ dàng [4] . Ông đề ra một ý tưởng về người được gọi là Toolsmiths, tạm dịch là Thợ xây dựng công cụ. Những người thợ này sẽ làm việc với các kỹ sư kiểm thử và cung cấp cho họ công cụ và các tiện ích dựa trên nhu cầu của kỹ sư kiểm thử. Toolsmiths cần có kỹ năng lập trình tốt, có các kỹ năng kiểm thử đầy đủ và có kiến thức sâu rộng về các công cụ và phương pháp tự động. Có rất nhiều công cụ tốt sẵn có cho nhu cầu tự động hóa và rất nhiều trong số chúng là miễn phí.

Thực hiện tự động hóa kiểm thử trên qui mô nhỏ có lẽ là một chiến lược rất tốt khi bắt đầu tiếp cận với tự động, bởi nó không yêu cầu đầu tư lớn và cần trợ giúp nhanh. Đặc biệt là nếu thực hiện tự động với qui mô lớn hơn, với khung kiểm thử tự

động (Automation testing framework) khiến cho rủi ro cũng lớn hơn thì cách tốt nhất là bắt đầu với qui mô nhỏ, đúc kết thêm nhiều kinh nghiệm và sau đó đầu tư vào khung lớn hơn.

Khi mà kiểm thử tự động được thực hiện đến mức cao, như là có thể thực hiện tự động "Nhấn vào một nút (Button)", thì hệ thống tự động có thể tự thực hiện qua đêm mà không cần có giám sát và trả về kết quả kiểm thử vào sáng hôm sau. Rõ ràng rằng để thực hiện tự động như thế này yêu cầu phải có một số hệ thống giúp tạo ra, thực thi và bảo trì các bài kiểm thử (bài kiểm thử) một cách dễ dàng. Hệ thống cũng cần phải cung cấp một số chức năng cốt lõi (như là giám sát và báo cáo), và cho phép tự mở rộng để có thể tạo ra những kiểu bài kiểm thử mới. Những hệ thống kiểu như thế này khớp với định nghĩa về "Framework" và áp dụng cho kiểm thử tự động nên có thể gọi nó một cách phù hợp là "Framework kiểm thử tự động".

Khung kiểm thử tự động được xuất hiện từ rất lâu, và được tổng quát hóa như ở Hình 5.[15]



Hình 5: Các thế hệ khung kiểm thử tự động

1. Thế hệ đầu tiên của khung là phi cấu trúc, có dữ liệu kiểm thử được nhúng vào trong đoạn mã (Script) và thường có một đoạn mã cho mỗi bài kiểm thử. Các đoạn mã chủ yếu được sinh ra bởi sử dụng công cụ "chụp và chạy lại" (Capture and

replay) nhưng cũng có thể được viết thủ công. Kiểu đoạn mã như thế này thường không có khả năng bảo trì và khi mà hệ thống được kiểm thử có thay đổi thì đoạn mã liên quan sẽ phải thực hiện lại công việc chụp và chạy lại.

2. Thế hệ khung thứ hai có mã nguồn được thiết kế tốt, mô-đun hóa, mạnh, có tài liệu và do đó có khả năng bảo trì. Đoạn mã không chỉ thực thi kiểm thử mà ví dụ cũng có thể cài đặt, làm sạch, phát hiện lỗi và phục hồi. Dữ liệu kiểm thử vẫn được nhúng trực tiếp vào trong đoạn mã nên vẫn có một mã nguồn điều khiển (Driver scripts) cho một bài kiểm thử. Mã nguồn hầu hết được viết bằng tay và khả năng bảo trì thì yêu cầu cần có kỹ năng lập trình mà có thể kiểm thử viên không có.

3. Thế hệ khung thứ ba có tất cả các đặc tính tốt được tìm thấy ở thế hệ thứ hai, ngoài ra thì việc dữ liệu kiểm thử được tách ra khỏi mã nguồn cũng có thêm nhiều lợi ích đáng kể. Lợi ích đầu tiên là một đoạn mã có thể dùng cho rất nhiều bài kiểm thử bằng cách chỉ cần điều chỉnh dữ liệu kiểm thử và thêm một số bài kiểm thử một cách bình thường. Lợi ích thứ hai đó là thực hiện thiết kế kiểm thử và việc thực hiện viết mã nguồn để thực thi kiểm thử là hai công việc riêng rẽ, việc đầu tiên có thể thực hiện bởi người có kiến thức về lĩnh vực kiểm thử, biết cách xây dựng những bài kiểm thử có chất lượng tốt và việc thứ hai được thực hiện bởi người có kĩ năng lập trình mà có thể không có kĩ năng thiết kế bài kiểm thử. Khái niệm này được gọi là kiểm thử hướng dữ liệu (Data-driven testing). Kiểm thử hướng từ khóa (Keyword-driven testing) đó là thêm các từ khóa điều khiển việc thực thi kiểm thử vào dữ liệu kiểm thử. [4]

Việc áp dụng khung thế hệ thứ ba trong kiểm thử tự động chính là mục tiêu của khóa luận này. Chi tiết về kiểm thử hướng dữ liệu và kiểm thử hướng từ khóa sẽ được giới thiệu cụ thể hơn ở chương 2.

1.6 Tình hình ứng dụng kiểm thử tự động

Kiểm thử tự động ngày càng nhận được sự quan tâm và đầu tư nhiều hơn của các công ty phần mềm nhằm tăng năng suất lao động và giảm chi phí. Cùng với đó là sự ra đời của rất nhiều các công cụ hỗ trợ tự động cho từng giai đoạn của kiểm thử phần mềm như quản lý tài liệu kiểm thử, thực thi kiểm thử, tự động sinh các bài

kiểm thử tự động. Bảng 1 liệt kê ra một số công cụ hỗ trợ tự động hóa trong kiểm thử phần mềm. [10]

Bảng 1. Phân loại các công cụ kiểm thử phần mềm tự động

#	Loại công cụ	Miêu tả chức năng
1	Quản lý thông tin kiểm thử (Test information management)	Công cụ và các giải pháp hỗ trợ các kỹ thuật kiểm thử và đảm bảo chất lượng, giúp tạo, cập nhật và bảo trì các thông tin kiểm thử đa dạng, bao gồm bài kiểm thử, kịch bản, dữ liệu, kết quả kiểm thử và các vấn đề được phát hiện.
2	Điều khiển và thực thi kiểm thử (Test execution and control)	Công cụ và giải pháp hỗ trợ thiết lập và thực hiện kiểm thử, tập hợp và đánh giá kết quả kiểm thử.
3	Tạo kiểm thử tự động (Test generation)	Giải pháp và công cụ tạo ra các chương trình kiểm thử một cách tự động.
4	Phân tích độ phủ của kiểm thử (Test coverage analysis)	Công cụ và giải pháp thực hiện phân tích mật độ phủ của bài kiểm thử dự trên các tiêu chí được lựa chọn.
5	Đo hiệu năng (Performance testing and measurement)	Công cụ và giải pháp hỗ trợ việc đo và kiểm tra hiệu năng của hệ thống.
6	Giả lập phần mềm (Software simulators)	Chương trình được phát triển để giả lập chức năng hoặc hành động của hệ thống bên ngoài hoặc các thành phần/hệ thống con cần thiết cho việc kiểm thử.
7	Kiểm thử hồi qui (Regression testing)	Công cụ hỗ trợ kiểm thử hồi qui và các hoạt động “Ghi” và “Chụp và chạy lại” (recording, capturing and replaying)

1. Quản lý thông tin kiểm thử (Test information management)

Có 3 loại công cụ quản lý thông tin kiểm thử:

▪ Công cụ quản lý thông tin: hỗ trợ các kiểm thử viên trong việc tạo mới, cập nhật và quản lý tất cả các loại thông tin kiểm thử, như là yêu cầu kiểm thử, các bài kiểm thử, dữ liệu, kết quả kiểm thử.

▪ Công cụ quản lý bộ kiểm thử: Cho phép kiểm thử viên tạo, cập nhật, quản lý mã nguồn cho việc thực thi kiểm thử.

▪ Công cụ quản lý vấn đề: Hỗ trợ các kiểm thử viên ghi chép (bookkeeping) và quản lý những vấn đề chưa được phát hiện trong qui trình kiểm thử.

2. Điều khiển và thực thi kiểm thử (Test execution and control)

Những công cụ này cho phép điều khiển và thực thi các chương trình kiểm thử và mã kiểm thử một cách tự động. Chúng có khả năng cài đặt mã nguồn đã được chọn trước và dữ liệu kiểm thử, gọi là thực thi chúng sau đó đánh giá kết quả kiểm thử dựa trên kết quả mong muốn. WinRunner của Mercury Interactive's là một ví dụ của công cụ này. Một số ví dụ khác cũng được thể hiện ở Bảng 2.

Bảng 2 Công cụ kiểm thử tự động và nhà cung cấp

#	Loại công cụ	Nhà cung cấp	Công cụ
1	Công cụ quản lý các vấn đề (Problem management tools)	Rational Inc. Microsoft Corp. Imbus AG	ClearQust, ClearDDTS PVCS Tracker Imbus Fehlerdatenbank
2	Công cụ quản lý thông tin kiểm thử (Test information management tools)	Rautional Inc. Mercury Interactive	TestManager TestDirectory
3	Công cụ quản lý bộ bài kiểm thử	Evalid Rational Inc. SUN	TestSuiter TestFactory

	(Test suite management tools)		JavaTest, JavaHarness
4	Công cụ kiểm thử hộp trắng (White-box test tools)	McCabe & Associates IBM	McCabe IQ2 IBM COBOL Unit Kiểm thử viên IBM ATC Coverage Assistant Source Audit Assistant Distillation Assistant Unit Test Assistant
5	Công cụ thực hiện kiểm thử (Test execution tools)	OC Systems Softbridge AutoKiểm thử viên Rational Inc. SQA Mercury Interactive Sterling Software Compuware Seque Software RSW Software Inc. Cyrano Gmbh	Aprob ATF/TestWright AutoKiểm thử viên Visual Test Robot WinRunner Vision TestPro QARun SilkTest e-Test Cyrano Robot
6	Công cụ phân tích mật độ phủ của code (Code coverage analysis tools)	Case Consult Corp. OC Systems IPL Software Product Group ATTOL Testware SA Compuware NuMega Software Research Rational Inc SUN ParaSoft Software Automation Inc.	Analyzer, Analyzer Java Aprob Cantata/Cantata++ Coverage TruCoverage TestWorks Coverage PureCoverage JavaScope TCA Panorama

7	Công cụ kiểm thử hiệu năng (Load test and performance tools)	Rational Inc. InterNetwork AG Compuware Mercury Interactive RSW Software Inc. SUN Seque Software Client/Server Solutions, Inc.	Rational Suite PerformanceStudio sma@rtTest QA-Load LoadRunner e-Load JavaLoad SilkPerformer Benchmark Factory
8	Công cụ kiểm thử hồi qui (Regression testing tools)	IBM	Regression Testing Tool(ARTT) Distillation Assistant
9	Công cụ ghi/chơi lại (GUI record/replay)	Software Research Mercury Interactive Astra Auto	eValid Xrunner Astra QuickTest AutoTester, AutoTester One

3. Tự động tạo ra các bài kiểm thử

Những công cụ này liên quan đến các chương trình cho phép sinh tự động các bài kiểm thử (Test generation) cho các phần mềm cần kiểm thử. Có hai lớp của công cụ này:

- Công cụ sinh tự động kiểm thử hộp trắng: Tạo ra các bài kiểm tra hộp trắng dựa trên mã nguồn và cấu trúc của chương trình cần kiểm thử, như là kĩ thuật kiểm thử độ bao phủ mã nguồn (Path testing technique).

- Công cụ sinh tự động kiểm thử hộp đen: Tạo ra các bài kiểm tra hộp đen dựa trên yêu cầu của chương trình bằng cách sử dụng các phương pháp kiểm tra hộp đen, chẳng hạn như thử nghiệm ngẫu nhiên và các phương pháp phân tích giá trị biên.

4. Phân tích độ phủ của kiểm thử (Test coverage analysis)

Những công cụ này có thể phân tích và giám sát độ phủ của kiểm thử cho qui trình kiểm thử dựa trên tiêu chí lựa chọn bài kiểm thử. Ví dụ như trong kiểm thử hộp trắng, thường lựa chọn tiêu chí bao phủ nhánh và bao phủ câu lênh. Trong kiểm thử hộp đen, có thể lựa chọn tiêu chí kiểm thử giá trị biên. Một số công cụ phân tích độ phủ của kiểm thử được giới thiệu ở Bảng 2.

5. Đo hiệu năng (Performance testing and measurement)

Những công cụ này hỗ trợ thực hiện đo đạc và kiểm thử hiệu năng, bao gồm việc đánh giá, đo lường và phân tích hiệu năng. Những công cụ này được phát triển dựa trên những mô hình và số liệu hiệu năng đã được định nghĩa từ trước. Công cụ LoadRunner là một trong những ví dụ điển hình. Nó có thể sử dụng thể thực hiện kiểm thử tải (Load testing) cho một số chương trình máy chủ để kiểm tra suất tải của phần mềm ở mức hệ thống. Một số công cụ khác được giới thiệu ở Bảng 2 Công cụ kiểm thử tự động và nhà cung cấp.

6. Giả lập phần mềm (Software simulators)

Đây là những chương trình được phát triển để giả lập chức năng và một số thực thể bên ngoài như phần mềm hoặc phần cứng, các thành phần, hệ thống con. Các chương trình giả lập là cần thiết và hữu ích cho việc tích hợp chương trình và kiểm thử hệ thống. Opnet, phát triển bởi Mil 3Inc là một ví dụ cho công cụ này. Opnet là một chương trình mô phỏng mạng lưới mà rất hữu ích trong việc giả lập hành vi của hệ thống mạng.

7. Kiểm thử hồi qui (Regression testing)

Những công cụ hỗ trợ kiểm thử hồi qui bao gồm những chức năng như sau:

- Phân tích thay đổi của phần mềm: Những công cụ này xác định rất nhiều loại thay đổi của phần mềm và phát hiện ra những ảnh hưởng của các thay đổi đó. Những thay đổi này bao gồm thêm mới, cập nhật, xóa các thành phần và yếu tố giữa các phiên bản của phần mềm.

- Lựa chọn kiểm thử: Trong kiểm thử hồi qui, kĩ sư kiểm thử cần thực hiện lựa chọn những bài kiểm thử dựa trên các thông tin thay đổi của phần mềm.

- Phân tích thay đổi kiểm thử: Chức năng này liên quan đến việc xác định các thay đổi cần thiết của việc thiết kế kiểm thử dựa trên những thay đổi của phần mềm. Nhiệm vụ chính của việc phân tích thay đổi của thiết kế kiểm thử là để tìm ra những bài kiểm thử đã bị lỗi thời, những ca có thể tái sử dụng hay phải thêm mới hoặc cập nhật bài kiểm thử.

- Công cụ chụp và chạy lại (Test recorder and replayer): Đây là công cụ thực hiện ghi lại những bài kiểm thử đã được thực hiện, sau đó chơi lại để thực hiện kiểm thử lại. WinRunner của Mercury Interactive’s là một công cụ như thế.

CHƯƠNG 2. GIẢI PHÁP KIỂM THỬ TỰ ĐỘNG HƯỚNG DỮ LIỆU VÀ TỪ KHÓA

Chương 1 đã giới thiệu những khái niệm chung về kiểm thử tự động phần mềm, trong đó nhấn mạnh luân văn tập trung vào những khung kiểm thử tự động trên qui mô lớn. Khung kiểm thử 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 như là chức năng, nguồn dữ liệu kiểm thử chi tiết các đối tượng và mô-đun có thể tái sử dụng khác nhau [12] . Với mục tiêu áp dụng khung kiểm thử tự động hướng dữ liệu và từ khóa, trong Chương 2 sẽ trình bày về yêu cầu của khung kiểm thử tự động, nghiên cứu giải pháp kiểm thử hướng dữ liệu (Data –driven testing) và hướng từ khóa (Keyword-driven testing).

2.1. Yêu cầu chức năng của khung tự động hóa kiểm thử

Những yêu cầu mức cao của khung kiểm thử tự động được liệt kê ngắn gọn trong Bảng 3. [4]

Bảng 3: Yêu cầu mức cao cho khung kiểm thử tự động

Tự động thực thi kiểm thử	Là yêu cầu số một của kiểm thử tự động, nhưng nếu chỉ thực thi kiểm thử thôi thì chưa đủ, khung kiểm thử tự động còn phải có khả năng phân tích kết quả kiểm thử, kiểm soát lỗi và xuất ra báo cáo.
Dễ sử dụng	Cho phép người dùng có thể thiết kế và chỉnh sửa kịch bản kiểm thử, sau đó chạy và giám sát trạng thái của quá trình kiểm thử mà không cần phải có kỹ năng lập trình.
Khả năng bảo trì	Khả năng bảo trì dữ liệu kiểm thử và mã nguồn của khung kiểm thử tự động phải có thể sửa đổi thật nhanh và dễ dàng khi hệ thống đang kiểm thử có thay đổi. Có khả năng tạo thêm nhiều tính năng mới cho khung kiểm thử tự động

Những yêu cầu như ở Bảng 3 có thể phân chia ra nhiều yêu cầu chi tiết hơn như sau:

2.1.1. Thực thi kiểm thử mà không cần phải giám sát

Khung kiểm thử (Framework) phải có thể bắt đầu thực thi kiểm thử sau khi được nhấn vào một nút (Button), điều này có nghĩa rằng khung phải có thể cài đặt môi trường kiêm thử cũng như phải kiểm tra được tất cả các điều kiện đó đã được thỏa mãn.

