

Lời cảm ơn

Lời đầu tiên, chúng em xin chân thành cảm ơn quý thầy cô trong khoa Toán Tin Học Trường Đại Học Khoa Học Tự Nhiên Thành Phố Hồ Chí Minh đã tận tình giảng dạy chúng em suốt 4 năm học vừa qua. Chính nhờ những kiến thức ấy, chúng em mới có đủ nền tảng để thực hiện đề tài này. Chúc các thầy cô thật nhiều sức khỏe và thành công.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến Thạc sĩ Nguyễn Đạt Thông, người Thầy đã tận tình chỉ bảo, hướng dẫn chúng em nhiệt tình xuyên suốt quá trình làm luận văn, để chúng em hoàn thành tốt được luận văn này.

Một lần nữa xin chân thành cảm ơn tất cả mọi người đã quan tâm đến đề tài này.

Mục lục

Lời cảm ơn	ii
Mục lục	iii
Danh sách bảng	vi
Danh sách hình vẽ	vii
1 Tổng Quan Về Đề Tài	1
1.1 Đặt vấn đề	1
1.2 Đối tượng và phạm vi nghiên cứu	1
1.3 Tầm quan trọng của đề tài	2
1.4 Mục tiêu của đề tài	2
2 Cơ Sở Lý Thuyết	3
2.1 Tổng quan về kiểm thử tự động	3
2.1.1 Khái niệm kiểm thử tự động	3
2.1.2 Mục đích của kiểm thử tự động	3
2.1.3 Các trường hợp kiểm thử tự động	4
2.1.4 Các loại hình kiểm thử tự động	4
2.2 Các phương pháp kiểm thử tự động	5
2.2.1 Phương pháp Data Driven Testing	5
2.2.1.1 Giới thiệu về data driven testing	5
2.2.1.2 Đặc điểm của data driven testing	6
2.2.2 Phương pháp keyword driven testing	6

2.2.2.1	Giới thiệu về keyword driven testing	6
2.2.2.2	Đặc điểm của keyword driven testing	6
2.2.3	Phương pháp action base testing	7
2.2.3.1	Giới thiệu action base testing	7
2.2.3.2	Đặc điểm của action base testing	7
2.3	Coded UI	8
2.4	White Framework	9
2.5	Các công cụ kiểm thử tự động	11
2.5.1	Công cụ kiểm thử Test Complete	11
2.5.1.1	Giới thiệu về Test Complete	11
2.5.1.2	Đặc điểm của Test Complete	11
2.5.1.3	Giao diện phần mềm	12
2.5.2	Công cụ kiểm thử TestArchitect™	15
2.5.2.1	Giới thiệu về TestArchitect™	15
2.5.2.2	Đặc điểm của TestArchitect™	15
3	Phân Tích Thiết Kế Hệ Thống	18
3.1	Tổng quan về hệ thống	18
3.2	Các Chức Năng Của Hệ Thống	18
3.2.1	Chức Năng Quản Lý	18
3.2.2	Chức Năng Sao Lưu Dữ Liệu	19
3.2.3	Chức Năng Thực Thi	19
3.2.4	Chức Năng Báo Cáo Kết Quả (Report)	19
3.2.5	Chức Năng Tìm Kiếm Các Tiêu Chí (Spy)	19
3.2.6	Chức Năng Hỗ Trợ Cài Đặt	20
3.2.7	Cấu Hình Hệ Thống	20
3.3	Lược Đồ Use Case Và Đặc Tả	21
3.3.1	Sơ Đồ Tổng Quát	21
3.3.1.1	Chức Năng Quản Lý Project	22
3.3.1.2	Chức Năng Quản Lý Data	23

3.3.1.3	Chức Năng Quản Lý Interface	24
3.3.1.4	Chức Năng Quản Lý Test Script	25
3.3.1.5	Chức Năng Thực Thi Chương Trình	26
3.3.1.6	Chức Năng Quản Lý Báo cáo	27
3.4	Giao Diện Của Hệ Thống	28
3.5	Hướng Dẫn Sử Dụng	34
4	Tổng Kết	43

Danh sách bảng

3.1	Project Management	22
3.2	Data Management	23
3.3	Interface Management	24
3.4	Script Management	25
3.5	Report Management	27

Danh sách hình vẽ

2.1	Kiến trúc MSAA	9
2.2	Kiến trúc White framework	10
2.3	Giao diện chính	13
3.1	Kiến trúc hệ thống	20
3.2	Sơ đồ tổng quát	21
3.3	Giao diện hệ thống	28
3.4	Giao diện soạn thảo Data	29
3.5	Giao diện soạn thảo Interface	30
3.6	Giao diện soạn thảo Test Script	31
3.7	Giao diện Report	32
3.8	Giao diện Spy	33
3.9	Kiểm tra tính hợp lệ của Interface	34
3.10	Open Project	35
3.11	Browse for Folder	35
3.12	Create Project	36
3.13	New Project	36
3.14	Giao diện chương trình	37
3.15	Soạn thảo nội dung	38
3.16	Chọn file Script để chạy chương trình	38
3.17	Chọn file Data cho quá trình Test	39
3.18	Chạy chương trình	40

Chương 1

Tổng Quan Về Đề Tài

1.1 Đặt vấn đề

Ngày nay, tự động hóa được ứng dụng ở nhiều lĩnh vực khác nhau trong cuộc sống. Ngành công nghệ thông tin mà cụ thể hơn là ngành công nghệ phần mềm cũng không ngoại lệ. Để tạo ra một sản phẩm công nghệ thông tin hay một phần mềm có chất lượng, thì hoạt động kiểm thử phần mềm đóng vai trò rất quan trọng, trong khi đó, hoạt động này lại tiêu tốn nhiều thời gian và công sức trong một dự án. Do vậy, nhu cầu tự động hóa qui trình kiểm thử phần mềm được đặt ra.