2.1.2.Bắt đầu và dừng thực thi kiểm thử

Khung kiểm thử tự động phải có khả năng bắt đầu thực thi kiểm thử một cách thủ công. Tốt hơn nữa nếu có thể tự động thực thi kiểm thử tại một thời gian xác định hay sau một sự kiện nào đó (ví dụ: Có phiên bản mới của hệ thống cần kiểm thử). Cách dễ nhất để bắt đầu thực thi kiểm thử đó là từ câu lệnh được viết thủ công sử dụng các tính năng của hệ điều hành cho việc tạo lịch. Bắt đầu kiểm thử sau khi có một sự kiện nào đó xuất hiện có thể được thực hiện tương tự như việc sử dụng một công cụ bên ngoài (External tools).

2.1.3.Kiểm soát lỗi

Một phần của việc thực thi kiểm thử mà không cần giám sát đó là phục hồi từ lỗi mà gây ra bởi hệ thống được kiểm thử hoặc do môi trường kiểm thử hoạt động không như mong muốn. Khung kiểm thử tự động phải bắt được lỗi đó, ghi log lại rồi tiếp tục thực hiện kiểm thử những ca tiếp theo mà không cần phải có sự can thiệp thủ công. Nên tránh kiểm soát tất cả những lỗi có thể xảy ra nhưng lại rất hiếm khi xảy ra để không tốn nhiều công sức, thời gian, chi phí. Những trường hợp kiểm thử ít khi xảy ra hay chỉ xảy ra một lần thì kiểm thử thủ công sẽ là phương án phù hợp hơn so với kiểm thử tự động.

2.1.4.Thẩm định kết quả kiểm thử

Một trong những phần quan trọng của thực thi kiểm thử đó là thẩm định kết quả kiểm thử. Fewster và Graham đã định nghĩa thẩm định như là một hay nhiều phép so sánh giữa kết quả thực tế và kết quả mong muốn mà đã được định nghĩa từ

trước [4]. Họ cũng giải thích rất nhiều kỹ thuật so sánh mà không nằm trong phạm vi của luận văn này.

2.1.5.Gán trạng thái kiểm thử

Sau khi kiểm thử được thực thi và kết quả của nó được xác minh thì cần gán trạng thái cho kết quả cuối cùng. Nếu như trong quá trình thực thi kiểm thử, không xuất hiện lỗi gì và tất cả các phép so sánh giữa kết quả thực tế và kết quả mong muốn hoàn toàn khớp với nhau thì bài kiểm thử (Test case) có trạng thái là đạt yêu cầu (Pass), nghĩa là nội dung cần kiểm thử đã hoạt động đúng như yêu cầu. Nếu ở trong các trường hợp khác thì trạng thái kiểm thử là thất bại (Failed), nghĩa là nội dung cần kiểm thử đã không thỏa mãn yêu cầu của bài kiểm thử.

Bên cạnh trạng thái, bài kiểm thử cần cũng được gán thêm một nội dung thông báo nào đó ngắn gọn nhưng thể hiện được đầy đủ ý nghĩa. Thông báo này thường không quan trọng đối với các bài kiểm thử đã đạt yêu cầu nhưng lại cực kỳ quan trọng với bài kiểm thử bị thất bại, đó là ghi ra chi tiết về nguyên nhân của vấn đề nảy sinh khiến bài kiểm thử thất bại (ví dụ: Calculator Failed: mong muốn 2, thực tế 3). Fewster và Graham đề xuất rằng nên có nhiều trạng thái hơn là chỉ có Pass hoặc Failed.

2.1.6. Xử lý các lỗi như mong muốn

Một trong những phần khó chịu của việc phân tích các lỗi đó là phải đi qua những bài kiểm thử mà biết kết quả trả về là thất bại (Failed) và kiểm tra xem liệu chúng đã bị thất bại từ trước hay do xuất hiện lỗi mới.

Để có thể phân biệt được kết quả thất bại như mong mốn (Expected failures) xuất phát từ một lỗi mới xuất hiện hay không thì khung làm việc (Framework) phải biết được kết quả mong muốn khi bài kiểm thử thất bại. Điều này có nghĩa là khung làm việc phải lưu trữ cả kết quả mong muốn khi đạt (Pass) và mong muốn khi thất bại (Failed) của bài kiểm thử. Khi mà bài kiểm thử thất bại như mong muốn, thì cần so sánh kết quả này với kết quả thực tế của bài kiểm thử. Nếu hai giá trị này giống nhau thì bài kiểm thử trả về giá trị lỗi như mong muốn, còn nếu sai thì có thể do một lỗi nào đó mới xuất hiện.

2.1.7.Ghi lại thông tin chi tiết

Khung kiểm thử tự động thường đưa đủ thông tin cho kỹ sư kiểm thử và người phát triển phần mềm để họ có thể điều tra bài kiểm thử mà đã bị thất bại. Chỉ có đưa trạng thái của bài kiểm thử cùng với một số thông điệp trạng thái thôi là chưa đủ. Khung cần ghi lại thông tin (Log) mà nó thực sự làm ở bên trong để giúp gỡ lỗi (Debug) vấn đề dễ dàng hơn. Như ở bảng dưới đây gợi ý một số mức thông tin log mà hệ thống cần ghi lại.

Bảng 4: Các mức ghi lại thông tin chi tiết

Fail/Không đạt yêu cầu	Sử dụng để log trạng thái của bài kiểm thử không đạt yêu cầu
Pass/Đạt yêu cầu	Sử dụng để log trạng thái của bài kiểm thử đạt yêu cầu
Fatal/Nghiêm trọng	Sử dụng khi gặp những thất bại chưa lường trước được mà làm ảnh hưởng đến việc thực thi các kiểm thử. Những vấn đề này thường gặp trong môi trường kiểm thử (ví dụ như ổ đĩa đầy hay mạng lỗi)
Error/Lỗi	Sử dụng khi một lỗi không mong muốn làm cản trở việc thực thi bài kiểm thử xuất hiện, ví dụ như trường hợp không tồn tại nút (Button) để bấm.
Failure/Thất bại	Sử dụng khi kết quả thực thi kiểm thử không khớp với kết quả mong muốn.
Warning	Sử dụng khi có lỗi mà có thể lường trước được, không ảnh hưởng đến việc thực thi kiểm thử, như trường hợp xóa một số tệp tin không thành công.
Info	Sử dụng để log các bài kiểm thử bình thường như là bắt đầu kiểm thử và các xác minh đã đạt yêu cầu.
Gỡ lỗi (debug)	Đưa thông tin chi tiết về việc framework hay bài kiểm thử cụ thể đang làm gì.
Trace	Tương tự như gỡ lỗi (Debug) nhưng có thêm nhiều thông tin hơn.

Tình trạng khó xử đối với bất kỳ loại log nào đó là log bao nhiêu? log quá nhiều thông tin sẽ có thể dẫn đến khó khăn khi tìm dữ liệu cần tìm, hoặc dẫn đến vấn đề lưu trữ. Ngược lại nếu log quá ít thì có thể khiến cho những thông tin đó không có tác dụng trong việc tìm hiểu nguyên nhân của vấn đề. Không có giải pháp hoàn hảo cho vấn đề này nhưng có một số phương pháp – ví dụ sử dụng các mức log khác nhau có thể khiến cho vấn đền này nhỏ hơn.

2.1.8.Báo cáo tự động

Từ những thông tin hệ thống ghi ra (log) đã có tất cả các thông tin thực thi kiểm thử nhưng, đặc biệt là trong trường hợp thông tin này quá dài, kết quả kiểm thử chỉ hiện lên trong nháy mắt là rất khó để tìm thấy. Vì vậy cần có một báo cáo ngắn gọn, cung cấp đầy đủ thông tin thống kê về chất lượng của hệ thống được kiểm thử và chúng không chỉ quan tọng cho kiểm thử viên mà còn với người phát triển và toàn bộ những ai liên quan trong dự án.

Báo cáo không cần quá dài hoặc quá chi tiết, có một tổng hợp ngắn và một danh sách những bài kiểm thử đã thực hiện với trạng thái tương ứng của bài kiểm thử là đủ. Danh sách những mục cần có trong báo cáo được thể hiện như dưới đây:

- Tên hoặc phiên bản của hệ thống được kiểm thử
- Định danh của các bộ kiểm thử đã được thực thi
- Tổng số những bài kiểm thử đã được thực thi
- Danh sách các lỗi được tìm thấy.
- Một danh sách tất cả các bài kiểm thử đã thực thi cùng với trạng thái của nó (tùy chọn)

Khi đã có khung kiểm thử tự động thỏa mãn các điều kiện như trên, kiểm thử viên cần thực hiện thiết kế các bài kiểm thử cho ứng dụng của mình. Như đã giới thiệu ở chương 1, có rất nhiều phương pháp tiếp cận khung kiểm thử tự động và mục tiêu của luận văn đó là áp dụng hai phương pháp kiểm thử hướng dữ liệu và kiểm thử hướng từ khóa. Phần dưới đây sẽ đi sâu về giải pháp này.

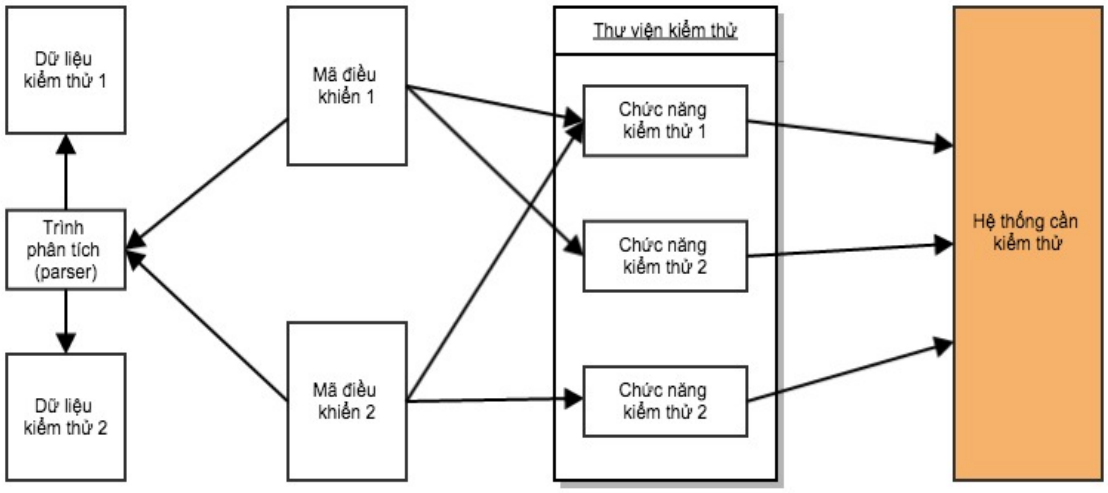
2.2. Kiểm thử hướng dữ liệu (Data-driven testing)

2.2.1.Giới thiệu

Những mã nguồn kiểm thử đơn giản thường có dữ liệu được nhúng sẵn ở trong, dẫn đến vấn đề là khi cần thay đổi dữ liệu kiểm thử, mã nguồn cũng cần thay đổi theo.

Điều này có thể không phải là vấn đề lớn khi người cần thay đổi mã nguồn chính là người mà đã viết nên mã nguồn này ngay từ ban đầu, nhưng sẽ là vấn đề lớn đối với những người khác và không có kinh nghiệm lập trình. Việc nhúng dữ liệu kiểm thử trong mã nguồn kiểm thử cũng có thêm một vấn đề khác nữa đó là khi muốn tạo thêm bài kiểm thử với các bước tương tự nhưng có sự thay đổi về dữ liệu kiểm thử thì luôn luôn yêu cầu phải lập trình. Nhiệm vụ này có thể rất dễ dàng – sao chép dữ liệu ban đầu và sửa phần dữ liệu kiểm thử – nhưng yêu cầu ít nhất phải có kiến thức lập trình. Việc tái sử dụng như thế này sẽ dẫn đến việc nếu phần mềm bị thay đổi thì toàn bộ mã nguồn phải viết lại. [4]

Bởi những vấn đề này mà việc nhúng dữ liệu kiểm thử vào trong mã nguồn kiểm thử không phải là giải pháp phù hợp khi xây dựng một khung kiểm thử tự động. Cách tiếp cận tốt hơn đó là đọc dữ liệu kiểm thử từ một nguồn dữ liệu bên ngoài và thực hiện các bài kiểm thử dựa trên nó. Cách tiếp cận này được gọi là kiểm thử hướng dữ liệu.và được minh họa ở Hình 6.



Hình 6: Kiểm thử hướng dữ liệu

Nguồn dữ liệu kiểm thử bên ngoài cần phải dễ dàng thay đổi, cập nhật bởi kiểm thử viên mà không cần có kĩ năng lập trình. Dữ liệu kiểm thử thường được tạo dưới dạng bảng như ở Hình 7.

	A	B	C	D	E
1	**Test case**	**Number 1**	**Operator**	**Number 2**	**Expected**
2	Add 01	1	+	2	3
3	Add 02	1	+	-2	-1
4	Sub 01	1	-	2	-1
5	Sub 02	1	-	-2	3
6	Mul 01	1	*	2	2
7	Mul 02	1	*	-2	-2
8	Div 01	2	/	1	2
9	Div 02	2	/	-2	-1

Hình 7: Dữ liệu kiểu kiểm thử hướng dữ liệu

2.2.2.Sửa đổi và lưu trữ dữ liệu

Dữ liệu kiểm thử được lưu trữ ở dưới dạng bảng và có thể sử dụng các chương trình bảng tính như excel để sửa đổi dữ liệu một cách dễ dàng. Các tệp tin dạng bảng có thể được lưu trữ dưới dạng CSV (comma-separated values), TSV (tab-separated values), hay dưới định dạng tự nhiên của chương trình bảng tính. TSV và CSV rất tiện dụng vì chúng rất dễ dàng để phân tích nhưng tiếc là hầu hết các chương trình bảng tính đều có vẻ thay đổi dữ liệu khi mở những tệp tin này. Ví dụ như nếu lưu trữ số điện thoại +358912345 và số phiên bản 1.5.1 vào một tập tin CSV và mở tệp tin này bằng Excel thì Excel đã "tự động điều chỉnh" thành 358912345 và 1.5.2001 tương ứng. Giải pháp dễ dàng nhất cho vấn đề này đó là lưu trữ dữ liệu dưới định dạng gốc của chương trình và chỉ xuất dữ liệu sang dạng CSV, TSV. Một khả năng khác đó là xử lý định dạng gốc trực tiếp nhưng nó đòi hỏi một số nỗ lực ban đầu.

Bảng dạng HTML cung cấp một định dang dễ phân tích để trình bày dữ liệu kiểm thử. Chỉnh sửa HTML với một trình soạn thảo đồ họa khá thuận tiện, gần như sử dụng một chương trình bảng tính. Lưu trữ dữ liệu kiểm thử vào một tệp tin phẳng

có nhiều vấn đề về khả năng mở rộng tệp tin. Ví dụ dữ liệu kiểm thử được tạo ra, sửa đổi và được sử dụng trong nhiều môi trường kiểm thử, sẽ dẫn đến việc khó để có thể có cùng phiên bản của dữ liệu kiểm thử ở khắp mọi nơi. Khi đó việc cấu hình quản lý và có hướng dẫn rõ ràng sẽ giúp ích nhưng vẫn là chưa đủ. Giải pháp thực tế vấn đề khả năng mở rộng đó là lưu trữ dữ liệu vào trong một cơ sở dữ liệu trung tâm. Bằng cách đó, dữ liệu kiểm thử sẽ được sửa đổi ví dụ như cho giao diện web và mã điều khiển có thể truy vấn dữ liệu trực tiếp từ cơ sở dữ liệu. Việc xây dựng một hệ thống như vậy đã là một dự án lớn và chỉ nên thực hiện khi gặp vấn đề thực sự về khả năng mở rộng.

2.2.3.Xử lý dữ liệu kiểm thử

Việc cài đặt một đoạn mã để thực hiện phân tích dữ liệu hướng kiểm thử có thể dễ dàng đến mức ngạc nhiên với ngôn ngữ kịch bản hiện đại. Ví dụ dữ liệu trong Hình 7 có thể được xuất ra một tệp TSV và phân tích với bốn dòng lệnh Python (Hình 8):

```
data = open ('testdata.tsv).read()
line = data.splitlines () [1:]          #[1:] bao gồm cả tên dòng
for line in lines :
    testId, number1, operator, number2, expected = line.split ('\t')
    #thực hiện kiểm thử
```

Hình 8: Đọc dữ liệu từ tệp tin

Như ở ví dụ trên, dữ liệu kiểm thử đầu tiên được đọc cho biến 'data' và sau đó được chia thành các dòng, bỏ qua phần đầu của dòng. Sau đó các dòng dữ liệu được xử lý lần lượt. Các dòng được chia thành các ô bởi kí tự \t và dữ liệu ở các ô được gán cho các biến, khiến cho dữ liệu kiểm thử đã sẵn sàng cho thực hiện kiểm thử thật.

Ở phần đọc dữ liệu từ tệp tin - hàm đọc dữ liệu có thể bị lỗi nếu như có dữ liệu trống hay các cột không được sắp xếp. Nếu có nhiều kiểm thử chức năng được yêu cầu

hơn từ bộ phân tích (Parser) thì nó nên được cài đặt như là một mô-đun dùng chung mà tất cả đoạn mã điều khiển (Driver scrtips) có thể sử dụng như ở Hình 6. [3]

2.2.4.Hứa hẹn và vấn đề của kiểm thử hướng dữ liệu

Có rất nhiều lợi ích của kiểm thử hướng dữ liệu:

- Sửa kịch bản kiểm thử và thêm mới dữ liệu không cần đòi hỏi kĩ năng lập trình
- Dữ liệu test có thẻ chuẩn bị sớm, trước cả khi cài đặt kiểm thử hay là trước khi hệ thống cần test hoàn thành
- Có thể được tái sử dụng khi kiểm thử thủ công
- Giúp bảo trì: Khi hệ thống cần kiểm thử có thay đổi, thường sẽ dẫn đến sự thay đổi của hoặc dữ liệu kiểm thử hoặc mã nguồn, và trách nhiệm bảo trì có thể được chia sẻ cho những người khác nhau.

Tuy nhiên, cũng có môt số điểm yếu của kiểm thử hướng dữ liệu đó là đối với các bài kiểm thử có dữ liệu giống nhau, việc tạo mới kiểu kiểm thử sẽ phải cài đặt thêm một đoạn mã mới mà có thể đọc hiểu được dữ liệu.Ví dụ kiểm thử như trong Hình 7 và Hình 8 được thiết kế để thực hiện phép tính toán chỉ với hai dòng cần phải thay đổi nhiều để có thể xử lý được các bài kiểm thử kiểu 5*8+2=42. Tổng quan, dữ liệu kiểm thử và đoạn mã điều khiển được đặc biệt khuyến nghị đồng bộ hóa nếu một trong hai có thay đổi.

2.3. Kiểm thử hướng từ khóa (Keyword-driven testing)

2.3.1.Giới thiệu

Phần giới thiệu về kiểm thử hướng dữ liệu (data-driven testing) đã cho thấy đây là một hướng tiếp cận có nhiều hứa hẹn, nhưng đồng thời cũng cho thấy giới hạn lớn nhất của nó đó là tất cả các bài kiểm thử là tương đương nhau và việc tạo ra các bài kiểm thử hoàn toàn mới thì yêu cầu phải có nỗ lực lập trình. Một giải pháp cho giới hạn này, được đưa ra bởi Fewster và Graham và Kaner đó là tiếp cận hướng từ khóa mà trong đó không chỉ có dữ liệu kiểm thử mà cả những chỉ đạo về việc làm gì với dữ liệu kiểm thử cũng được tách ra khỏi mã nguồn và đưa ra các tệp

tin bên ngoài. Những chỉ đạo này được gọi là từ khóa (Keywords) và các kĩ sư kiểm thử có thể sử dụng chúng để xây dựng nên bài kiểm thử một cách dễ dàng. [4]

Ý tưởng đơn giản cũng giống như trong kiểm thử hướng dữ liệu đó là đọc dữ liệu kiểm thử từ một tệp tin bên ngoài và chạy các kiểm thử dựa trên đó. Như Fewster và Graham, kiểm thử hướng từ khóa chính là mở rộng của kiểm thử hướng dữ liệu. [4]

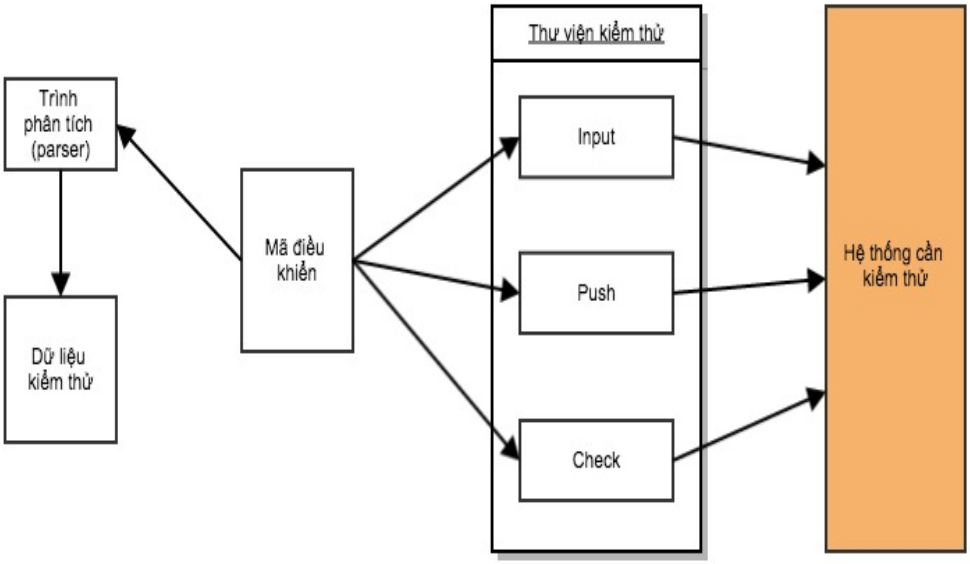
Hình 9 là một ví dụ về dữ liệu của kiểm thử hướng từ khóa mà bao gồm có hai bài kiểm thử đơn giản để kiểm thử ứng dụng Calculator. Bài kiểm thử bao gồm các từ khóa Input và Push và Check và các đối số là những giá trị đầu vào và kết quả mong muốn của bài kiểm thử. Như có thể thấy, điều này là dễ dàng để thêm những bài kiểm thử khác mà không cần phải viết thêm từ khóa.

	A	B	C
1	**Test case**	**Keyword**	**Argument 1**
2	Add 01	Input	1
3		Push	+
4		Input	2
5		Push	=
6		Check	3
7	Longer 01	Input	5
8		Push	*
9		Input	8
10		Push	+
11		Input	2
12		Push	=
13		Check	42

Hình 9: Kiểm thử hướng từ khóa

2.3.2.Sửa đổi và lưu trữ dữ liệu kiểm thử

Có thể sử dụng một bảng dạng HTML hoặc sử dụng một cơ sở dữ liệu trung tâm với tính mở rộng cao hơn để lưu trữ dữ liệu. Sau đó dữ liệu kiểm thử sẽ được đọc lên bởi mã điều khiển tương ứng với từng từ khóa thông qua trình phân tích như trong minh họa ở Hình 10. Khi có thay đổi về dữ liệu kiểm thử, kiểm thử viên chỉ cần sửa bản thân dữ liệu đó thôi mà không cần phải sửa đổi mã nguồn kiểm thử.



Hình 10: Đưa các từ khóa ra thư viện

Như ở Hình 10, các từ khóa ở trong Hình 9 đã được tạo ra và xuất ra thành thư viện riêng và có thể sử dụng trong nhiều kịch bản kiểm thử.

2.3.3. Xử lý dữ liệu kiểm thử

Xử lý dữ một tệp dữ liệu kiểm thử hướng từ khóa được minh họa như trong Hình 9 không có sự khác biệt lớn so với việc xử lý một tệp dữ liệu kiểm thử hướng dữ liệu. Trình phân tích, cũng giống như ở Hình 8, có thể lấy từ khóa và các đối số của nó vào mã nguồn điều khiển.

Sau khi một dữ liệu kiểm thử được phân tích, mã nguồn điều khiển phải có thể biên dịch từ khóa, và thực thi các hành động sử dụng những đối số đã được gán. Để giữ cho khung được mô đun hóa, tốt hơn là nên thực hiện cài đặt riêng từng từ khóa và đặt chúng ở thư viện ngoài như ở Hình 10.

2.3.4.Từ khóa ở các mức khác nhau

Một trong những quyết định quan trọng cần thực hiện khi thiết kế từ khóa đó là mức của từ khóa có thể sử dụng.Trong Hình 8, từ khóa được sử dụng ở mức khá thấp (như là Input, Push), làm cho chúng trở nên phù hợp với kiểm thử mức chi tiết ở mức giao diện. Khi kiểm thử chức năng ở mức cao hơn, như là mức logic nghiệp vụ (Business logic), từ khóa mức thấp thường khiến cho các bài kiểm thử rất dài và

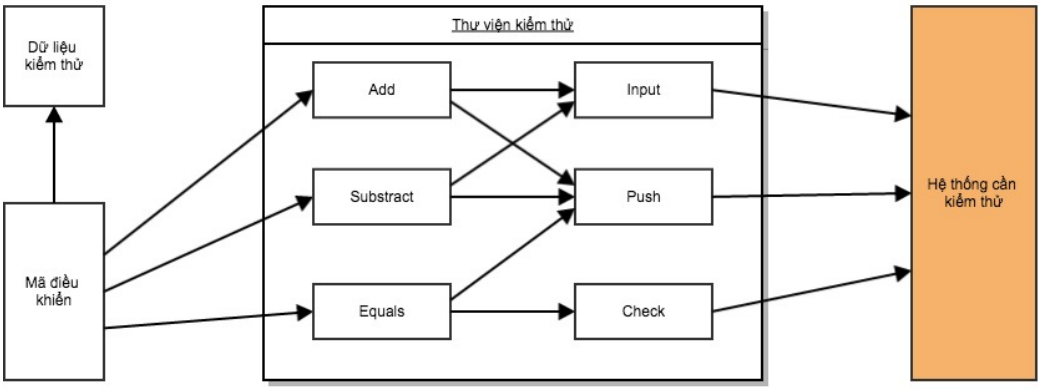
các từ khóa ở mức cao là có khả năng sử dụng hơn. Đôi khi có thể sử dụng từ khóa vừa mức thấp, vừa mức cao cùng nhau. Kiểm thử viên có thể tự xây dựng các từ khóa mức thấp, sau đó tiếp tục xây dựng các từ khóa mức cao dựa trên những từ khóa mức thấp đã được xây dựng. Ví dụ như từ khóa mức thấp Input và Push có thể dùng để xây dựng lên các từ khóa mức cao như Add, Subtract, Multiply, Devide và tương tự từ khóa Equals có thể được tạo sử dụng từ Push và Check. Hình 11 cũng cho thấy từ khóa mức cao làm cho các bài kiểm thử trở nên ngắn hơn.

	A	B	C
1	**Test case**	**Keyword**	**Argument 1**
2	Add 01	Input	1
3		Add	2
4		Equals	3
5	Longer 01	Input	5
6		Multiply	8
7		Add	2
8		Equals	42

Hình 11: Sử dụng các từ khóa ở mức cao

Một cách đơn giản để xây dựng từ khóa mức cao mới là để cho các lập trình viên mà phát triển nên khung kiểm thử tự động hướng từ khóa xây dựng sẵn các từ khóa mức thấp và mức cao tương ứng và kiểm thử viên chỉ cần sử dụng lại những từ khóa này khi tạo bài kiểm thử mà không cần viết thêm mới như trong

Tuy nhiên việc cài đặt sẵn từ khóa trong khung kiểm thử có những giới hạn của nó, đó là kiểm thử viên sẽ bị giới hạn trong những từ khóa có sẵn trong thư viện và khó để có thể viết thêm mới những từ khóa khác.



Hình 12: Xây dựng các từ khóa mức cao trong thư viện kiểm thử

Một giải pháp linh hoạt hơn cho vấn đề này đó là để cho kiểm thử viên tự xây dựng các từ khóa ở mức độ cao, bằng cách sử dụng giao diện giống như giao diện họ sử dụng để thiết kế bài kiểm thử. Những kiểu kĩ thuật này được giới thiệu với nhiều cách cài đặt khác nhau bởi Buwalda et al và Nagle [4]. Ví dụ họ có thể sử dụng các từ ngữ giống như trong bài kiểm thử để miêu tả từ khóa, để cho các kiểm thử viên khác không trực tiếp viết mã nguồn cho những từ khóa này cũng có thể hiểu được nội dung của từ khóa để làm gì. Hình 13 cho thấy việc các từ khóa mới được xây dựng như thế nào .Để phân biệt những từ khóa được cài đặt trực tiếp trong thư viện và những từ khóa được viết thêm mới, người ta ta thường gọi những từ khóa được viết thêm mới là các từ khóa người dùng hay là từ khóa dùng chung (User keyword và base keyword).

	A	B	C	D
1	**User Keyword**	**Keyword**	**Argument 1**	**Argument 2**
2	Add	Arguments	${Number}	
3		Push	+	
4		Input	${Number}	
5	Multiply	Arguments	${Number}	
6		Push	*	
7		Input	${Number}	
8	Equals	Arguments	${Expected}	
9		Push	=	
10		Check	${Expected}	

Hình 13 Tạo ra các từ khóa mức cao từ hệ thống thiết kế kiểm thử

Lợi ích chính của việc cho phép tạo ra những từ khóa mới sử dụng hệ thống thiết kế kiểm thử đó là các từ khóa mới có thể được tạo ra và được bảo trì dễ dàng hơn mà không đòi hỏi phải có kĩ năng lập trình. Tất nhiên có thể thấy rằng việc tạo từ khóa mới chính là lập trình rồi nhưng ít nhất việc lập trình này cũng dễ dàng hơn và các kiểm thử viên tự động hóa có thể học.

2.3.5.Những khó khăn và hứa hẹn

Kiểm thử hướng từ khóa có tất cả những lợi ích giống như kiểm thử hướng dữ liệu. Như đã đề cập, điểm có lợi chính hơn hẳn kiểm thử hướng dữ liệu đó là kỹ thuật hướng từ khóa khiến cho kiểm thử viên – kể cả những người không có kỹ năng lập trình dễ dàng hơn trong việc tạo ra những bộ kiểm thử mới.

So sánh Hình 7: **Dữ liệu kiểu kiểm thử hướng dữ liệu** và Hình 9: Kiểm thử hướng từ khóa sẽ minh chứng cho điều này rất rõ ràng: Kiểm thử hướng từ khóa là bước lớn từ kiểm thử hướng dữ liệu khi mà trong kiểm thử hướng dữ liệu tất cả các bài kiểm thử đều giống nhau và việc tạo ra bộ kiểm thử mới yêu cầu phải thực hiện viết mã nguồn mới còn trong kiểm thử hướng từ khóa thì không.

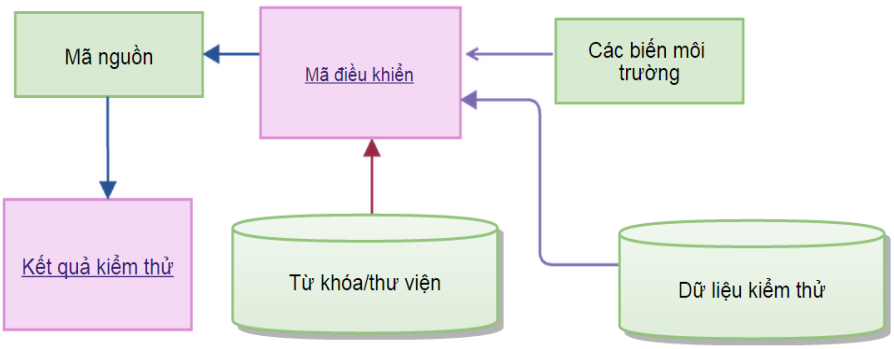
Vấn đề với kiểm thử hướng từ khóa đó là bài kiểm thử thường rất dài và phức tạp hơn so với việc sử dụng tiếp cận hướng dữ liệu. Ví dụ, bài kiểm thử Add 001 trong Hình 7 chỉ 1 dòng nhưng trong Hình 9 thì tốn 5 dòng. Điều này có thể được giải quyết nhờ sử dụng từ khóa mức cao như ở Hình 13. Vấn đề chính của kiểm thử hướng từ khóa đó là nó yêu cầu khung kiểm thử phức tạp hơn kiểm thử hướng dữ liệu.

2.4.Phương pháp tích hợp kiểm thử hướng dữ liệu và từ khóa

Nếu như kiểm thử hướng từ khóa xuất hiện đã loại bỏ được những nhược điểm của kiểm thử tự động hướng dữ liệu thì việc áp dụng cả hai giải pháp này sẽ giúp tạo ra một khung làm việc có tất cả các ưu điểm của hai giải pháp và hạn chế nhược điểm của từng giải pháp.

Như minh họa ở Hình 14, trong khung làm việc tích hợp giữa kiểm thử hướng dữ liệu và từ khóa, dữ liệu kiểm thử được lưu trữ riêng rẽ ra khỏi mã nguồn kiểm thử. Khi có thay đổi về mặt dữ liệu kiểm thử, chỉ ảnh hưởng đến việc sửa đổi dữ liệu mà hạn chế ảnh hưởng đến mã nguồn. Mã nguồn kiểm thử được phát triển theo

hướng từ khóa, mỗi bài kiểm thử sẽ được ghép lại bởi nhiều từ khóa, tận dụng tối đa việc tái sử dụng các từ khóa.



Hình 14: Tích hợp kiểm thử hướng dữ liệu và từ khóa

Việc áp dụng giải pháp kiểm thử tự động hướng từ khóa và hướng dữ liệu có rất nhiều lợi ích như sau: [13]

1. Tận dụng thế mạnh của từng thành viên trong đội kiểm thử

Hầu hết các thành viên trong tổ kiểm thử nắm vững chắc qui tắc nghiệp vụ của hệ thống cần kiểm thử, nhưng có ít kinh nghiệm trong lập trình. Còn những thành viên chuyên thực hiện vai trò kỹ sư kiểm thử tự động thường là các thành viên có kinh nghiệm về lập trình hoặc khoa học máy tính, nhưng lại ít có kinh nghiệm trong kiểm thử, hoặc nghiệp vụ và khó khăn trong việc thiết kế ra các bài kiểm thử có chất lượng cao. Việc áp dụng giải pháp kiểm thử hướng từ khóa sẽ giúp cho tất cả các thành viên trong tổ kiểm thử đều có thể tham gia vào qui trình kiểm thử tự động bằng cách thiết kế các từ khóa ở mức tương ứng. Đó là kiểm thử viên có kỹ năng kiểm thử tốt sẽ thiết kế các bài kiểm thử cùng với các từ khóa mức cao. Việc tiếp theo các kiểm thử viên có kỹ năng lập trình sẽ thực hiện viết các từ khóa kiểm thử ở mức thấp. Cách tiếp cận này cho phép kiểm thử viên tập trung vào việc tạo ra những bài kiểm thử có chất lượng tốt, trong khi kỹ sư kiểm thử tự động sẽ tập trung vào các khó khăn thách thức về kỹ thuật khi thực hiện viết mã nguồn kiểm thử.

2. Giảm thiểu tối đa việc bảo trì kiểm thử tự động

Bênh cạnh lợi ích của kiểm thử hướng dữ liệu về việc giảm thiểu bảo trì mã nguồn khi dữ liệu kiểm thử thay đổi, kiểm thử hướng từ khóa cũng giúp giảm bảo trì kiểm thử tự động bằng cách cho phép người sử dụng định nghĩa những bài kiểm

thử của họ ở mức nghiệp vụ. Thay vì định nghĩa các bài kiểm thử như là một chuỗi các hoạt động tương tác với giao diện, kiểm thử viên có thể định nghĩa các bài kiểm thử như là một chuỗi các hoạt động nghiệp vụ. Ví dụ một ứng dụng ngân hàng có thể bao gồm hành động "Mở tài khoản mới", "Rút tiền", "Gửi tiền". Mặc dù nếu giao diện của hệ thống thay đổi, qui trình thực hiện nghiệp vụ vẫn như vậy, do đó kiểm thử viên thiết kế kiểm thử không cần phải thay đổi bài kiểm thử. Mà đó là công việc của kiểm thử viên tự động phải cập nhật các từ khóa do ảnh hưởng bởi thay đổi của giao diện, và việc này thông thường chỉ diễn ra ở một nơi.