Tuy nhiên, một thực tế là chi phí cho các phần mềm kiểm thử tự động thương mại rất cao, đôi lúc làm cho dự án không còn khả năng sinh lời, với tình hình kinh tế hiện nay, các doanh nghiệp gia công phần mềm đang phải gồng mình cung cấp các gói dịch vụ giá thấp nhưng có chất lượng cao hơn để thu hút khách hàng. Để khắc phục điều này, kiểm thử viên buộc phải sử dụng nhiều phần mềm kiểm thử khác nhau cho một dự án phát triển phần mềm. Nắm bắt được vấn đề đó, chúng tôi quyết định thực hiện đề tài: Xây dựng hệ thống kiểm thử tự động các ứng dụng Windows dựa trên công nghệ .Net nhằm cung cấp một giải pháp toàn diện giải quyết các vấn đề mà công cụ kiểm thử tự động các ứng dụng Desktop.

1.2 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài này là các phương pháp, kĩ thuật đang được sử dụng trong quy trình kiểm thử tự động và các công cụ, phần mềm hỗ trợ quá trình kiểm tra tự động, đặc biệt là các công cụ kiểm thử tự động mã nguồn mở.

Phạm vi nghiên cứu của đề tài này là tập trung vào việc nghiên cứu các công việc thường được thực hiện trong quá trình quản lý các dự án kiểm thử tự động, đặc biệt quá trình quản lý Test Case và Test Script.

1.3 Tầm quan trọng của đề tài

Về thực tiễn: Mong muốn của chúng tôi là một trong công cụ mã nguồn mở hỗ trợ cho các kiểm thử viên trong quá trình kiểm thử phần mềm.

Về ứng dụng: Phần mềm ra đời phát triển nhằm trở thành công cụ hỗ trợ đắc lực cho cho các kiểm thử viên, tiết kiệm chi phí cho các doanh nghiệp và cho các dự án.

Về sử dụng: Với giao diện đơn giản, thân thiện sẽ mang lại cho người dùng thoải mái và tiện lợi khi dùng sản phẩm.

1.4 Mục tiêu của đề tài

Mục tiêu của đề tài là giải quyết các vấn đề phát sinh trong việc sử dụng phần mềm kiểm thử tự động mã nguồn mở, nghĩa là chúng tôi muốn cung cấp một phần mềm kiểm thử tự động ứng dụng Desktop mã nguồn mở hoàn chỉnh về mặt chức năng, tin cậy trong quá trình sử dụng và thân thiện với người sử dụng, có thể sử dụng cho mọi dự án kiểm thử ứng dụng Desktop. Quan trọng hơn cả, chúng tôi mong muốn chung tay, góp sức cùng với các bạn trẻ khác để thúc đẩy sự phát triển của ngành công nghệ phần mềm, đặc biệt là ngành kiểm thử phần mềm ở Việt Nam.

Chương 2

Cơ Sở Lý Thuyết

2.1 Tổng quan về kiểm thử tự động

2.1.1 Khái niệm kiểm thử tự động

Hiện nay có nhiều định nghĩa khác nhau về kiểm thử tự động. Chúng tôi đưa ra một phép đối chiếu nhỏ giữa kiểm thử thủ công và kiểm thử tự động. Điều này sẽ giúp dễ dàng hiểu được kiểm thử tự động là như thế nào.

Kiểm thử thủ công yêu cầu người thực hiện mọi thứ một cách thủ công. Mọi thao tác từ việc thiết kế Test Case đến việc thực thi Test Case, thống kê và báo cáo kết quả kiểm thử được thực hiện hoàn toàn bằng tay.

Chẳng hạn đối với kiểm thử thủ công, nếu bạn muốn kiểm thử chức năng của Calculator, bạn cần phải nhập điều kiện ban đầu, thực thi, quan sát kết quả, so sánh kết quả. Sau đó, so sánh kết quả thực tế với kết quả mong muốn trong Test Case, ghi nhận lại kết quả. Tất cả đều được thực hiện bằng tay.

Trong khi đó, kiểm thử tự động sẽ thực hiện mọi công việc trên cho bạn thông qua một công cụ hoặc phần mềm kiểm thử (testing tool). Các công cụ và phần mềm kiểm thử sẽ giúp kiểm thử viên tự động hóa một số giai đoạn nào đó trong quy trình kiểm thử phần mềm.

2.1.2 Mục đích của kiểm thử tự động

Tự động hóa quá trình kiểm thử phần mềm giúp rút ngắn thời gian phát hành phần mềm mới, giảm công sức thực hiện, giảm sai sót, tăng độ tin cậy, giảm sự nhầm lẫn trong quá trình kiểm thử phần mềm và nâng cao kỹ năng lập trình cho đội ngũ kiểm thử viên.

Ngoài ra, kiểm thử tự động sẽ giúp kiểm thử viên giả lập nhiều tình huống khó có thể thực hiện nếu kiểm thử thủ công như giả lập truy cập cùng lúc 1000 tài khoản vào ứng dụng cùng

một lúc, giúp ích cho quá trình đo lường hiệu năng và hiệu suất của chương trình, đặc biệt là quá trình kiểm tra hiệu năng (performance testing) của ứng dụng.

Do một số giai đoạn đã được tự động hóa, nên nguồn nhân lực sử dụng cho quá trình kiểm thử phần mềm cũng được giảm đi đáng kể. Điều này, làm hạ giá thành sản phẩm phần mềm, nhưng vẫn đảm bảo chất lượng sản phẩm, mang lại lợi nhuận cho công ty hoặc doanh nghiệp sản xuất phần mềm, cũng như tiết kiệm chi phí cho những khách hàng của họ.