3. Nâng cao chất lượng kiểm thử tự động

Trong kiểm thử tự động hướng từ khóa, kiểm thử viên thiết kế các bài kiểm thử tuân theo hướng tiếp cận từ trên xuống (Top-down) để đảm bảo rằng có mục tiêu cho mỗi bài kiểm thử.

Bước thứ nhất đó là xác định xem làm thế nào toàn bộ bài kiểm thử tự động có thể được chia nhỏ thành các mô-đun kiểm thử riêng biệt. Sau đó định nghĩa yêu cầu cho mỗi mô đun. Yêu cầu này rất quan trọng bởi nó giúp cho lập trình viên xem xét cái gì đang được kiểm thử ở mỗi mô đun và có thể ghi chú lại chính xác. Sau khi đã định nghĩa yêu cầu kiểm thử, kiểm thử viên tự động – lập trình viên có thẻ bắt đầu viết các ca kiểm thử sử dụng các từ khóa đã được xác định trước hoặc định nghĩa mới. Kiểm thử viên có thể thiết kế bài kiểm thử của họ tương ứng với các qui trình nghiệp vụ mức cao, điều đó khiến cho bài kiểm thử dễ đọc hơn các bài kiểm thử thiết kế sử dụng các tương tác ở mức thấp.

4. Chiến lược kiểm thử tự động thuận lợi

Kiểm thử hướng từ khóa cung cấp một khung làm việc mà tích hợp toàn bộ tổ chức kiểm thử trong việc hỗ trợ để kiểm thử tự động đạt hiệu quả. Như nhân viên phân tích nghiệp vụ (Business Analysts), kiểm thử viên, kỹ sư tự động hóa, nhóm trưởng…có thể làm việc chung trong cùng một khung làm việc để hoàn thành việc lập kế hoạch kiểm thử, thiết kế kiểm thử cũng như thực thi kiểm thử.

5. Cho phép hợp tác hiệu quả giữa các đội kiểm thử làm việc phân tán

Với việc các đội kiểm thử thường được phân bổ ở nhiều nơi trong một quốc gia hay trên toàn thế giới, thách thức trong việc chia sẻ thông tin, các bài kiểm thử và thư viện kiểm thử tự động được nhân lên gấp nhiều lần. Kiểm thử hướng từ khóa cung cấp một khuôn khổ đã được chứng minh trong việc tổ chức các tài liệu kiểm thử, thư viện kiểm thử tự động với cấu trúc rõ ràng, ngăn chặn sự gián đoạn có thể xảy ra bởi sự khác biệt múi khoảng cách và thời gian.

Chương II đã giới thiệu về hai phương pháp luận trong việc tiếp cận khung kiểm thử tự động đó là kiểm thử hướng dữ liệu và kiểm thử hướng từ khóa. Việc áp dụng hai phương pháp luận này trong tự động hóa thực thi kiểm thử hồi qui cho chức năng "Get changes/ Post changes" của phần mềm Ads Manager sẽ được trình bày ở chương III.

CHƯƠNG 3. THỬ NGHIỆM KIỂM THỬ HƯỚNG DỮ LIỆU VÀ TỪ KHÓA

3.1. Mô tả đối tượng kiểm thử

3.1.1.Phần mềm Ads Editor

Ads Editor là một ứng dụng quản lý các chiến dịch quảng cáo trực tuyến (Ads advertising campaigns), hoạt động trên nền Window, cho phép người dùng tải dữ liệu, sửa đổi và đẩy dữ liệu từ máy tính lên server của Google Adword.

Bất kỳ người dùng sở hữu tài khoản nào của Adword cũng có thể sử dụng công cụ Ads Editor để quản lý dữ liệu, nhưng công cụ này sẽ đặc biệt hữu ích cho các tài khoản có nhiều chiến dịch và với khối lượng lớn từ khóa. Một số chức năng cơ bản của ứng dụng này như sau:

- Quản lý các tài khoản quảng cáo của người dùng
- Quản lý các chiến dịch quảng cáo: Thêm mới, xóa, sửa dữ liệu của chiến dịch quảng cáo– Campaign, nhóm quảng cáo – Ad group, dữ liệu quảng cáo văn bản – Text ads, từ khóa – Keyword
- Cho phép thay đổi nhiều dữ liệu cùng lúc
- Xuất và nhập các tệp tin để chia sẻ các data thay đổi cho một tài khoản
- Xem dữ liệu thống kê (statistic) cho tất cả hay một tập các chiến dịch quảng cáo
- Sao chép/di chuyển dữ liệu giữa các nhóm dữ liệu, chiến dịch quảng cáo

Qui trình cơ bản để sử dụng công cụ Ads Editor như sau: Tải một hoặc nhiều tài khoản, thực hiện thay đổi ở trên máy tính cá nhân rồi đẩy dữ liệu lên máy chủ Google Adword. Sau đây là các bước chi tiết:

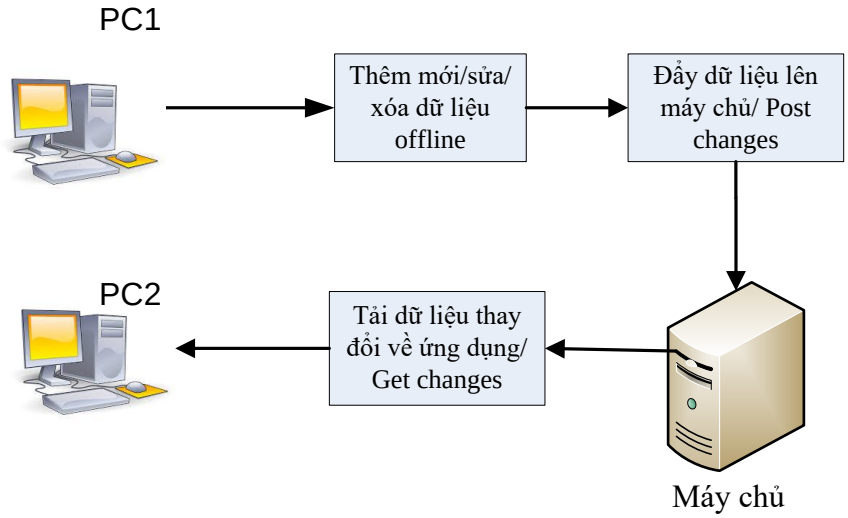
1. Tải dữ liệu của tài khoản: Có thể tải dữ liệu của một hoặc nhiều tài khoản.
2. Thực hiện các thay đổi dữ liệu tài khoản: Có thể thay đổi một vài dữ liệu hay thay đổi lượng lớn dữ liệu
3. Chia sẻ (Share) thay đổi này cho người phụ trách review: Người phụ trách có thể kiểm tra thay đổi, sửa đổi rồi xuất ngược lại ra tệp tin.
4. Đẩy dữ liệu lên máy chủ (Post changes): Sau khi thay đổi dữ liệu, người dùng có thể đẩy dữ liệu lên máy chủ Google Ads. Sau khi đẩy dữ liệu thành công, dữ liệu người dùng sẽ xuất hiện trên trang web của Google Ads

5. Tải dữ liệu thay đổi từ máy chủ (Get changes): Người dùng thực hiện cập nhật các dữ liệu thay đổi của toàn bộ chiến dịch hoặc một số chiến dịch đã được tải về ứng dụng trên máy tính để đảm bảo dữ liệu luôn luôn được cập nhật mới nhất.

Trong các bước thuộc qui trình trên của ứng dụng Ads Editor, bước 4 và bước 5 là các bước rất quan trọng, luôn luôn được kiểm thử hồi qui mỗi khi có thay đổi phiên bản của ứng dụng.

3.1.2.Chức năng "Post Changes" và "Get Changes" dữ liệu

Sau khi đăng nhập vào ứng dụng, người dùng thực hiện thay đổi dữ liệu như thêm mới, sửa, xóa dữ liệu của từng chiến dịch, sau đó đẩy dữ liệu lên máy chủ, như hình minh họa dưới đây:



Hình 15: Mô hình "Post changes" và "Get changes" dữ liệu

Nếu một trong hai chức năng này không hoạt động được hoặc hoạt động sai, sẽ ảnh hưởng trực tiếp đến doanh thu quảng cáo của dữ liệu. Do vậy việc kiểm thử các bài kiểm thử của các chức năng này là đặc biệt quan trọng.

3.1.3.Các vấn đề cần khắc phục khi kiểm thử chức năng "Post changes" và "Get changes" dữ liệu

Để thực hiện kiểm thử được một bài kiểm thử "post change/ get changes", hiện nay kiểm thử viên phải thực hiện các công việc như sau:

Bước 1: Chuẩn bị 2 ứng dụng đăng nhập cùng tài khoản và tải một số chiến dịch (campaign) giống nhau

Bước 2: Ở app 1, kiểm thử viên thực hiện thay đổi dữ liệu cho từng trường của chiến dịch, sau đó đẩy dữ liệu lên máy chủ

Bước 3: Ở app2, thực hiện tải dữ liệu thay đổi về ứng dụng trên máy tính, sau đó so sánh dữ liệu đã tải về với dữ liệu đã được đẩy lên trên máy chủ xem có giống nhau không.

Lặp lại bước từ 1 đến 3 với các "ad group/text ad/ keywords" tùy theo từng bài kiểm thử.

Hiện nay công ty đang sử dụng mô hình phát triển phần mềm Scrum, mỗi giai đoạn ngắn (sprint) sẽ nâng cấp hoặc sửa đổi, thêm mới một số chức năng cho ứng dụng. Trước khi gửi sản phẩm đến cho khách hàng, luôn luôn có quá trình kiểm thử hồi qui, trong đó phần kiểm thử "post change/get changes" là phần quan trọng và chiếm rất nhiều thời gian kiểm thử do thao tác sửa đổi dữ liệu, so sánh dữ liệu lấy về và dữ liệu đã đẩy lên trên máy chủ chiếm nhiều thời gian. Mỗi giai đoạn (sprint) dài khoảng một tháng, vậy nên lượng kiểm thử hồi qui là rất lớn, tốn nhiều chi phí và nguồn lực.

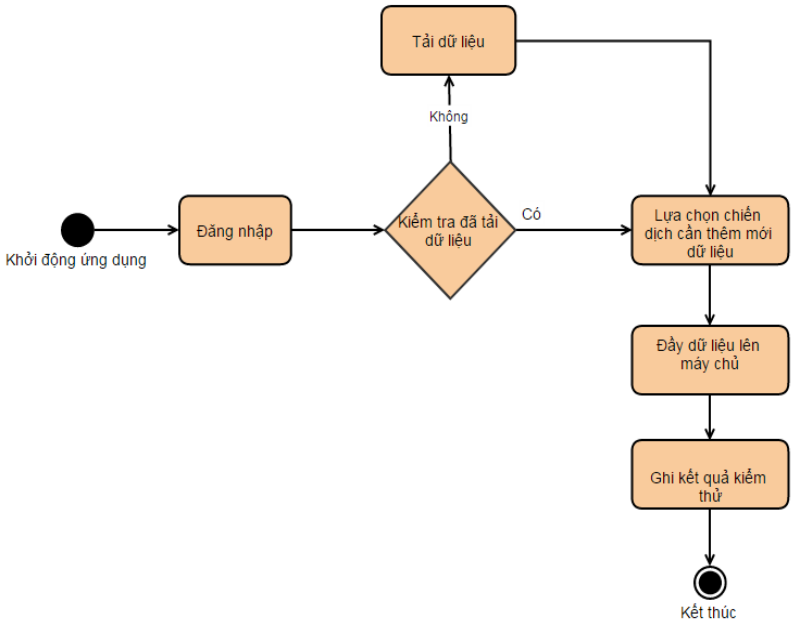
Vì vậy, vấn đề đặt ra đó là làm sao để có thể tự động hóa quá trình kiểm thử, mà trước mắt là tự động hóa cho kiểm thử chức năng post change/ get changes, nhằm giảm thiểu nhiều nhất chi phí cho việc kiểm thử mà vẫn đảm bảo chất lượng sản phẩm?

3.2.Yêu cầu tự động hóa kiểm thử

Từ các vấn đề như trên, cần phải xây dựng một hệ thống giúp cho kiểm thử viên thực hiện kiểm thử tự động "post change/ get changes" dữ liệu. Gọi hệ thống này là "Hệ thống kiểm thử tự động" (HTKTTĐ). Dưới đây là các yêu cầu mà HTKTTĐ phải đáp ứng:

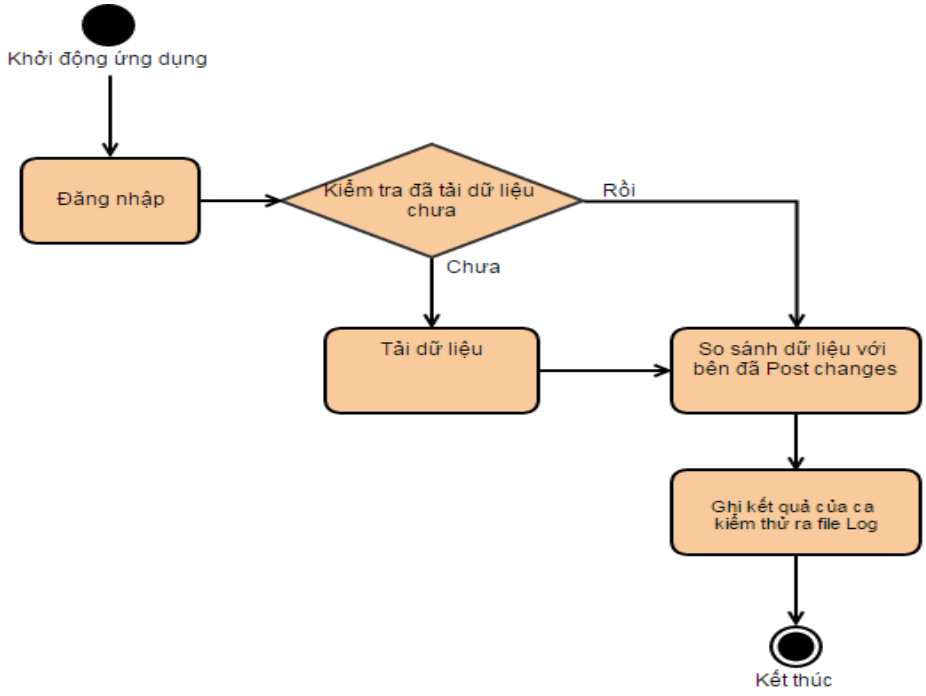
- Đọc dữ liệu kiểm thử từ một tệp tin excel: Tệp tin này bao gồm các dữ liệu chỉnh sửa của "campaign, ad group, text ad, keyword" theo từng bài kiểm thử

- Thực hiện "Post change" tự động cho ứng dụng thứ nhất. Hình 16 dưới đây minh họa việc thêm mới dữ liệu ở trên ứng dụng Ads Editor và sau đó đẩy ứng dụng lên trên máy chủ.



Hình 16: Sơ đồ thực hiện thêm mới dữ liệu và đẩy lên máy chủ ở ứng dụng 1

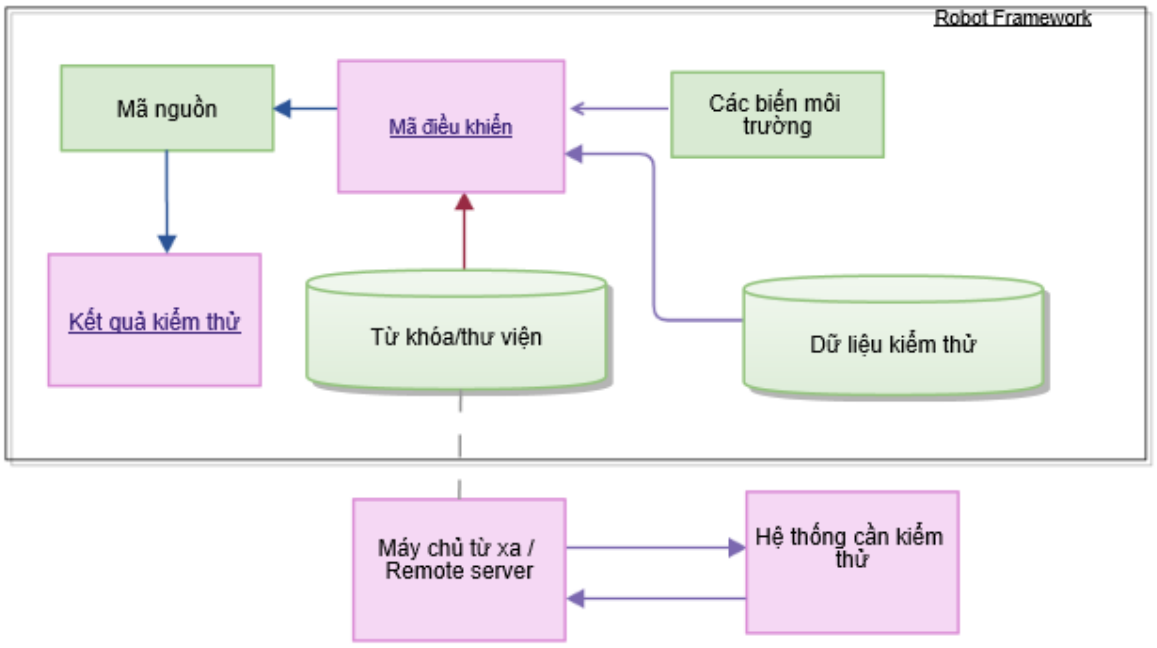
- Thực hiện "Get changes" tự động và so sánh dữ liệu "Get changes" ở ứng dụng 2 với dữ liệu đã "Post changes" ở ứng dụng thứ nhất. Xuất kết quả ra tệp tin log để kiểm thử viên có thể kiểm tra được kết quả của bài kiểm thử.