2.1.3 Các trường hợp kiểm thử tự động

Qua thực tế cho thấy, việc áp dụng hợp lý kiểm thử tự động sẽ mang lại thành công cho hoạt động kiểm thử phần mềm. Tuy nhiên, không phải mọi dự án, mọi phần mềm, đều kiểm thử tự động. Kiểm thử tự động thường được áp dụng trong một số tình huống sau:

- Không đủ tài nguyên: khi số lượng các Test Case quá nhiều, mà các kiểm thử viên không thể hoàn tất bằng tay trong một thời gian cụ thể nào đó.
- Kiểm thử hồi quy: thực tế cho thấy, mỗi phiên bản phần mềm bao gồm những tính năng mới hoặc tính năng cũ được sửa lỗi hay nâng cấp. Việc bổ sung hoặc sửa lỗi mà chương trình ở các phiên bản mới có thể làm cho những tính năng khác đã kiểm tra tốt chạy sai, mặc dù phần mã chương trình của nó không hề bị chỉnh sửa. Để khắc phục điều này, đối với từng phiên bản, kiểm thử viên không chỉ kiểm tra các chức năng mới hoặc chức năng sửa đổi, mà phải kiểm tra lại tất cả những tính năng đã kiểm tra tốt trước đó. Điều này khó thực hiện trong một thời gian ngắn nếu kiểm tra bằng tay.
- Kiểm tra sự vận hành của phần mềm trong môi trường đặc biệt: Đây là phép kiểm tra nhằm đánh giá xem sự vận hành của phần mềm có thỏa mãn yêu cầu đặt ra hay không. Thông qua đó, kiểm thử viên có thể xác định được các yếu tố về phần cứng có ảnh hưởng đến khả năng vận hành của phần mềm hay không.

Kiểm thử bằng tay đối với các tình huống trên cực kỳ khó, thậm chí không thể thực hiện được.

2.1.4 Các loại hình kiểm thử tự động

Hiện nay kiểm thử tự động được phân loại theo nhiều tiêu chí khác nhau. Nhưng nhìn chung, có một số loại hình kiểm thử sau được sử dụng phổ biến:

- Kiểm thử chức năng (functionality testing): Còn được gọi là kiểm thử hộp đen, dùng để phát hiện những lỗi kỹ thuật làm cho chức năng của ứng dụng hoạt động sai hoặc

không ổn định dựa trên các hành vi của chúng. Quá trình kiểm thử chức năng thường dựa trên các đặc tả kỹ thuật của dự án hoặc các yêu cầu từ phía khách hàng.

- Kiểm thử hồi quy (regression testing): Nhằm đảm bảo phiên bản mới thực hiện tốt chức năng như phiên bản cũ và sự thay đổi không gây ra lỗi mới trên những chức năng vốn đã làm việc tốt. Kiểm thử hồi quy tốn nhiều thời gian và công sức nhất, nhưng “không được phép” bỏ qua, vì nó giúp kiểm thử viên ngăn chặn việc phát sinh những lỗi nghiêm trọng tưởng chừng không có hoặc đã được kiểm tra và sửa chữa rồi.
- Kiểm thử hiệu năng (performance testing): Giúp chúng ta đoán trước được những lỗi có thể xảy ra khi triển khai ứng dụng vào trong môi trường thực tế nhiều người dùng. Bên cạnh đó, nó còn giúp chúng ta tìm ra hiệu năng thực thi tối đa của ứng dụng và tìm ra nơi cần cải tiến cho ứng dụng.
- Kiểm thử giao diện (interface testing): Giúp kiểm thử viên phát hiện ra những yếu tố (phần cứng hoặc phần mềm) ảnh hưởng đến ứng dụng, làm giao diện ứng dụng sai lệch so với lúc thiết kế, khi cài đặt ứng dụng trong môi trường thực tế. Hơn nữa, nó còn giúp tìm ra những thành phần giao diện có thể hoạt động sai khi thực thi trong những môi trường đó.
- Kiểm thử bảo mật (security testing): Giúp kiểm thử đánh giá khả năng bảo mật của ứng dụng trước các cuộc tấn công hệ thống như DoS (Denial of Service), Cross Site Scripting, v.v... Đồng thời nó cũng giúp ta tìm ra những lỗ hổng rò rỉ có thể xảy ra khi ứng dụng bị tấn công.

2.2 Các phương pháp kiểm thử tự động

2.2.1 Phương pháp Data Driven Testing

2.2.1.1 Giới thiệu về data driven testing

Các kịch bản kiểm thử đơn giản thường nhúng dữ liệu kiểm thử vào trong chính mã kịch bản. Điều này dẫn đến một vấn đề, đó là khi dữ liệu thay đổi sẽ dẫn đến mã kịch bản phải thay đổi và cập nhật lại toàn bộ. Đối với mã kịch bản đơn giản hay quy mô nhỏ thì điều này không ảnh hưởng lớn, nhưng nếu mã kịch bản với quy mô lớn và không có cấu trúc thì việc cập nhật lại sẽ mất rất nhiều thời gian. Ngoài ra, nó sẽ gây khó khăn trong việc tìm kiếm và hiệu chỉnh dữ liệu sau này.

Một vấn đề nữa, là đối với người viết ra kịch bản thì việc sửa đổi dữ liệu không quá phức tạp đối với họ, nhưng đối với người sử dụng mã kịch bản, khi chỉnh sửa có thể gặp nhiều

khó khăn. Như vậy, rõ ràng nếu cùng một mã kịch bản mà chúng ta kiểm thử với nhiều bộ dữ liệu thì sẽ mất rất nhiều thời gian, công sức và chi phí.

Với những vấn đề trên, việc nhúng dữ liệu trực tiếp vào mã kịch bản là hoàn toàn không khả thi trong các mô hình kiểm thử. Một phương pháp tiếp cận tốt hơn, là chúng ta sẽ xây dựng riêng một mã kịch bản và sau đó đọc dữ liệu kiểm thử được lưu trữ bên ngoài vào trong mã kịch bản này. Phương pháp này gọi là data driven testing (kiểm thử hướng dữ liệu).

2.2.1.2 Đặc điểm của data driven testing

Phương pháp data driven testing có nhiều đặc điểm, nhưng đáng chú ý nhất là 4 đặc điểm sau đây:

- Giúp kiểm thử viên chạy được nhiều bộ Test Case một cách nhanh chóng, tiết kiệm thời gian, chi phí so với cách tiếp cận truyền thống.
- Công việc sửa đổi, kiểm tra và bổ sung dữ liệu trở nên dễ dàng mà không đòi hỏi bất cứ kĩ năng lập trình nào.
- Giúp kiểm thử viên tạo ra các bộ dữ liệu kiểm thử (Test Data) một cách nhanh chóng ngay cả khi chưa viết kịch bản kiểm thử.
- Phương pháp này dễ dàng áp dụng cho các mô hình kiểm thử tự động quy mô lớn.