Hình 17: Sơ đồ thực hiện tải dữ liệu và so sánh ở ứng dụng 2

3.3.Môi trường thử nghiệm

Để ứng dụng giải pháp kiểm thử tự động sử dụng cả kiểm thử tự động hướng dữ liệu và kiểm thử hướng từ khóa, kiến trúc kiểm thử ứng dụng Ads Editor được thể hiện như ở dưới đây:



Hình 18: Môi trường kiểm thử tự động ứng dụng Ads Editor

Trong đó khung kiểm thử tự động là Robot framework, các dữ liệu kiểm thử được tách riêng, mã nguồn được xây dựng hướng từ khóa và có một số từ được xuất ra tệp tin thư viện. Việc giao tiếp giữa Robot framework với ứng dụng cần kiểm thử được thực hiện thông qua thư viện từ xa và máy chủ từ xa Ranorex.[17]

3.3.1.Khung kiểm thử tự động Robot framework

Dựa vào kết quả nghiên cứu phương pháp kiểm thử tự động cùng với yêu cầu của tự động hóa kiểm thử bằng cách sử dụng các công cụ miễn phí, em đã lựa chọn công cụ Robot Framework để áp dụng cho luận văn của mình.

Robot Framework là một chương trình mã nguồn mở, cung cấp một nền tảng kiểm thử dựa trên ngôn ngữ lập trình Python. Cách tiếp cận của nền tảng kiểm thử này là hướng từ khoá (keyword-driven testing). Ý tưởng cho Robot framework được định hình trong luận văn của Pekka Klarch's trong năm 2005. Cùng năm đó, phiên bản đầu tiên được phát triển tại Nokia Siement Network.[18]s

Robot Framework cung cấp cho chúng ta mọi thứ cần thiết để có thể phát triển một kịch bản kiểm thử như điều kiện đầu vào, điều kiện kết thúc, các báo cáo kiểm thử… Các tính năng của Robot Framework hỗ trợ chúng ta thiết kế các kịch bản kiểm thử ở dạng bảng một cách dễ dàng.

Robot Framework đưa ra kết quả thực thi các kịch bản kiểm thử và các log ở dạng HTML. Điều này giúp chúng ta đọc và phân tích kết quả dễ dàng và nhanh chóng.

Robot Framework hỗ trợ chức năng đánh dấu các kịch bản kiểm thử và chúng ta có thể lựa chọn các kịch bản kiểm thử để thực thi một cách nhanh chóng.

Cuối cùng, thế mạnh nhất của Robot Framework so với các nền tảng kiểm thử khác là khả năng chạy trên nhiều hệ điều hành khác nhau mà không cần chỉnh sửa kịch bản kiểm thử hay các từ khoá ở tầng dưới.

Cài đặt Robot Framework

Trước hết, vì Robot Framework là một nền tảng kiểm thử dựa trên nền tảng Python cho nên, chúng ta phải cài đặt Python trước khi nói đến Robot Framework.

Sau khi đã cài đặt Python, để cài đặt Robot Framework trên hệ điều hành Window, chỉ cần mở cửa sổ Command Line ở thư mục cài đặt Python và thực thi câu lệnh: easy_install robotframework. Như vậy là Robot Framework đã sẵn sàng để sử dụng trên Windows.

Robot framework sử dụng cú pháp lập trình đơn giản, như dùng java hoặc python, có thể sử dụng cả thư viện sẵn có cũng như thư viện bên ngoài đã tích hợp. Ngoài ra người dùng có thể tự tạo thư viện cho mình.

Thư viện của Robot framework

Các từ khóa có sẵn của Robot framework được định nghĩa ở trong các thư viện. Thư viện mới có thể được tạo ra bằng ngôn ngữ Python hoặc Java. Có hai loại thư viện là thư viện chuẩn và thư viện ngoài (Standard and external). Các thư viện chuẩn được phân phối bởi Robot framework và các thư viện ngoài được phát hành trong các gói riêng rẽ.[11]

Thư viện chuẩn:

Bảng 5: Các thư viện chuẩn của Robot Framework

Tên thư viện	Giải thích
Built In	Chứa từ khóa liên quan nhu cầu kiểm thử chung như xác minh biến, chuyển đổi, trì hoãn (delay)
OperatingSystem	Chứa từ khóa liên quan đến các tác vụ của hệ điều hành như là thực thi các câu lệnh
Telnet	Chứa từ khóa liên quan đến xử lý kết nối telnet
Collections	Chứa từ khóa liên quan đến xử lý danh sách và các từ điến.
String	Chứa từ khóa liên quan đến xử lý chuỗi
Dialog	Chứa từ khóa liên quan đến việc lấy thông tin người dùng nhập vào trong khi đang thực thi kiểm thử.
Screen shot	Chứa từ khóa cho việc chụp và lưu ảnh chụp màn hình
Remote	Proxy giữa Robot Framework và một thư viện

Thư viện ngoài

Bảng 6: Các thư viện ngoài của Robot Framework

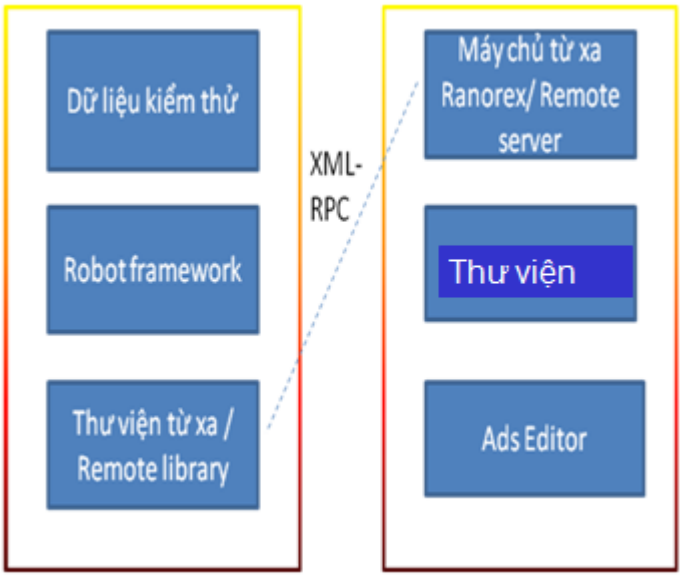
Swings	Chứa từ khóa xử lý các giao diện của phần mềm xây dựng trên Java Swing
SSH	Chứa từ khóa thực thi các câu lệnh thông qua kết nối SSH
AutoIt	Từ khóa để kiểm thử các ứng dụng Window với công cụ AutoIt
Database	Từ khóa để xử lý cơ sở dữ liệu
Selenium	Từ khóa để kiểm thử ứng dụng web với công cụ Selenium

3.3.2.Ranorex

Robotframework không tương tác trực tiếp được với ứng dụng cần kiểm thử Ads Editor mà cần tương tác thông qua một thư viện từ xa Ranorex (Remote library).

Từ thư viện từ xa của Ranorex (Remote library) sẽ kết nối đến máy chủ từ xa của Ranorex (Remote server) để truyền các mã nguồn điền khiển đến ứng dụng cần kiểm thử Ads Editor và chuyển những trạng thái của ứng dụng cần kiểm thử trở lại cho Robot framework.

Thư viện từ xa đóng vai trò như là proxy giữa Robot Framework và máy chủ từ xa Ranorex.

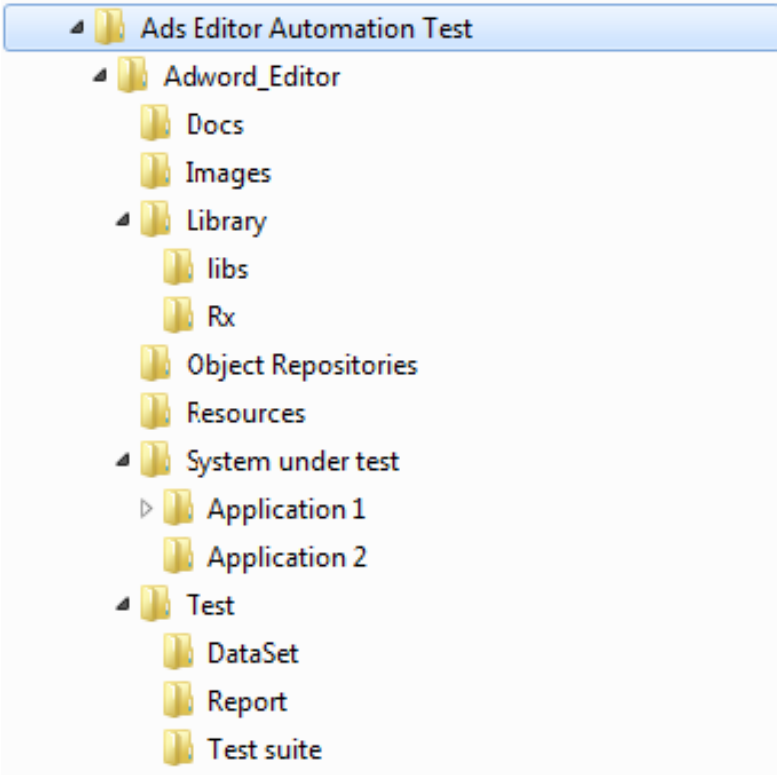


Hình 19: Kiến trúc bậc cao thể hiện giao tiếp của Robot framework và Ads Editor

Trước khi thực thi kiểm thử tự động bằng Robot framework, cần có thao tác khởi động thư viện từ xa.

3.4.Thiết kế kiểm thử hướng dữ liệu và từ khóa

Cấu trúc thư mục các dữ liệu, tài nguyên, thư viện liên quan đến kiểm thử tự động được bố trí như sau:



Hình 20: Cấu trúc thư mục kiểm thử tự động

Trong đó:

- Docs: Chứa các tài liệu hướng dẫn liên quan đến kiểm thử tự động, như cách cài đặt Robot framework, giới thiệu về kiểm thử hướng dữ liệu và kiểm thử hướng từ khóa…

- Images: Chứa các hình ảnh để nhận diện đối tượng trong trường hợp cần nhận diện các đối tượng bằng hình ảnh

- Library: Nơi chứa các thư viện từ khóa, thư viện từ xa

- Object Repositories: Nơi chứa tệp tin Gui_Mapping là tệp tin cho phép ánh xạ giữa tên dễ hiểu của đối tượng, phần tử với ID của nó

- Resource: Nơi chứa các tài nguyên như từ khóa, biến

- System under test: Nơi chứa thư mục cài đặt của ứng dụng cần kiểm thử, trong trường hợp này chúng ta có hai thư mục, một là ứng dụng để Get changes, hai là ứng dụng để Post changes

- Test: Nơi chứa các tài liệu như dữ liệu kiểm thử, bài kiểm thử (test case), các bộ kiểm thử, báo cáo.

3.4.1.Thiết kế bài kiểm thử

Với từng bài kiểm thử đã được chuẩn bị bởi các kiểm thử viên có kỹ năng, kinh nghiệm kiểm thử và thiết kế ra những bài kiểm thử có chất lượng, kiểm thử viên có kỹ năng lập trình sẽ viết mã nguồn tương ứng cho từng bài kiểm thử. Thông thường, kiểm thử viên tự động hóa (Test automator) sẽ đọc từng bước trong bài kiểm thử, thiết kế từ khóa cho từng bước, sau đó tùy thuộc vào việc có thể tái sử dụng của từ khóa đó không mà sẽ đưa nó vào thư viện dùng chung hay không. Tuy nhiên kiểm thử viên tự động – người viết mã nguồn phải trình bày các bước ở trong bài kiểm thử tự động một cách dễ hiểu sao cho những người không biết đến mã nguồn cũng phải hiểu được ý định của bài kiểm thử làm gì, và cần đầu vào đầu ra như thế nào.

Dưới đây sẽ trình bày các bước khi thiết kế bài kiểm thử "Post changes thành công các ad group đã được thêm mới/ Post changes added new ad group sucessfully". Mục tiêu của bài kiểm thử này đó là thực hiện kiểm tra việc các dữ liệu "ad group" được thêm mới ở ứng dụng có thể được đẩy lên máy chủ thành công không, sau đó ở ứng dụng khác có thể tải được những thay đổi này về, đồng thời dữ liệu của hai bên có bị thay đổi, sai khác gì không.

Các bước trong bài kiểm thử mà kiểm thử viên chuẩn bị như ở Bảng 7.

Bảng 7: Bài kiểm thử "Post changes added new ad group sucessfully"

Điều kiện tiền đề	Các bước thực hiện kiểm thử	Kết quả mong muốn
- Have download campaign C	Application 1: 1. Select campaign C 2. Click on tab [Ad group] 3. Click on button [Add new] 4. Edit data as in file data test - For example: Name = A, max cpc =1 5. Post changes	Application 1: 3. New ad group is generated. 4. Data is edited as in file data test Application 2:

	Application 2: 1. Get changes data of campaign C 2. Compare data get changes on application 2 with data post change on application one	1. Can get changes data with number get changes as number data post changes 2. Data get changes and data post changes must be map

3.4.2.Thiết kế kiểm thử hướng dữ liệu

Dữ liệu kiểm thử cho bài kiểm thử "Post changes added new ad group sucessfully" sẽ được tạo ra từ tệp tin excel, tại Sheet 1: "Add new ad group".

Mỗi lần thực thi kiểm thử cho bài kiểm thử trên, dữ liệu kiểm thử sẽ được đọc lên từ tệp tin excel, mỗi lần truyền vào một dòng dữ liệu, sau khi kết thúc kiểm thử cho dữ liệu này sẽ lặp lại với các dữ liệu khác.

Kết quả kiểm thử tương ứng cũng được thể hiện cho từng dữ liệu, để đảm bảo khi bài kiểm thử không đạt yêu cầu (failed), kiểm thử viên có thể điều tra dữ liệu sai một cách nhanh chóng.

Do đặc thù dữ liệu của các trường trong "Ad group" là có chặn trên và chặn dưới nên việc chuẩn bị dữ liệu được thực hiện thủ công. Sau đây là hình minh họa dữ liệu chuẩn bị cho bài kiểm thử thêm mới "ad group"

Data	Campaign name	Ad Group Name	Status	Default Max CPC	Max.CPM bid	CPA bid	Mobile Bid adjustment
1	Campaign #5	Lawyers Australia	Enabled	0.5	4	12	-300
2	Campaign #5	V VS ST Student Schoolarship Australia	Paused	1000	6	65	-100
3	Campaign #5	V VS TS Visitor Schoolarship Australia (Business)	Paused	2	59	1	-90
4	Campaign #5	Ad group A1	Enabled	5	1	1	-89
5	Campaign #5	Ad group A2	Paused	10	2	2	0
6	Campaign #5	Ad group A3	Paused	50	0.25	0.25	
7	Campaign #5	Ad group A4	Enabled	1	2	3	1
8	Campaign #5	Ad group A5	Paused	33	0.6	100	10
9	Campaign #5	Ad group A6	Paused	5.45	10	1000	200
10	Campaign #5	Ad group A7	Enabled	1.99	100	50	299
11	Campaign #5	Ad group A8	Paused	0.9	1000	20	300

Hình 21: Dữ liệu kiểm thử cho bài kiểm thử

3.4.3.Thiết kế các từ khóa cho bài kiểm thử

3.4.3.1.Thiết kế từ khóa mức cao

Với điều kiện tiền đề, để có thể thỏa mãn điều kiện " Đã tải dữ liệu của chiến dịch C", kiểm thử viên tự động phải thiết kế từ bước mở ứng dụng, đăng nhập với tài khoản tương ứng, sau đó kiểm tra xem chiến dịch C có tồn tại chưa. Nếu chưa tồn tại thì thực hiện tải dữ liệu của chiến dịch C về. Các từ khóa nên được chia nhỏ thể có thể tái sử dụng trong những bài kiểm thử khác ở cùng bộ kiểm thử hoặc khác bộ kiểm thử. Tương ứng sẽ có những từ khóa như sau được viết:

Bảng 8: Các từ khóa thiết kế để thực hiện điều kiện tiền đề

#	Tên từ khóa	Miêu tả
1	Start Ad Editor	Khởi động Ad Editor
2	Sign in Ad Editor	Đăng nhập vào ứng dụng
3	Add account	Lựa chọn đăng nhập với tài khoản khác
4	Fill In Email Address And Password	Điền thông tin của email và mật khẩu
5	Wait For All Campaigns Downloaded	Kiểm tra campaign đã được tải về chưa
6	Click Close Button At Download Campaigns Screen	Đóng màn hình sau khi đã tải dữ liệu xong

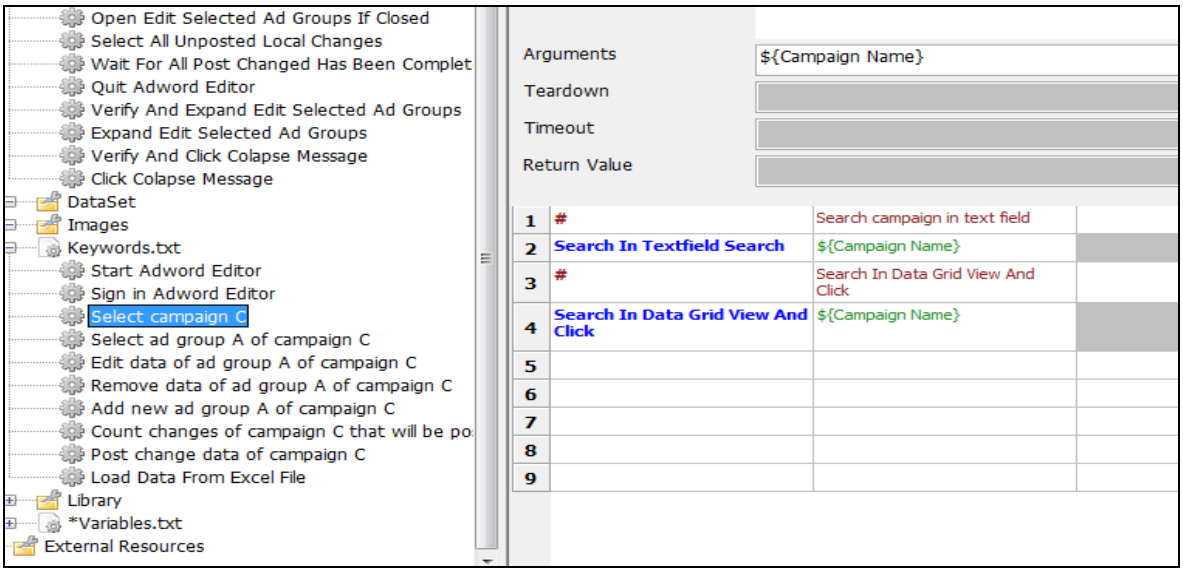
Tiếp theo đến các bước thực hiện bài kiểm thử. Việc xây dựng từ khóa cũng tương tự như việc thiết kế các từ khóa ở điều kiện tiền đề của bài kiểm thử: Chia nhỏ từ khóa đến mức nhỏ hơn để có thể tái sử dụng, tương ứng với việc xây dựng từ khóa mức thấp, rồi sau đó ghép các từ các từ khóa ở mức thấp lại với nhau để được từ khóa ở mức cao hơn. Hay nói cách khác, từ khóa mức cao bao gồm nhiều hành động của người dùng, mỗi hành động được thực hiện bởi một từ khóa mức thấp.

Từ các từ khóa 1,2,3 ở Bảng 10: Các từ khóa xây dựng cho các bước thực hiện bài kiểm thử sẽ xây dựng được một từ khóa mức cao hơn "Select campaign C".

Bảng 9: Xây dựng từ khóa mức cao từ các từ khóa mức thấp

#	Tên keyword	Miêu tả
1	Select Campaigns Tabpage	Chuyển sang của sổ campaign
2	Search In Textfield Search	Tìm kiếm trên trường văn bản (text field)
3	Search In Data Grid View And Click	Tìm kiếm trên bảng dữ liệu và lựa chọn

Giao diện trên Robot framework - IDE của từ khóa "Select campaign C"



Hình 22: Xây dựng từ khóa mức cao

Trong Hình 22, từ khóa "Select campaign C" được xây dựng từ hai từ khóa "Search In Textfield Search" và "Search In Data Grid View And Click" tương ứng với thao tác thực tế của người dùng đó là gồm bước tìm kiếm campaign, sau đó lựa chọn campaign trong kết quả tìm kiếm đang hiển thị.

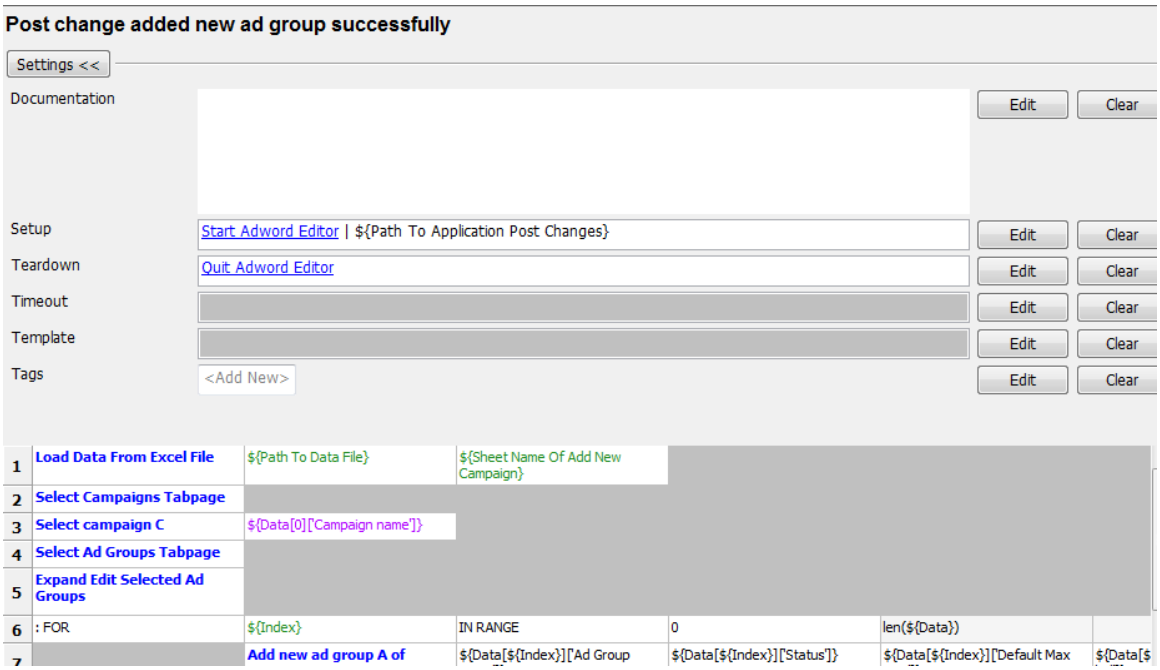
Từ cách làm như vậy, ta có thể xây dựng được các từ khóa mức cao cho các bước thực hiện kiểm thử bài kiểm thử "Thêm mới Ad group" như sau.

Bảng 10: Các từ khóa xây dựng cho các bước thực hiện bài kiểm thử

#	Tên keyword	Miêu tả
1	Load Data From Excel File	Thực hiện tải dữ liệu kiểm thử từ file excel
2	Select Campaigns Tabpage	Lựa chọn cửa sổ Campaign

#	Tên keyword	Miêu tả
3	Select campaign C	Lựa chọn campaign
4	Select Ad Groups Tabpage	Lựa chọn cửa sổ Ad groups
5	Add new ad group A of campaign C	Thêm mới dữ liệu
6	Count changes of campaign C that will be posted	Đếm các thay đổi của campaign C được đẩy lên máy chủ (để sau đó so sánh với số lượng dữ liệu được tải về)
7	Post change data of campaign C	Đẩy dữ liệu lên máy chủ
6	Get changes	Thực hiện tải các dữ liệu thay đổi từ máy chủ về ứng dụng
7	Select campaign tabpage	Mở tab Campaign
8	Select campaign C	Lựa chọn campaign tương ứng có thay đổi dữ liệu
9	Select ad group tab page	Mở tab Ad group của campaign có thay đổi dữ liệu
10	Verify Data In Grid View	Thực hiện so sánh dữ liệu cho từng ad group

Dưới đây là hình minh họa của các bước thực hiện bài kiểm thử thể hiện trên Robot framework IDE:



Hình 23: Xây dựng bài kiểm thử từ các từ khóa

Việc xây dựng các bài kiểm thử khác cũng tương tự như miêu tả cho bài kiểm thử ở Hình 23.

Các từ khóa được xây dựng ra cần được bố trí hợp lý để dễ tái sử dụng trong các bài kiểm thử khác. Việc bố trí các từ khóa này sẽ được giới thiệu ở phần sau.

3.4.3.2.Thiết kế từ khóa mức thấp

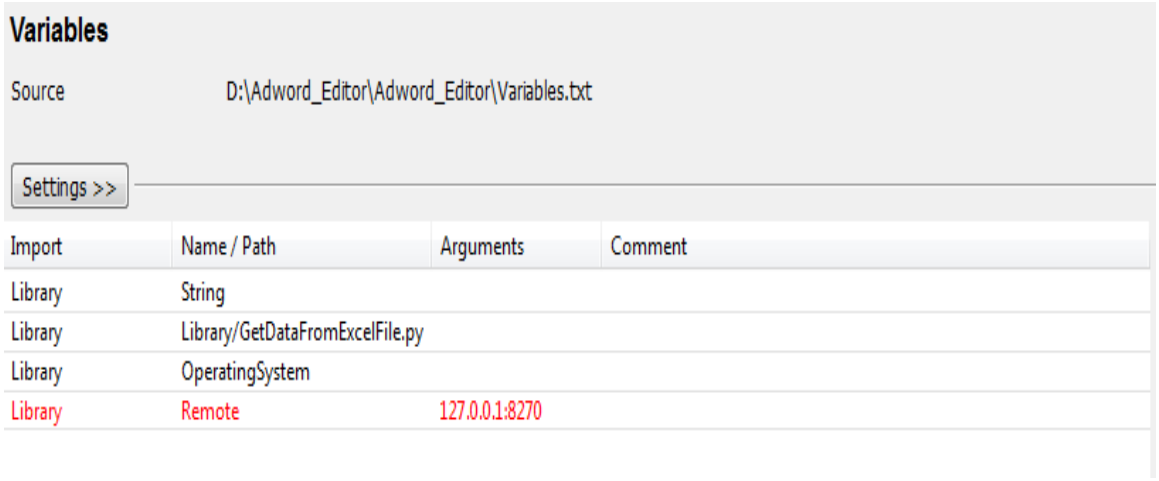
Dựa vào hành động (action) của người dùng ở trong bài kiểm thử mà kiểm thử viên tự động sẽ thiết kế ra những từ khóa dựa trên những hành động này. Những từ khóa có thể được dùng nhiều lần ở trong mọi bộ kiểm thử (test suites) thì sẽ được đưa vào thư viện để dùng chung. Còn những từ khóa chỉ dùng ở trong một bộ kiểm thử thì được viết riêng cho bài kiểm thử đó mà không cần đưa vào thư viện. Ở đây chúng tôi chỉ đưa từ khóa đọc dữ liệu từ tệp tin (GetDataFromExcelFile) thành thư viện. Những từ khóa thể hiện riêng nghiệp vụ của chức năng cần kiểm thử sẽ được viết trong mục các từ khóa nghiệp vụ (Business keywords)

Danh sách từ khóa đã được đưa thành thư viện như sau:

Bảng 11: Danh sách các từ khóa trong thư viện

#	Tên keyword	Miêu tả
1	GetDataFromExcelFile	Thực hiện đọc dữ liệu trong tệp tin

Để sử dụng những từ khóa trong thư viện này, chỉ cần có thao tác thêm thư viện vào bộ kiểm thử cần sử dụng đến nó như Hình 24.



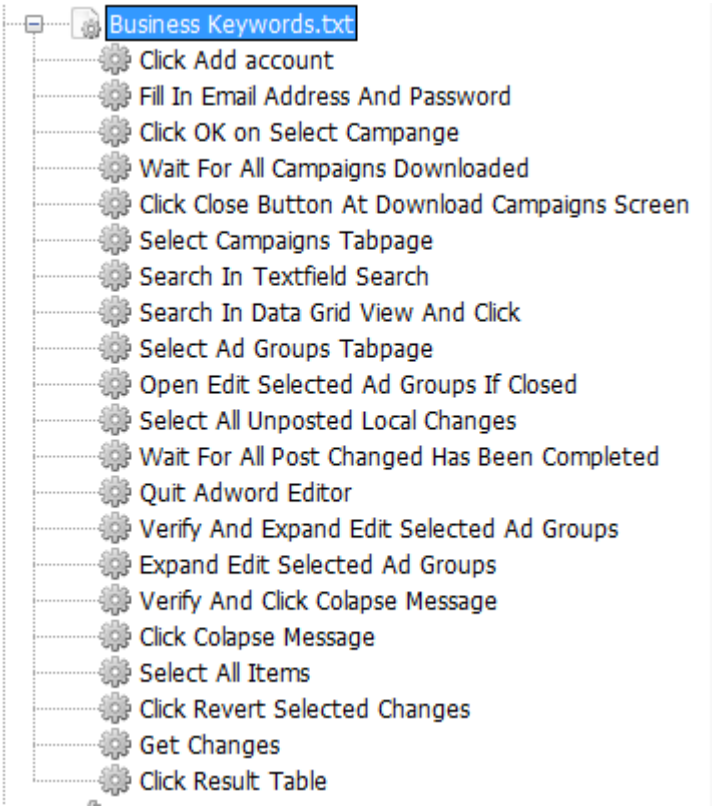
Hình 24: Thêm danh sách từ khóa vào tài nguyên

Danh sách các từ khóa nghiệp vụ được thể hiện ở Bảng 12.

#	Tên keyword	Miêu tả
1	Add account	Lựa chọn đăng nhập tài khoản khác
2	Fill In Email Address And Password	Điền thông tin của email và mật khẩu
3	Click OK on Select Campange	Chọn nút (button) OK khi lựa chọn Campaign
4	Wait For All Campaigns Downloaded	Kiểm tra campaign đã được tải về chưa
5	Click Close Button At Download Campaigns Screen	Đóng màn hình sau khi đã tải dữ liệu xong
6	Select Campaigns Tabpagec	Chuyển sang tab campaign
7	Search In Textfield Search	Tìm kiếm trên trường văn bản (text

#	Tên keyword	Miêu tả
		field)
8	Search In Data Grid View And Click	Tìm kiếm trên bảng dữ liệu và lựa chọn
9	Select Ad Groups Tabpage	Lựa chọn cửa sổ Ad groups
10	Open Edit Selected Ad Groups If Closed	Mở cửa sổ sửa ad group nếu cửa sổ này đang đóng
11	Select All Unposted Local Changes	Lựa chọn tất cả các thay đổi ở máy mà chưa được đẩy lên máy chủ
12	Wait For All Post Changed Has Been Completed	Đợi cho đến khi tất cả các thay đổi đã được đẩy lên máy chủ
13	Quit Ads Editor	Đóng ứng dụng
14	Verify and Expand Edit Selected Ad Groups	Kiểm tra xem cửa sổ sửa dữ liệu của Ad group có mở không
15	Expand Edit Selected Ad Groups	Mở rộng cửa sổ sửa dữ liệu của Ad group
16	Verify and Click Colapse Message	Kiểm tra xem có đang hiển thị các cửa sổ thông báo lỗi, cảnh cáo
17	Click collapse message	Thu nhỏ cửa sổ thông báo lỗi, cảnh cáo
18	Select all items	Lựa chọn tất cả các dữ liệu thay đổi
19	Click revert selected changes	Hủy tất cả các thay đổi
20	Click result table	Click vào kết quả hiển thị trên lưới dữ liệu (data gridview)

Bảng 12: Danh sách các từ khóa nghiệp vụ



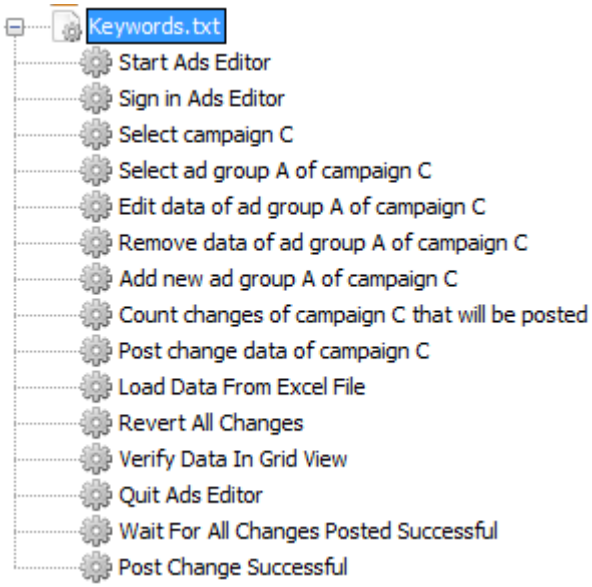
Hình 25: Danh sách các từ khóa nghiệp vụ

Bên cạnh các từ khóa nghiệp vụ, các từ khóa có thể tái sử dụng khi viết bài kiểm thử khác thể hiện ở Bảng 13.

Bảng 13: Danh sách những từ khóa dùng chung

#	Tên keyword	Miêu tả
1	Start Ad Editor	Khởi động Ad Editor
2	Sign in Ads Editor	Đăng nhập
3	Select campaign C	Lựa chọn dữ liệu cần sửa
4	Select ad group A of campaign C	Lựa chọn dữ liệu cần sửa
5	Edit data of ad group A of campaign C	Sửa dữ liệu
6	Remove data of ad group A of campaign C	Xóa dữ liệu
7	Add new ad group A of campaign C	Thêm mới dữ liệu
8	Count changes of campaign C	Đếm số lượng thay đổi

#	Tên keyword	Miêu tả
	that will be posted	
9	Post change data of campaign C	Đẩy dữ liệu lên server
10	Load Data From Excel File	Tải dữ liệu từ một tệp tin excel
11	Revert All Changes	Đưa các dữ liệu có thay đổi ở ứng dụng về giá trị chưa có thay đổi
12	Verify Data In Grid View	Kiểm tra dữ liệu ở bảng dữ liệu
13	Quit Ads Editor	Tắt ứng dụng Ads Editor
14	Wait For All Changes Posted Sucessful	Đợi đến khi tất cả các thay đổi được đẩy lên máy chủ
15	Post Change Sucessful	Thực hiện đẩy dữ liệu thay đổi lên máy chủ thành công



Hình 26: Danh sách các từ khóa dùng chung

Ở đây sẽ miêu tả việc xây dựng một từ khóa dùng chung, hay từ khóa ở mức thấp "Add new ad group A of campaign C". Mã nguồn của từ khóa này như sau:

.Arguments

${Ad Group Name} | ${Status} | ${Default max.CPC bid} | ${Max CPM bid} | ${CPA bid} | ${Mobile bid adjustment}

${Is Add Ad Group Button Exist}= Rn Wait For Element Exist ${Add Ad Group Button} ${Time Out}

Run Keyword If ${Is Add Ad Group Button Exist}==False Fail Add ad group button is not exist
Rn Click Element ${Add Ad Group Button}
 # Set name
Rn Send Keys ${Ad Group Name}
 # set status
Rn Send Keys {Tab}
Run Keyword If '${Status}'=='Paused' Rn Send Keys {Down}
 # Default max. CPC bid
Rn Send Keys {Tab}
Rn Send Keys ${Default max.CPC bid}
 # Max. CPM bid
Rn Send Keys {Tab}
Rn Send Keys {Tab}
Rn Send Keys ${Max CPM bid}
 # CPA bid
Rn Send Keys {Tab}
Rn Send Keys ${CPA bid}
 # Mobile bid adjustment
Rn Send Keys {Tab}
Rn Send Keys {Tab}
Rn Send Keys ${Mobile bid adjustment}

Trong đó, các biến được cài đặt ở trong phần argument là các biến tương ứng với từng trường dữ liệu của ad group mà sẽ được thay đổi dữ liệu thành dữ liệu trong tệp tin dữ liệu kiểm thử của ad group A: *${Ad Group Name}* | *${Status}* | *${Default max.CPC bid}* | *${Max CPM bid}* | *${CPA bid}* | *${Mobile bid adjustment}*

Để thực hiện các bước thêm mới ad group và truyền dữ liệu từ tệp tin kiểm thử tự động vào các trường tương ứng của ad group, ta kiểm tra xem nút (Button) thêm mới dữ liệu có tồn tại không. Nếu có thì thực hiện tiếp các bước sau, nếu không thì thông báo lỗi *"Fail Add ad group button is not exist "*.