Tuy nhiên, hạn chế lớn nhất của phương pháp data driven testing là đa số các Test Case thì tương tự nhau. Mặt khác, nếu thay đổi cấu trúc tổ chức của tập tin chứa dữ liệu kiểm thử thì đồng nghĩa với việc phải cập nhật lại hoàn toàn Test Script.

2.2.2 Phương pháp keyword driven testing

2.2.2.1 Giới thiệu về keyword driven testing

Một giải pháp đưa ra bởi Fewster và Graham (1999) để khắc phục giới hạn của phương pháp data driven testing, là phương pháp keyword driven testing. Theo phương pháp này, bạn không chỉ lấy dữ liệu kiểm thử từ các tập tin bên ngoài, mà còn lấy thêm những chỉ dẫn cho biết dữ liệu đó dùng để làm gì? Những chỉ dẫn đó được gọi là từ khóa (keyword). Đây chính là ý tưởng của phương pháp keyword driven testing.

2.2.2.2 Đặc điểm của keyword driven testing

Do được phát triển từ phương pháp data driven testing nên phương pháp keyword driven testing thừa hưởng được mọi đặc điểm ưu tú phương pháp data driven testing. Ngoài ra, nó còn có thêm một số đặc điểm khác:

- Cho phép xây dựng nhiều Test Case một cách linh động.
- Không làm thay đổi Test Script khi cấu trúc tổ chức của tập tin chứa dữ liệu kiểm thử bị thay đổi.

Tuy nhiên, khó khăn của phương pháp keyword driven testing là các Test Case tạo ra sẽ dài hơn và phức tạp hơn so với phương pháp data driven testing. Như vậy, phương pháp này sẽ đòi hỏi framework phức tạp hơn và dài hơn so với phương pháp data driven testing.

2.2.3 Phương pháp action base testing

2.2.3.1 Giới thiệu action base testing

Theo phương pháp keyword driven testing, khi chúng ta triển khai quá trình kiểm thử tự động trên những ứng dụng lớn, bạn phải tạo ra nhiều từ khóa khác nhau, mỗi từ khóa tương ứng với mỗi tác vụ khác nhau trong Test Script. Vấn đề này, gây khó khăn cho công việc quản lý, tìm kiếm nếu không được tổ chức hợp lý. Phương pháp action base testing ra đời, để giải quyết tình trạng trên. Nó cho phép kiểm thử tạo ra nhiều action khác nhau, mà mỗi action lại chứa nhiều từ khóa trong đó.

Mô hình Action Base Testing (ABT): Gồm ba thành phần chính riêng biệt là Interface, Data, Script. Interface để định nghĩa các đối tượng kiểm thử, Data lưu trữ các bộ dữ liệu và Script là nơi khai báo các Test Case. Mỗi phần đều nằm tách biệt nên khi chỉnh sửa một file bất kì sẽ không ảnh hưởng đến toàn bộ hệ thống, đây cũng chính là ưu điểm của phương pháp Action Base Testing so với những phương pháp khác. Chúng tôi cũng nắm bắt được vấn đề đó nên đã phát triển phần mềm theo phương pháp ABT này. Ngoài ra còn nhiều phần mềm kiểm thử nổi tiếng nổi tiếng được vận hành theo phương pháp này có thể kể đến đó là TestAchitect của hãng kiểm thử nổi tiếng LogiGear

2.2.3.2 Đặc điểm của action base testing

Dưới đây là những đặc điểm chính mà phương pháp action base testing đem lại cho quá trình kiểm thử tự động:

- Phương pháp action base testing giúp tiết kiệm chi phí kiểm thử, thực thi và bảo trì. Đồng thời, nó còn giúp tìm lỗi nhanh hơn từ đó được chi phí sửa lỗi.
- Phương pháp action base testing giúp kiểm thử viên tìm được nhiều lỗi hơn. Đồng thời, nó còn kiểm tra kết quả rất hiệu quả, làm giảm đi khối lượng công việc mà các kỹ sư kiểm thử phải làm.

- Phương pháp action base testing thừa hưởng tất cả những ưu điểm của phương pháp kiểm thử hướng từ khóa
- Phương pháp action base testing là một cải tiến lớn, dễ bảo trì, tái sử dụng và khả năng quản lý cao.

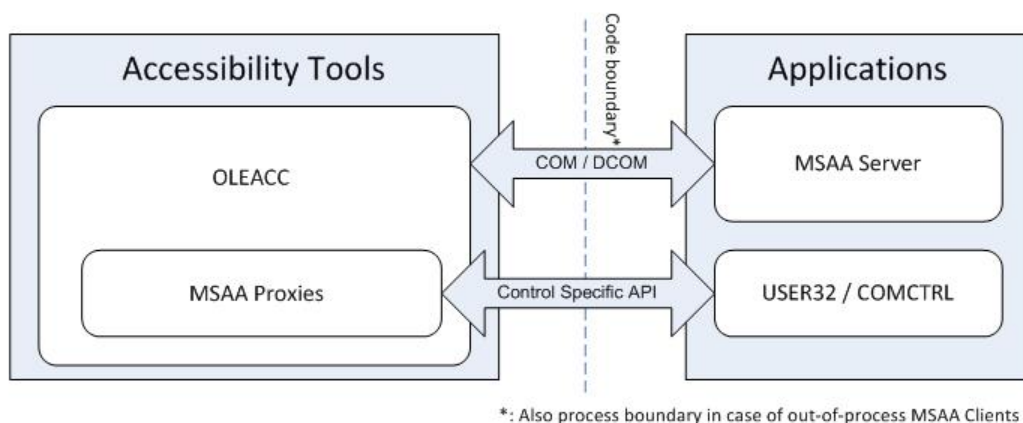
2.3 Coded UI

Microsoft Active Accessibility (MSAA) một nền tảng cho các ứng dụng tự động của Microsoft, trong đó có các framework như White, Cutter... phát triển dựa trên công nghệ này. MSSA là một ứng dụng giao diện lập trình (API) cho người dùng tiếp cận, đã được giới thiệu như là một nền tảng tiện ích cho Microsoft Windows 95 vào năm 1997. MSSA được thiết kế để hỗ trợ sản phẩm Assistive Technology (AT) tương tác tiêu chuẩn và tùy chỉnh với giao diện người dùng thành phần của ứng dụng (hoặc hệ điều hành), cũng như để truy cập, xác định và thao tác với các ứng dụng. Chương trình AT làm việc với MSSA kích hoạt các ứng dụng để cung cấp truy cập tốt hơn cho những người có những khó khăn, suy yếu và khuyết tật về thể chất hoặc nhận thức. Một số ứng dụng của chương trình AT là cho phép người dùng có tầm nhìn hạn chế hoặc khuyết tật về mắt có thể thao tác mới máy tính thông qua màn hình máy tính bằng thao tác dùng tay rà lên màn hình để xác định được nội dung. Ngoài ra MSSA cũng có thể được sử dụng cho các công cụ kiểm thử tự động và các ứng dụng đào tạo dựa trên máy tính.

Mục tiêu:

Các yếu tố thúc đẩy đằng sau sự phát triển của MSAA là cho phép một cơ chế giao tiếp có sẵn và liền mạch của hệ điều hành cơ bản hoặc các ứng dụng và các sản phẩm công nghệ hỗ trợ.

Mục tiêu của chương trình MSAA là cho phép Windows Controls thể hiện những thông tin cơ bản chẳng hạn như tên, vị trí trên màn hình hoặc loại Control và thông tin trạng thái như khả năng hiển thị, kích hoạt hoặc lựa chọn.



Hình 2.1: Kiến trúc MSAA

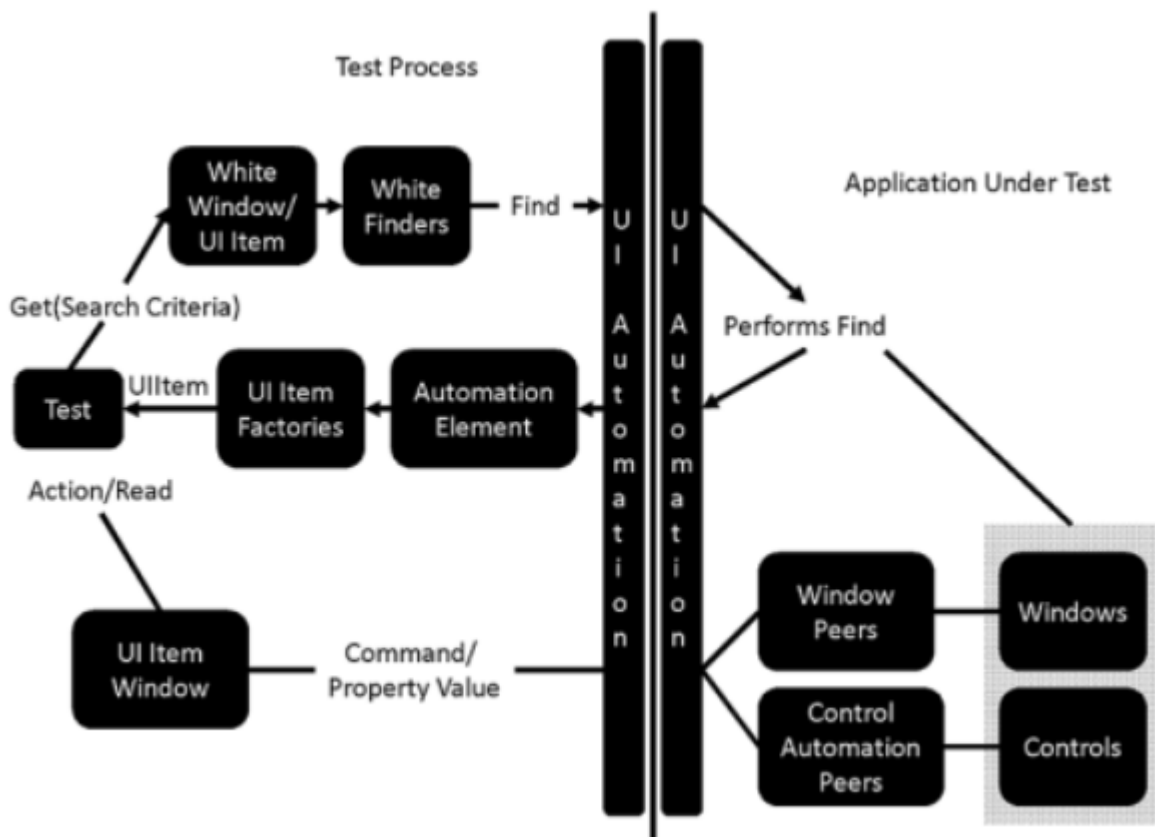
Thừa hưởng những nền tảng của MSAA thì lần lượt các hệ thống tự động càng phát triển hơn và ứng dụng nhiều trong công nghệ Microsoft. Có thể kể tới là Coded UI.

Coded UI (Code User Interface): Thực hiện hành động điều khiển giao diện người dùng trên các ứng dụng, như tổ hợp phím, click chuột, người dùng quan sát những thay đổi mà kết quả trong giao diện người dùng và xác minh rằng các điều khiển được điều khiển chính xác với các giá trị đúng.

2.4 White Framework

White là một framework dành cho việc tự động hóa các ứng dụng phong phú của Client dựa trên Win32, WinForms, WPF, Silverlight và SWT (Java) platforms. Framework được xây dựng dựa trên .Net và không yêu cầu sử dụng bất kỳ ngôn ngữ độc quyền nào. Chương trình kiểm tra/ tự động hóa bằng cách sử dụng White có thể được viết với bất kỳ ngôn ngữ .NET, IDE và các công cụ bạn đã sử dụng. White cung cấp bộ API hướng đối tượng phù hợp, che dấu sự phức tạp của thư viện UIAutomation của Microsoft (là cơ sở của While) và Windows messages.