Sau khi kiểm tra được nút thêm mới dữ liệu vẫn tồn tại bình thường, cần lựa chọn (click) nút này để thực hiện thêm mới ad group.

Rn Click Element ${Add Ad Group Button}

Sau khi ad group mới được xuất hiện, việc còn lại cần làm đó là sửa đổi dữ liệu của ad group mới như dữ liệu ở trong dữ liệu kiểm thử. Lúc này con trỏ chuột đang ở trường ad group name, nên chỉ cần có bước truyền dữ liệu

 # Set name

Rn Send Keys ${Ad Group Name}

Sau khi truyền dữ liệu cho trường tên, cần sử dụng phím tab để di chuyển đến các trường sau và truyền dữ liệu:

 # set status

Rn Send Keys {Tab}

Run Keyword If '${Status}'=='Paused' Rn Send Keys {Down}

Tương tự như vậy với các trường khác của ad group. Như vậy ta đã tạo xong từ khóa "Add new ad group A of campaign C".

Việc tạo từ khóa bậc cao từ những từ khóa bậc thấp khá đơn giản, giống như việc ghép các bước lại với nhau để tạo thành một chuỗi các hành động hoàn chỉnh. Như từ khóa "Select campaign C" được tạo ra bởi việc ghép hai từ khóa mức thấp "Search In Textfield Search" và "Search In Data Grid View And Click"

Arguments

${Campaign Name}

 # Search campaign in text field

Search In Textfield Search ${Campaign Name}

 # Search In Data Grid View And Click

Search In Data Grid View And Click ${Campaign Name}

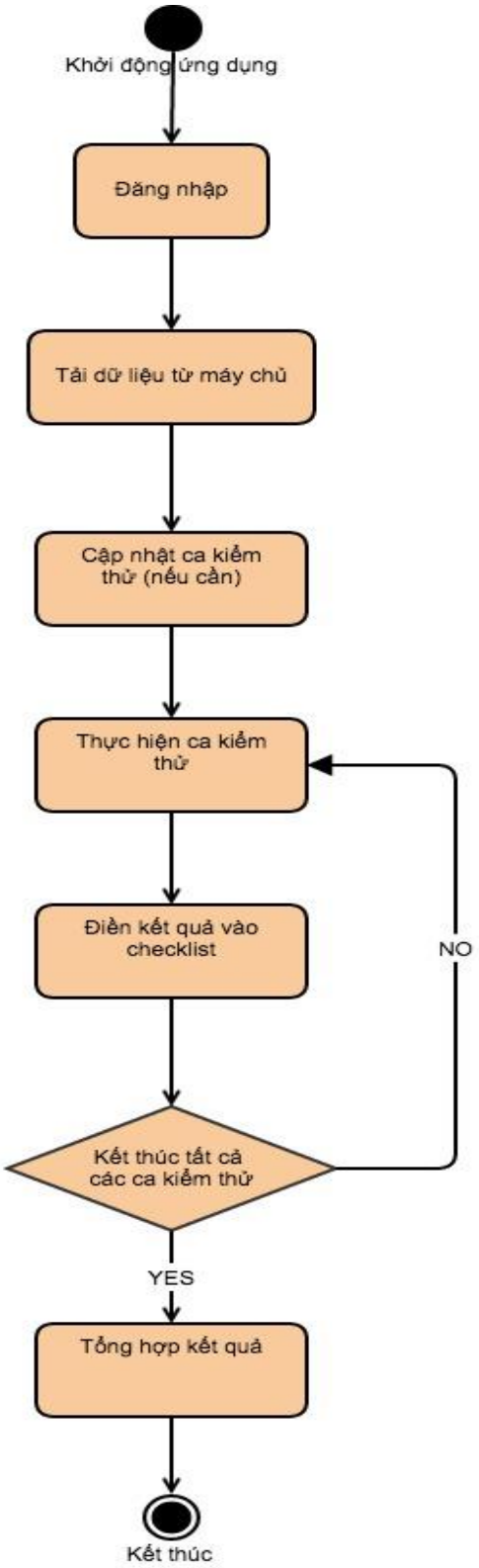
3.5. Thử nghiệm và đánh giá kết quả

Trong quá trình nhập dữ liệu được chỉnh sửa rồi post changes, hệ thống cần phải xác thực dữ liệu đầu vào để đảm tính đúng đắn của dữ liệu. Tất cả các thao tác sẽ được xuất tới tệp tin log. Trong trường hợp có lỗi, kiểm thử viên có thể kiểm tra log để xác định nguyên nhân lỗi và khắc phục sự cố.

Thông tin log hiển thị lỗi trả về tương ứng trong trường hợp bài kiểm thử có lỗi.

3.5.1.Phương pháp đánh giá

Để đưa ra phương pháp đánh giá hiệu quả của phương pháp kiểm thử tự động, cần phải phân tích các bước thực hiện kiểm thử thủ công trong thực tế đối với chức năng Post changes – Get changes (xem Hình 27).



Hình 27: Các bước thực hiện kiểm thử trong thực tế

Trong đó:

- Chuẩn bị môi trường: trước khi kiểm thử, kiểm thử viên phải tiến hành chuẩn bị môi trường kiểm thử theo bài kiểm thử được giao. Bao gồm việc đăng nhập tài khoản, download dữ liệu

- Cập nhật bài kiểm thử test case: Kiểm thử viên phải tiến hành kiểm tra nội dung của bài kiểm thử xem có đầy đủ chưa.

- Thực hiện kiểm thử: Kiểm thử viên tiến hành kiểm thử nội dung của mỗi bài kiểm thử. Sau khi hoàn thành thì thực hiện xác nhận, điền kết quả kiểm thử, thông tin về thời gian thực hiện, phiên bản thực hiện, lỗi liên quan vào tài liệu kết quả kiểm thử (check list hoặc báo cáo). Sau đó sẽ tiến hành kiểm thử bài kiểm thử tiếp theo.

- Tổng hợp kết quả: Sau khi kiểm thử xong tất cả các bài kiểm thử được giao, kiểm thử viên tổng hợp kết quả kiểm thử của mình và báo cáo với người quản lý.

Dựa trên thời gian thực hiện kiểm thử thì có thể tính được chi phí thực hiện cho hoạt động kiểm thử. Vì vậy để đánh giá hiệu quả của việc kiểm thử tự động so với kiểm thử bằng tay, sẽ thực hiện đánh giá thông qua thời gian thực hiện kiểm thử.

Thời gian thực hiện kiểm thử sẽ được tính là thời gian để thực hiện toàn bộ các bước kiểm thử trong Hình 22, trong đó thời gian thực hiện một test case được tính như sau:

$$T = t1 + t2 + t3$$

Với:

- t1: thời gian chuẩn bị dữ liệu kiểm thử
- t2: thời gian nhập dữ liệu kiểm thử
- t3: Thời gian so sánh dữ liệu đã get changes và dữ liệu post changes

Đối với mỗi t1, t2, t3, sẽ thực hiện tính toán xem thời gian thực hiện bằng tay và tự động tương ứng là bao nhiêu. Từ đó, sẽ tính ra thời gian để thực hiện của một bài kiểm thử.

Nếu gọi thời gian thực hiện toàn bộ bài kiểm thử bằng tay là T1, thời gian thực hiện toàn bộ bài kiểm thử bằng Robot framework là T2. Khi đó năng suất đạt được sẽ là:

$$[\text{Năng suất}] = (|T1 - T2|)/T1$$

Dựa vào giá trị [Năng suất] thì có thể xác định được hiệu quả của việc sử dụng kiểm thử tự động với Robot framework

Bên cạnh việc đánh giá kiểm thử tự động và kiểm thử thủ công về mặt thời gian, ta cũng thấy việc kiểm thử tự động và kiểm thử thủ công cho cùng kết quả kiểm thử cho các dữ liệu kiểm thử.

3.5.2.Phân tích kết quả

Thời gian thực hiện bài kiểm thử từ khi chuẩn bị đến khi kết thúc bài kiểm thử "Post changes added new ad group sucessfully" được tính như sau:

Bảng 14: Đánh giá kết quả kiểm thử tự động

Thao tác	Thực hiện bằng tay (phút)	Thực hiện tự động(phút)
Download dữ liệu	5	5
Thao tác khác	2	2
Khởi động Robot framework, khởi động remote library		1
Cập nhật test case	2	2
Thực hiện kiểm thử	9	7
Điền kết quả kiểm thử tới file kiểm thử	1	
Báo cáo kết quả kiểm thử	2	
Tổng thời gian	21	18
Năng suất tăng:		14.29%

Thống kê trên thực hiện trên một máy và cho một bài kiểm thử, do việc "post changes/ get changes" hoàn toàn phụ thuộc vào tình trạng mạng và máy chủ của Google Adword nên kiểm thử tự động chỉ giúp rút ngắn thời gian trong các bước sửa dữ liệu, so sánh dữ liệu sau khi đã "get changes" về và tổng hợp báo cáo kết quả kiểm thử.

Để có thể nâng cao năng suất hơn nữa, cần phải cấp cho kiểm thử viên nhiều máy tính để có thể thực thi các bài kiểm thử song song.

CHƯƠNG 4. KẾT LUẬN VÀ KHUYẾN NGHỊ

Thông qua việc tìm hiểu lý thuyết về kiểm thử và kiểm thử tự động, cũng như áp dụng lý thuyết vào việc xây dựng hệ thống kiểm thử tự động, luận văn đã đạt được những kết quả như sau:

- Tổng hợp lý thuyết về kiểm thử và kiểm thử tự động, lợi ích và thách thức của kiểm thử tự động các phương pháp luận tiếp cận thử động, các bước cần phải làm khi muốn áp dụng tự động hóa kiểm thử.
- Luận văn đã áp dụng lý thuyết để xây dựng kiểm thử tự động cho các chức năng quan trọng của phần mềm, góp phần giảm chi phí, nguồn lực và thời gian thực hiện kiểm thử. Đó là áp dụng khung Robotframework để xây dựng kiểm thử tự động.
- Luận văn đã xây dựng được các từ khóa dùng chung và từ khóa nghiệp vụ sử dụng để kiểm thử tự động cho chức năng của phần mềm Ads Editor. Các từ khóa dùng chung này có thể tái sử dụng khi kiểm thử các chức năng khác của Ads Editor. Nội dung mã nguồn của từng từ khóa sẽ được thể hiện ở phần phụ lục.
- Thông qua luận văn này, chúng tôi nhận thấy rằng kiểm thử tự động là một giải pháp tốt trong việc nâng cao năng suất chất lượng của kiểm thử. Đồng thời cũng giúp nâng cao kỹ năng và kiến thức cho kiểm thử viên. Góp phần giảm thời gian phát triển sản phẩm mà vẫn đảm bảo được chất lượng của phần mềm. Kiểm thử tự động là một thị trường rất tiềm năng không chỉ cho các nhà đầu tư mà còn là một lĩnh vực đang rất khát nhân lực ở Việt Nam, từ đó mang đến nhiều cơ hội cho các kiểm thử viên và các bạn trẻ đang ngồi trên ghế nhà trường.
- Tuy nhiên, để phát huy tốt nhất khả năng của kiểm thử phần mềm tự động, cần phải lựa chọn các chức năng tự động hóa một cách cẩn thận, hiệu quả. Không phải chức năng nào cũng có thể tự động hóa và mang lại hiệu quả. Việc lựa chọn đúng chức năng không những làm giảm thiểu chi phí tự động hóa mà còn nâng cao hiệu quả của tự động kiểm thử.
- Kiểm thử tự động sẽ chỉ đem lại lợi ích qua quá trình sử dụng lâu dài. Do vậy, khi có ý định sử dụng kiểm thử tự động, thì cần phải khuyến khích việc sử

dụng kiểm thử tự động, cũng như cơ chế, chính sách để phát triển kiểm thử tự động. Có như vậy, kiểm thử tự động mới thực sự đem lại hiệu quả to lớn cho tổ chức.

Trong tương lai, có thể tập trung nghiên cứu theo các hướng sau:

- Thứ nhất, Áp dụng tự động hóa cho việc tạo ra các bài kiểm thử, tạo dữ liệu kiểm thử tự động.
- Thứ hai, áp dụng các chức năng khác trong toàn bộ các bài kiểm thử liên quan đến kiểm thử hồi qui bao gồm thay đổi nhiều dữ liệu, thêm mới nhiều dữ liệu, các chức năng khác trong thao tác người dùng của hệ thống.
- Thứ ba, tiếp tục nghiên cứu sửa đổi hệ thống kiểm thử tự động hiện tại để có thể kiểm thử tự động cho các ứng dụng khác. Có thể kế thừa mô hình của hệ thống kiểm thử tự động hiện tại, tuy nhiên cần phải sửa đổi phương pháp thiết lập test case và kịch bản kiểm thử để phù hợp với ứng dụng được hỗ trợ.

TÀI LIỆU THAM KHẢO

[1] Glenford J. Myers, Corey Sandler, Tom Badgett : The Art of Software Testing, 3rd Edition.

[2] Elfriede Dustin (1999), Automated Software Testing, Addison Wesley, 1999, ISBN 0-20143-287-0.

[3] Rex Black, Erik Van Veenendaal, Dorothy Graham, Foundation of software testing, ISTQB Certification.

[4] Pekka Laukkanen: Data-Driven and Keyword-Driven Test Automation Frameworks.

[5] Phạm Ngọc Hùng, Trương Anh Hoàng và Đặng Văn Hưng, Giáo trình kiểm thử phần mềm, tháng 1/2014.

[6] Douglas Hoffman (1999), Test Automation Architectures: Planning for Test Automation, Software Quality Methods, LLC.

[7] Mark Fewster and Dorothy Graham (1999), Software Test Automation: Effective use of test execution tools, ACM Press Books.

[8] Worksoft-Research-Report-2013-Trends-in-Automated-Testing.pdf

[9] http://www.vistacon.vn/en/about-vistacon/in-the-news/333-software-testing-opportunity-to-catch-up-the-global-trend.html

[10] Jerry Zeyu Gao, H.-S. Jacob Tsao and Ye Wu (2003), Testing And Quality Assurance for Component-Based Software, Artech House.

[11] Tuomas Pajunen, Tommi Takala, and Mika Katara, Model-Based Testing with a General Purpose Keyword-Driven Test Automation Framework.

[12] http://en.wikipedia.org/wiki/Test_automation

[13] http://onestoptestinghub.blogspot.com/2008/10/action-based-testing-framework.html

[14] Elfriede Dustin (2003), Effective Software Testing: 50 specific ways to improve your testing, Pearson Education, Inc.

[15] Elfriede Dustin, Implementing Automated Software Testing, Addison Wesley, ISBN 978-0321580511.