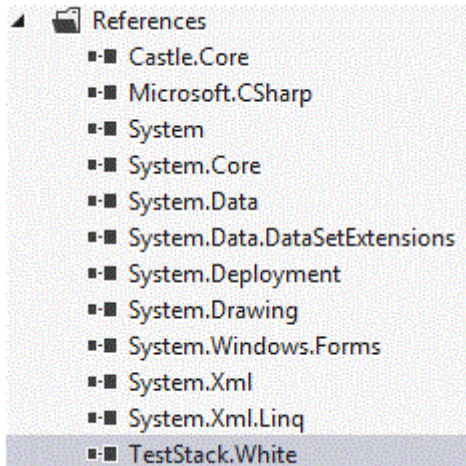
Kiến trúc của White framework:



Hình 2.2: Kiến trúc White framework

Thao tác cơ bản với White framework

Đầu tiên bạn phải cài đặt thư viện TestStack. White bằng cách tải thư viện và thêm vào references như hình sau:



Và hãy chắc chắn rằng bạn đã sử dụng những namespaces sau đây.

```
1 using White.Core;
2 using White.Core.UIItems;
3 using White.Core.UIItems.Finders;
4 using White.Core.UIItems.MenuItems;
5 using White.Core.UIItems.TreeItems;
6 using White.Core.UIItems.WindowItems;
7 using White.Core.UIItems.WindowStripControls;
```

Ví dụ:

Cách giữ được một đối tượng Windows:

```
Application application = Application.Launch("foo.exe");
Window window = application.GetWindow("bar", InitializeOption.NoCache);
```

Tìm kiếm một UI Item và thực hiện hành động:

```
Button button = window.Get<Button>("save");
button.Click();
```

Tìm một UIItem dựa trên SearchCriteria:

```
SearchCriteria searchCriteria = SearchCriteria.ByAutomationId("name").AndControlType(typeof(TextBox)).AndIndex(2);
TextBox textBox = (TextBox) window.Get(searchCriteria);
textBox.Text = "Anil";
```


2.5 Các công cụ kiểm thử tự động

Hiện nay trên thị trường có khá nhiều ứng dụng hỗ trợ việc kiểm tra tự động từ các công cụ kiểm tra tính năng, hiệu năng, đến công cụ hỗ trợ kiểm tra hệ thống mạng, cơ sở dữ liệu.

Các công cụ kiểm thử tự động thương mại chịu sự chi phối, điều khiển của một tổ chức, công ty hoặc doanh nghiệp sản xuất phần mềm. Các hoạt động sử dụng, phân phối phần mềm đều phải được sự đồng ý của họ, điều đó có nghĩa là bạn phải trả một chi phí khá cao nếu muốn sử dụng những công cụ này. Hơn thế nữa, các công cụ thương mại thường không công khai mã nguồn, mọi hành vi thay đổi hoặc sao chép (ý tưởng hoặc mã nguồn) đều được xem là vi phạm bản quyền (license key).

Trong khi đó, các công cụ kiểm thử tự động mã nguồn mở cho phép bạn tự do sử dụng, mà không chịu sự ràng buộc về mặt pháp lý. Bạn cũng có thể cải tiến, bổ sung tính năng mới và phân phối lại phần mềm đó, nhưng phải tuân thủ theo các giấy phép mã nguồn mở. Tuy nhiên, các công cụ kiểm thử tự động mã nguồn mở còn nhiều mặt hạn chế so với các công cụ kiểm thử tự động thương mại.

Dưới đây chúng tôi chọn ra một số công cụ kiểm thử tự động có uy tín trong lĩnh vực kiểm tra tự động, cụ thể là các công cụ kiểm tra tự động tính năng và hiệu năng của ứng dụng web để giới thiệu.

2.5.1 Công cụ kiểm thử Test Complete

2.5.1.1 Giới thiệu về Test Complete

TestComplete là một môi trường kiểm thử tự động cho một loạt các loại ứng dụng và công nghệ, bao gồm Windows, .NET, WPF, Visual C++, Visual Basic, Delphi, C++ Builder, Java và các ứng dụng web và dịch vụ.

TestComplete hiện nay được sử dụng bởi hơn 5000 công ty.

TestComplete được phát triển đầu tiên vào năm 1999 bởi công ty AutomatedQA với cái tên Aqtest. Từ đó cho đến năm 2014, TestComplete trải qua nhiều phiên bản khác nhau. Phiên bản hiện tại là TestComplete 9.31.

2.5.1.2 Đặc điểm của Test Complete

Các chức năng chính:

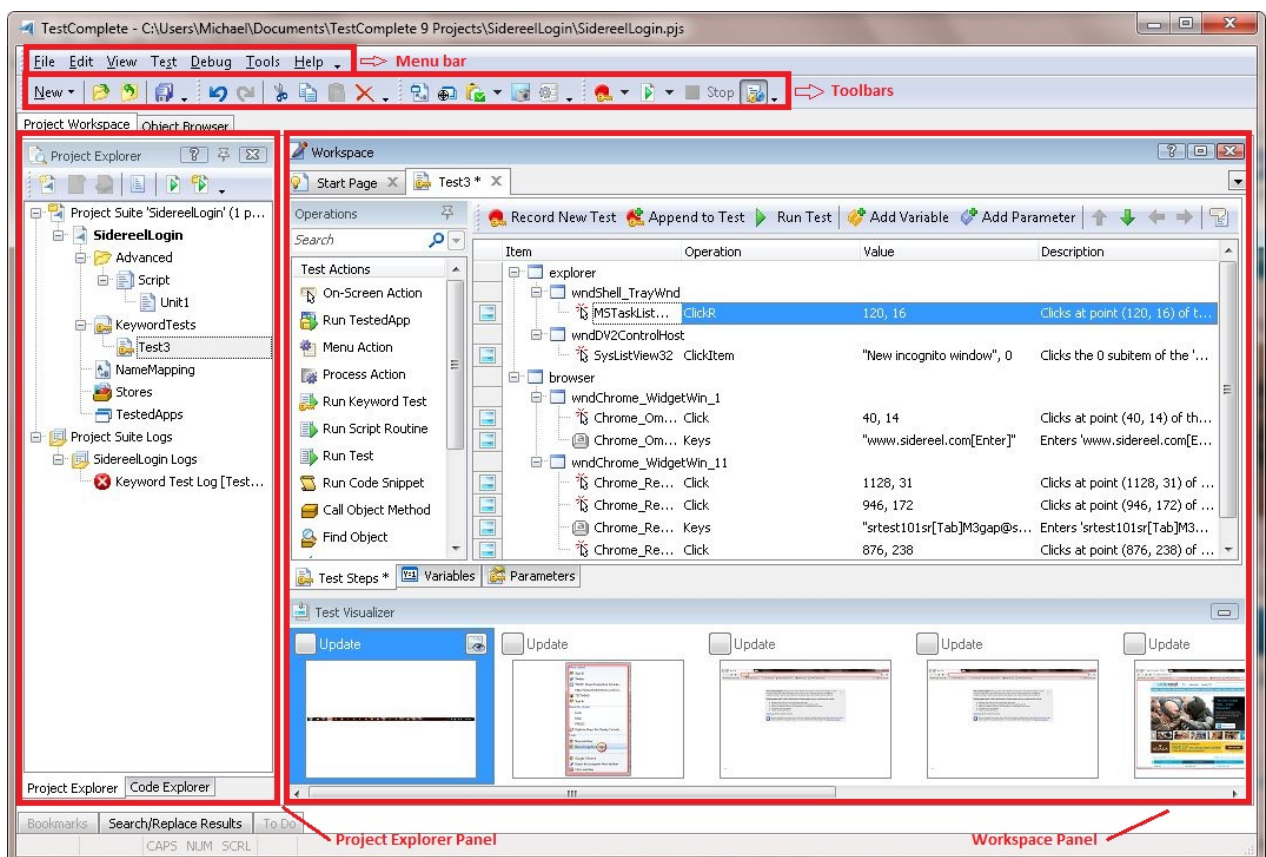
- Keyword Testing: Kiểm tra từ khóa.
- Full-Featured Script Editor: Chỉnh sửa đầy đủ các đoạn Script.