[16] C.Bhuvana, K.Munidhanalakshmi, Dr.R.Mahammad Shafi, Table-driven approach for automated testing of web applications (2013)

[17] https://github.com/alans09/robotframework-RanorexLibrary/wiki/Installation

[18] http://robotframework.org/

Phụ Lục 1: Danh sách các từ khóa dùng chung

*** Keywords ***

Start Ads Editor

```
[Arguments]    ${Path To Application}
Run Keyword If    ${Is Debug}==${True}    Rn Start Debug
Rn Run Application    ${Path To Application}
Rn Wait For Element Exist    /form[@title~'^Ads\ Editor\ -\ ']    ${Time Out}
```

Sign in Ads Editor

```
Click Add account
Fill In Email Address And Password    ${Email Address}    ${Password}
Click OK on Select Campange
Wait For All Campaigns Downloaded
Click Close Button At Download Campaigns Screen
Sleep    3s
```

Select campaign C

```
[Arguments]    ${Campaign Name}
#    Search campaign in text field
Search In Textfield Search    ${Campaign Name}
#    Search In Data Grid View And Click
Search In Data Grid View And Click    ${Campaign Name}
```

Select ad group A of campaign C

```
[Arguments]    ${Ad Group Name}
#    Search campaign in text field
Search In Textfield Search    ${Ad Group Name}
```

```
    #    Search In Data Grid View And Click
    Search In Data Grid View And Click    ${Ad Group Name}
```

Edit data of ad group A of campaign C

```
    [Arguments]    ${Old Ad Group Name}    ${Old Status}    ${Ad Group Name}    ${Status}    ${Default max.CPC bid}    ${Max CPM bid}
    ...    ${CPA bid}    ${Mobile bid adjustment}
    # search ad group
    Search In Textfield Search    ${Old Ad Group Name}
    Sleep    4s
    Search In Data Grid View And Click    ${Old Ad Group Name}
    # Set name
    Rn Send Keys    \    {Tab}
    Rn Send Keys    \    ${Ad Group Name}
    # Set Status
    Rn Send Keys    \    {Tab}
    Run Keyword If    '${Old Status}'=='Paused' and '${Status}'=='Enabled'    Rn Send Keys    \    {UP}
    Run Keyword If    '${Old Status}'=='Enabled' and '${Status}'=='Paused'    Rn Send Keys    \    {DOWN}
    # Default max. CPC bid
    Rn Send Keys    \    {Tab}
    Rn Send Keys    \    ${Default max.CPC bid}
    # Max. CPM bid
    Rn Send Keys    \    {Tab}
    Rn Send Keys    \    {Tab}
    Rn Send Keys    \    ${Max CPM bid}
    # CPA bid
```

Rn Send Keys \ {Tab}

Rn Send Keys \ ${CPA bid}

CPA bid

Rn Send Keys \ {Tab}

Rn Send Keys \ {Tab}

Rn Send Keys \ ${Mobile bid adjustment}

Remove data of ad group A of campaign C

[Arguments] ${Campaign Name}

search ad group

Search In Textfield Search ${Campaign Name}

Sleep 2s

${Element Locator}= Replace String /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]//table[@accessiblerole='Table']/row[@accessiblename~'']/cell[@accessiblename='<Campaign name>'] <Campaign name> ${Campaign Name}

${Is Campaign Exist}= Rn Wait For Element Exist ${Element Locator}

Run Keyword If ${Is Campaign Exist}==False Fail Could not found ${Campaign Name} in data grid view

Rn Right Click Element ${Element Locator}

Sleep 2s

Rn Send Keys \ m

Add new ad group A of campaign C

[Arguments] ${Ad Group Name} ${Status} ${Default max.CPC bid} ${Max CPM bid} ${CPA bid} ${Mobile bid adjustment}

Rn Send Keys /form[@title~'^Ads\ Editor\ -\ '] {LControlKey down}{GKey}{LControlKey up}

Sleep 1s

```
# Set name
Rn Send Keys    \    ${Ad Group Name}
# set status
Rn Send Keys    \    {Tab}
Run Keyword If    '${Status}'=='Paused'    Rn Send Keys    \    {Down}
# Default max. CPC bid
Rn Send Keys    \    {Tab}
Rn Send Keys    \    ${Default max.CPC bid}
# Max. CPM bid
Rn Send Keys    \    {Tab}
Rn Send Keys    \    {Tab}
Rn Send Keys    \    ${Max CPM bid}
# CPA bid
Rn Send Keys    \    {Tab}
Rn Send Keys    \    ${CPA bid}
# CPA bid
Rn Send Keys    \    {Tab}
Rn Send Keys    \    {Tab}
Rn Send Keys    \    ${Mobile bid adjustment}
```

Count changes of campaign C that will be posted

```
Select All Unposted Local Changes
${Number Of Items Have Been Changed}=    Evaluate    1
${Current Text}=    Rn Get Text    /form[@title~'^Ads\ Editor\ -\ ']/?/?/statusbar[@accessiblerole='StatusBar']/button[@accessiblename~'^Ad\ groups\ selected:\ ']
${Current Text}=    Replace String Using Regexp    ${Current Text}    ^Ad groups selected: [\\d]* of${SPACE}    ${EMPTY}
```

Comment Run Keyword If '${Current Text}'!='${Number Of Items Have Been Changed}' Fail Current items have been changed are "${Current Text}" while being expected "${Number Of Items Have Been Changed}"

Post change data of campaign C

[Arguments] ${Campaign Name}

Click post change button

${Is Post Changes Button Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/?/?/toolbar[@accessiblerole='ToolBar']/?/?/button[@accessiblename='Post changes'] ${Time Out}

Run Keyword If ${Is Post Changes Button Exist}==False Fail Post change button is not exist

Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/?/?/toolbar[@accessiblerole='ToolBar']/?/?/button[@accessiblename='Post changes']

select campaign to post changes

${Is Campaign Exist}= Rn Wait For Element Exist /form[@title='Post changes to Ads']//container[@accessiblerole='Grouping']/radiobutton[@accessiblename='Campaign "${Campaign Name}"'] ${Time Out}

Run Keyword If ${Is Campaign Exist}==False Fail Your campaign is not exist in post change screen

Rn Click Element /form[@title='Post changes to Ads']//container[@accessiblerole='Grouping']/radiobutton[@accessiblename='Campaign "${Campaign Name}"']

Sleep 3s

Verify Number Of Items Have Been Changed

Comment ${Current Number}= Rn Get Text /form[@title='Post changes to Ads']/?/?/container[@accessiblerole='PropertyPage']/text[9]

Comment ${Number Of Items Have Been Changed}= Evaluate len(${Data}}

Comment Should Be Equal As Strings ${Current Number} ${Number Of Items Have Been Changed}

Click post change button

Rn Click Element /form[@title='Post changes to Ads']/?/?/button[@accessiblename='Post']

Sleep 3s

Input password and click next button

Rn Send Keys \ ${Password}

Rn Click Element /form[@title~'^Sign\ in\ to\ Ads\ accoun']/?/?/button[@accessiblename='Next']

Wait For All Post Changed Has Been Completed

click close button

Rn Click Element /form[@title='Post changes to Ads']/?/?/button[@accessiblename='Close']

Load Data From Excel File

[Arguments] ${Path To Data File} ${Sheet Name}

${Data}= Get Test Data By Sheet Name ${Path To Data File} ${Sheet Name}

Set Test Variable ${Data}

Revert All Changes

Select Campaigns Tabpage

Select All Items

Click Revert Selected Changes

Select Ad Groups Tabpage

Select All Items

Click Revert Selected Changes

Verify Data In Grid View

[Arguments] ${Ad Group Name} ${Default Max CPC} ${CPA Bid} ${Mobile Bid Adjustment}

${Ad Group Name Element}= Replace String /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]//table[@accessiblerole='Table']/row/cell[@accessiblename='<ad group name>'] <ad group name> ${Ad Group Name}

${Is Ad Group Name Exist}= Rn Wait For Element Exist ${Ad Group Name Element} ${Time Out}

Run Keyword If ${Is Ad Group Name Exist}==False Fail Your Ad Group Name: "${Ad Group Name}" is not exist in grid view

${Default Max CPC}= Catenate SEPARATOR= ${Default Max CPC} .00

${Max CPC Element}= Replace String /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]//table[@accessiblerole='Table']/row/cell[@accessiblename='<accessible name>'] <accessible name> ${Default Max CPC}

${Is Max CPC Exist}= Rn Wait For Element Exist ${Max CPC Element} ${Time Out}

Run Keyword If ${Is Max CPC Exist}==False Fail Your Default Max CPC: "${Default Max CPC}" is not exist in grid view

Comment CPA Bid

${CPA Bid}= Catenate SEPARATOR= ${CPA Bid} .00

${CPA BID Element}= Replace String /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]//table[@accessiblerole='Table']/row/cell[@accessiblename='<accessible name>'] <accessible name> ${CPA Bid}

${Is CPA BID Exist}= Rn Wait For Element Exist ${CPA BID Element} ${Time Out}

Run Keyword If ${Is CPA BID Exist}==False Fail Your CPA bid: "${CPA Bid}" is not exist in grid view

${Mobile Exist}= Replace String /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]//table[@accessiblerole='Table']/row/cell[@accessiblename='<accessible name>'] <accessible name> ${Mobile Bid Adjustment}

${Is Mobile Exist}= Rn Wait For Element Exist ${Mobile Exist} ${Time Out}

Comment Run Keyword If ${Is Mobile Exist}==False Fail Your Mobile Bid Adjustment: "${Mobile Bid Adjustment}" is not exist in grid view

Quit Ads Editor

${Is Close Button Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/?/?/button[@accessiblename='Close']

Run Keyword If ${Is Close Button Exist}==True Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/?/?/button[@accessiblename='Close']

Sleep 5s

Wait For All Changes Posted Successful

[Arguments] ${Time Out}=30

${Start Time}= Evaluate time.time() time

${Current Time}= Set Variable

: FOR ${Index} IN RANGE 1 1000

\ ${Current Time}= Evaluate time.time() time

\ ${Current Time}= Evaluate str(int(round(${Current Time}-${Start Time})))

\ Log Wait for all changes are posted in ${Time Out}, current is: ${Current Time}

\ Run Keyword If ${Current Time}> ${Time Out} Exit For Loop ELSE Sleep 2s

Post Change Successful

[Arguments] ${Campaign Name} ${Ad Group Name} ${Status} ${Default Max CPC} ${Max. CPM bid} ${CPA bid}

... ${Mobile Bid adjustment} ${Time Out}

Select Campaigns Tabpage

Select campaign C ${Campaign Name}

Select Ad Groups Tabpage

Expand Edit Selected Ad Groups

Add new ad group A of campaign C ${Ad Group Name} ${Status} ${Default Max CPC} ${Max. CPM bid} ${CPA bid} ${Mobile Bid adjustment}

Count changes of campaign C that will be posted

Post change data of campaign C ${Campaign Name}

Quit Ads Editor

Wait For All Changes Posted Successful ${Time Out}

Start Ads Editor ${Path To Application Get Changes}

Get Changes

Select Campaigns Tabpage

Select campaign C ${Campaign Name}

Select Ad Groups Tabpage

Verify Data In Grid View ${Ad Group Name} ${Default Max CPC} ${Max. CPM bid} ${CPA bid}

Phụ Lục 2: Danh sách các từ khóa nghiệp vụ

*** Keywords ***

Click Add account

${v_isAddAccountExist}= Rn Wait For Element Exist /form[@title='Add/manage accounts']/?/?/button[@accessiblename='Add account'] ${Time Out}

Run Keyword If ${v_isAddAccountExist}==False Fail Add account button did not exist

Rn Click Element /form[@title='Add/manage accounts']/?/?/button[@accessiblename='Add account']

Fill In Email Address And Password

[Arguments] ${Email Address} ${Password}

Verify Add new Ads account exist

${Is Add New Ads Account Exist}= Rn Wait For Element Exist /form[@title='Add new Ads account'] ${Time Out}

Run Keyword If ${Is Add New Ads Account Exist}==False Fail Form Add new Ads account is not opened

Verify and input email address

${Is Email Address Exist}= Rn Wait For Element Exist /form[@title='Add new Ads account']/element/text[2] ${Time Out}

Run Keyword If ${Is Email Address Exist}==False Fail Email Address field is not exist

Rn Input Text /form[@title='Add new Ads account']/element/text[2] ${Email Address}

Verify and input password

Rn Send Keys \ {TAB}

Sleep 0.2

Rn Send Keys \ ${Password}

Verify and click next button

${Is Next Button Exist}= Rn Wait For Element Exist /form[@title='Add new Ads account']/?/?/button[@accessiblename='Next'] ${Time Out}

Run Keyword If ${Is Next Button Exist}==False Fail Next button is not exist in Add new account form

Rn Click Element /form[@title='Add new Ads account']/?/?/button[@accessiblename='Next']

Click OK on Select Campaign

${Is OK Button Exist}= Rn Wait For Element Exist /form[@title='Add new Ads account']/?/?/button[@accessiblename='OK'] ${Time Out}

Run Keyword If ${Is OK Button Exist}==False Fail OK button is not exist in Add new Ads account

Rn Click Element /form[@title='Add new Ads account']/?/?/button[@accessiblename='OK']

Wait For All Campaigns Downloaded

Wait for all campaigns downloaded

${Is Open A Different Account Exist}= Rn Wait For Element Exist /form[@title='Download complete']/?/?/button[@accessiblename~'^Open\ a\ different\ account\.'] ${Time Out}

Run Keyword If ${Is Open A Different Account Exist}==False Fail Could not find Open a different account... button

${Are All Campainge Downloaded}= Run Keyword And Ignore Error Rn Wait For Element Attribute /form[@title='Download complete']/?/?/button[@accessiblename~'^Open\ a\ different\ account\.'] Enabled ${True} 1800000

Run Keyword If '${Are All Campainge Downloaded[0]}'=='FAIL' Fail All campaigns have not been downloaded after 30 minutes

Click Close Button At Download Campaigns Screen

${Is Close Button Exist}= Rn Wait For Element Exist /form[@title='Download complete']/?/?/button[@accessiblename='Close'] ${Time Out}

Run Keyword If ${Is Close Button Exist}==False Fail Close button at download campaigns screen is not exist

Rn Click Element /form[@title='Download complete']/?/?/button[@accessiblename='Close']

Select Campaigns Tabpage

Click Result Table

Rn Send Keys /form[@title~'^Ads\ Editor\ -\ '] {LControlKey down}{D6}{LControlKey up}

Sleep 1s

Search In Textfield Search

[Arguments] ${Campaign Name}

${Is Search Text Field Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -']/text[@accessiblename~'^Search\ within\ this'] ${Time Out}

Run Keyword If ${Is Search Text Field Exist}==False Fail Could not find Search text field in Ads editor form

Rn Input Text /form[@title~'^Ads\ Editor\ -']/text[@accessiblename~'^Search\ within\ this'] ${Campaign Name}

Sleep 1s

Search In Data Grid View And Click

[Arguments] ${Campaign Name}

${Element Locator}= Replace String /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]//table[@accessiblerole='Table']/row[@accessiblename~'']/cell

[@accessiblename='<Campaign name>'] <Campaign name> ${Campaign Name}

${Is Campaign Exist}= Rn Wait For Element Exist ${Element Locator}

Run Keyword If ${Is Campaign Exist}==False Fail Could not found your campaign: "${Campaign Name}" in data grid view

Rn Double Click Element ${Element Locator}

Sleep 3s

Select Ad Groups Tabpage

Click Result Table

Rn Send Keys /form[@title~'^Ads\ Editor\ -\ '] {LControlKey down}{D3}{LControlKey up}

Sleep 1s

Open Edit Selected Ad Groups If Closed

${Is Opened}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/container[@accessiblename='Edit selected ad groups']/text[3]

Run Keyword If ${Is Opened}==False Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/text[@accessiblename='Edit selected ad groups']

Select All Unposted Local Changes

${Is Combobox Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/element/combobox[2]

Run Keyword If ${Is Combobox Exist}==False Fail Combobox View is not exist

Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/element/combobox[2]

Sleep 1s

Rn Send Keys \ {Down}{return}

Sleep 1s

Wait For All Post Changed Has Been Completed

Wait for Post change has been completed in 1 minute

${Is Completed}= Rn Wait For Element Exist /form[@title='Post changes to Ads']/?/?/container[@accessiblerole='PropertyPage']/text[@accessiblename='Post completed'] 60000

Run Keyword If ${Is Completed}==False Fail Post change has not been completed

Sleep 2s

Verify And Expand Edit Selected Ad Groups

${Path To Image}= Join Path ${CURDIR}/Images Edit selected ad groups.png

${Is Contain Image}= Rn Find Element Based On Image /form[@title~'^Ads\ Editor\ -\ '] ${Path To Image}

Run Keyword If ${Is Contain Image}==True Expand Edit Selected Ad Groups

Expand Edit Selected Ad Groups

${Is Edit Selected Ad Groups Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/container[@accessiblename='Edit selected ad groups']/text[3] 2000

${Is Edit Selected Ad Groups Exist1}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/text[@accessiblename='Edit selected ad groups'] 2000

Run Keyword If ${Is Edit Selected Ad Groups Exist}==False and ${Is Edit Selected Ad Groups Exist1}==True Rn Click Element /form[@title~'^Ads\

Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/text[@accessiblename='Edit selected ad groups']

Sleep 1s

Verify And Click Colapse Message

${Path To Image}= Join Path ${CURDIR}/Images Message.png

${Is Contain Image}= Rn Find Element Based On Image /form[@title~'^Ads\ Editor\ -\ '] ${Path To Image}

Run Keyword If ${Is Contain Image}==True Click Colapse Message

Click Colapse Message

${Is Colapse Message Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/button[@accessiblerole='PushButton']

Run Keyword If ${Is Colapse Message Exist}==True Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/?/?/button[@accessiblerole='PushButton']

Select All Items

Click Result Table

Rn Send Keys \ {LControlKey down}{Akey}{LControlKey up}

Sleep 1s

Click Revert Selected Changes

Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/?/?/container[@accessiblerole='Grouping']/container/toolbar[@accessiblerole='ToolBar']/?/?/button[@accessiblename='Revert selected changes']

Sleep 2s

Get Changes

${Is Get Change Button Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]/container[@accessiblerole='Grouping']/?/?/table[@accessiblerole='Table'] ${Time Out}

Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]/container[@accessiblerole='Grouping']/?/?/table[@accessiblerole='Table']

Rn Send Keys \ {LControlKey down}{Rkey}{LControlKey up}

Sleep 2s

Input password and click next button

Rn Send Keys \ ${Password}

Rn Click Element /form[@title~'^Sign\ in\ to\ Ads\ accoun']/?/?/button[@accessiblename='Next']

select All Campaigns

Click OK button

${OK Button Exist}= Rn Wait For Element Exist /form[@title='Update Ads account(s)']/?/?/button[@accessiblename='OK'] ${Time Out}

Run Keyword If ${OK Button Exist}==False Fail OK button is not exist in Update Ads account(s)

Rn Click Element /form[@title='Update Ads account(s)']/?/?/button[@accessiblename='OK']

Wait for download completed in 1 minute

${Is Download Completed Exist}= Rn Wait For Element Exist /form[@title='Download complete']/?/?/text[@accessiblename='Download complete'] 60000

Run Keyword If ${Is Download Completed Exist}==False Fail Download Completed is not exist in Download Complete form

Click Close Button

${Is Close Button Exist}= Rn Wait For Element Exist /form[@title='Download complete']/?/?/button[@accessiblename='Close'] ${Time Out}

Run Keyword If ${Is Close Button Exist}==False Fail Close button is not exist in Download Complete form

Rn Click Element /form[@title='Download complete']/?/?/button[@accessiblename='Close']

Click Result Table

${Is Result Table Exist}= Rn Wait For Element Exist /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]/container[@accessiblerole='Grouping']/?/?/table[@accessiblerole='Table'] ${Time Out}

Sleep 2s

Run Keyword If ${Is Result Table Exist}==True Rn Click Element /form[@title~'^Ads\ Editor\ -\ ']/element/element[1]/container[@accessiblerole='Grouping']/?/?/table[@accessiblerole='Table']