- Test Record and Playback: Cho phép ghi và chạy lại quá trình kiểm thử.
- Script Debugging Features: Gỡ lỗi.
- Access to Methods and Properties of Internal Objects: Truy cập đến phương thức và thuộc tính của bên trong đối tượng.
- Unicode Support: Hỗ trợ bộ gõ Unicode.
- Issue-Tracking Support.

Các ngôn ngữ viết được hỗ trợ:

- VBScript
- JScript
- DelphiScript
- C++Script
- C#Script

2.5.1.3 Giao diện phần mềm

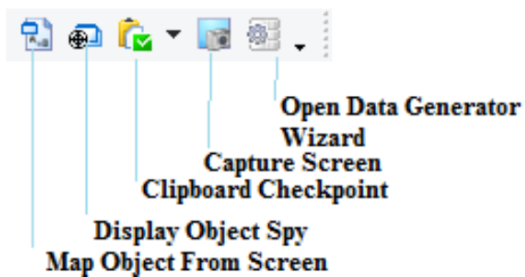


Hình 2.3: Giao diện chính

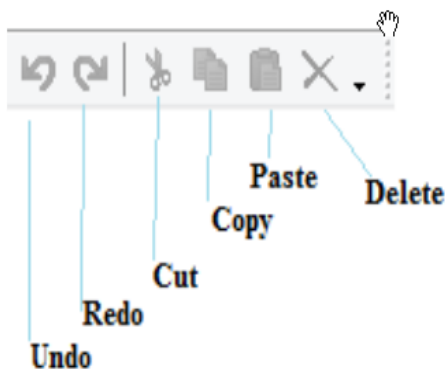
- Menubar: Hiển thị những Menu của công cụ.
- Toolbar: Chứa những button để giúp bạn quản lý test của bạn.
- Project Explorer Panel: Sử dụng để hiển thị và thay đổi cấu trúc của các dự án TestComplete và dãy dự án. Nó cũng hiển thị cấu trúc của các bản record của các dự án và chạy thử nghiệm.
- Workspace Panel: Là khu vực làm việc chính của bạn trong TestComplete và giữ chỗ cho các kiểm thử viên, cho phép bạn xem và chỉnh sửa các nội dung trong dự án, hạng mục công trình và các bản ghi đăng nhập.

Giới thiệu 1 số thanh công cụ trên giao diện chính:

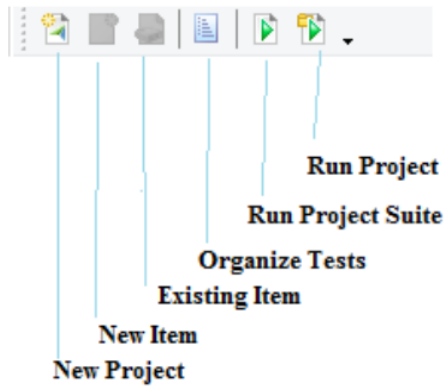
Tools: Cung cấp các lệnh có ảnh hưởng đến quá trình Record.



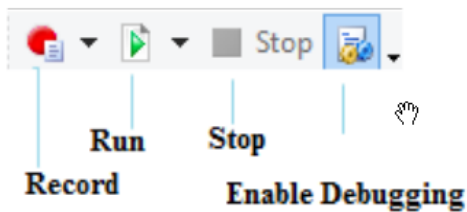
Edit: Cung cấp các lệnh chỉnh sửa tiêu chuẩn.



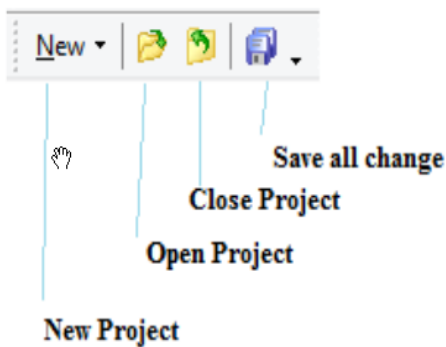
Project Explorer: Cung cấp các lệnh ảnh hưởng tới Project Explorer Panel.



Test Engine: Cung cấp các lệnh có ảnh hưởng ghi lại và gỡ rối kiểm thử.



Standard: Cung cấp các lệnh có ảnh hưởng đến cấu trúc của các bộ dự án hiện tại.



Sau một thời gian đọc tài liệu và sử dụng phần mềm nhóm có một số nhận xét về công cụ TestComplete cụ thể như sau:

Ưu điểm:

- Không giới hạn với các ứng dụng thử nghiệm.
- Không phụ thuộc vào công cụ phát triển.
- Hỗ trợ nhiều loại kiểm thử khác nhau.
- Hỗ trợ nhiều ngôn ngữ.
- Dễ sử dụng, dễ hiểu.

Nhược điểm:

- Không có bản free. Bản trial hạn chế nhiều tính năng của TestComplete

2.5.2 Công cụ kiểm thử TestArchitect™

2.5.2.1 Giới thiệu về TestArchitect™

Công cụ kiểm thử tự động TestArchitect™ là trung tâm của cấu trúc nền tảng kiểm thử tự động cho phép đội ngũ kiểm thử làm được nhiều việc hơn, giảm chi phí để có thể nhanh chóng tung ra thị trường. TestArchitect™ tích hợp phương pháp và công nghệ hiện đại vào một gói sản phẩm dễ dàng sử dụng.

TestArchitect™ là một phần của cấu trúc nền tảng Action Based Testing™, cho phép khách hàng sắp xếp hợp lý việc phát triển phần mềm với đội ngũ kiểm thử, thiết kế kiểm thử, kỹ thuật tự động.

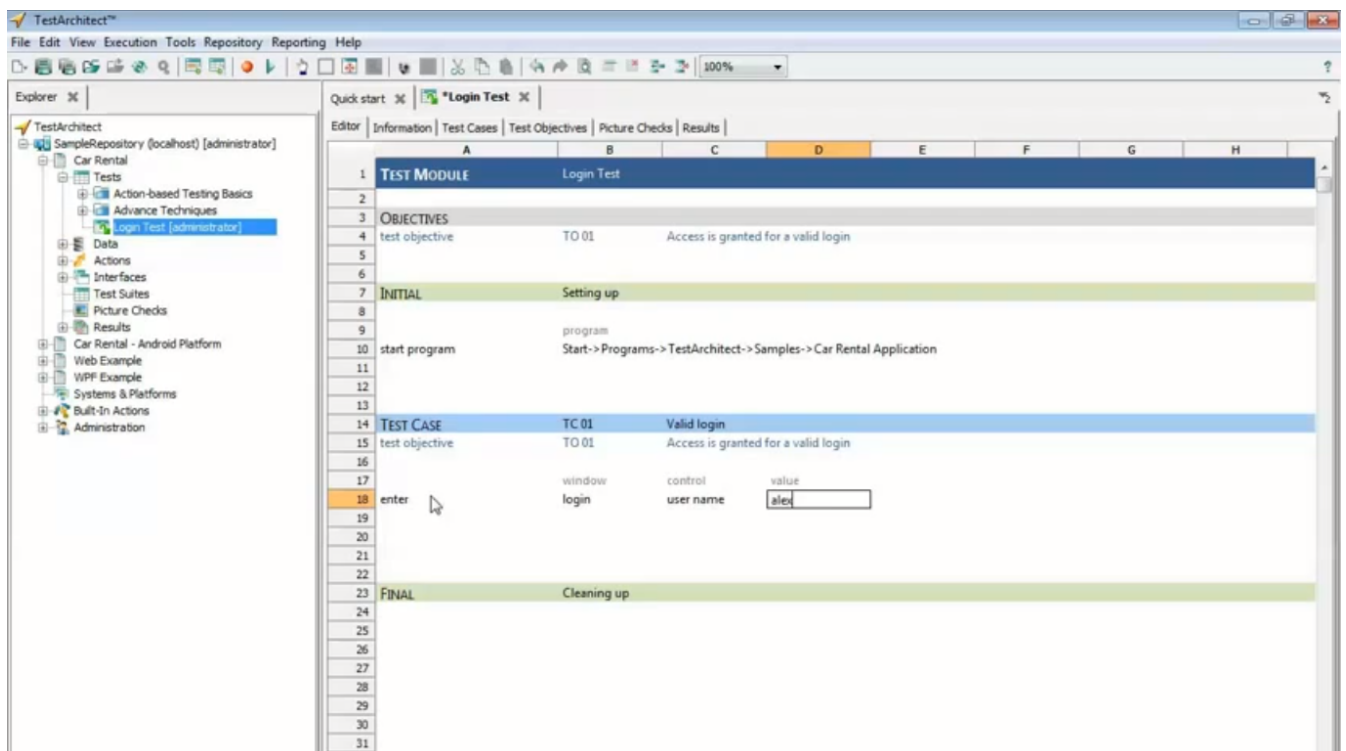
2.5.2.2 Đặc điểm của TestArchitect™

Chức năng:

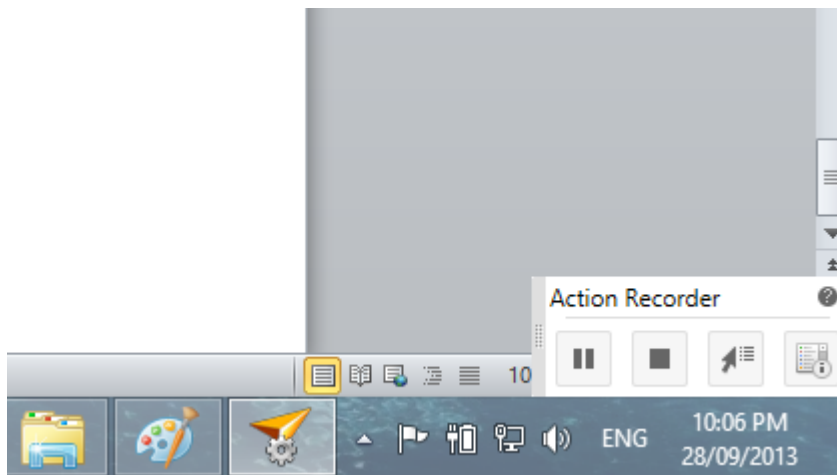
TestArchitect™ làm việc trên Test Module, quá trình làm việc dựa trên Test Case sẽ được đưa vào một bảng, trong đó chứa quá trình làm việc của từng Test Case.

Có thể viết Test Case thông qua 2 cách:

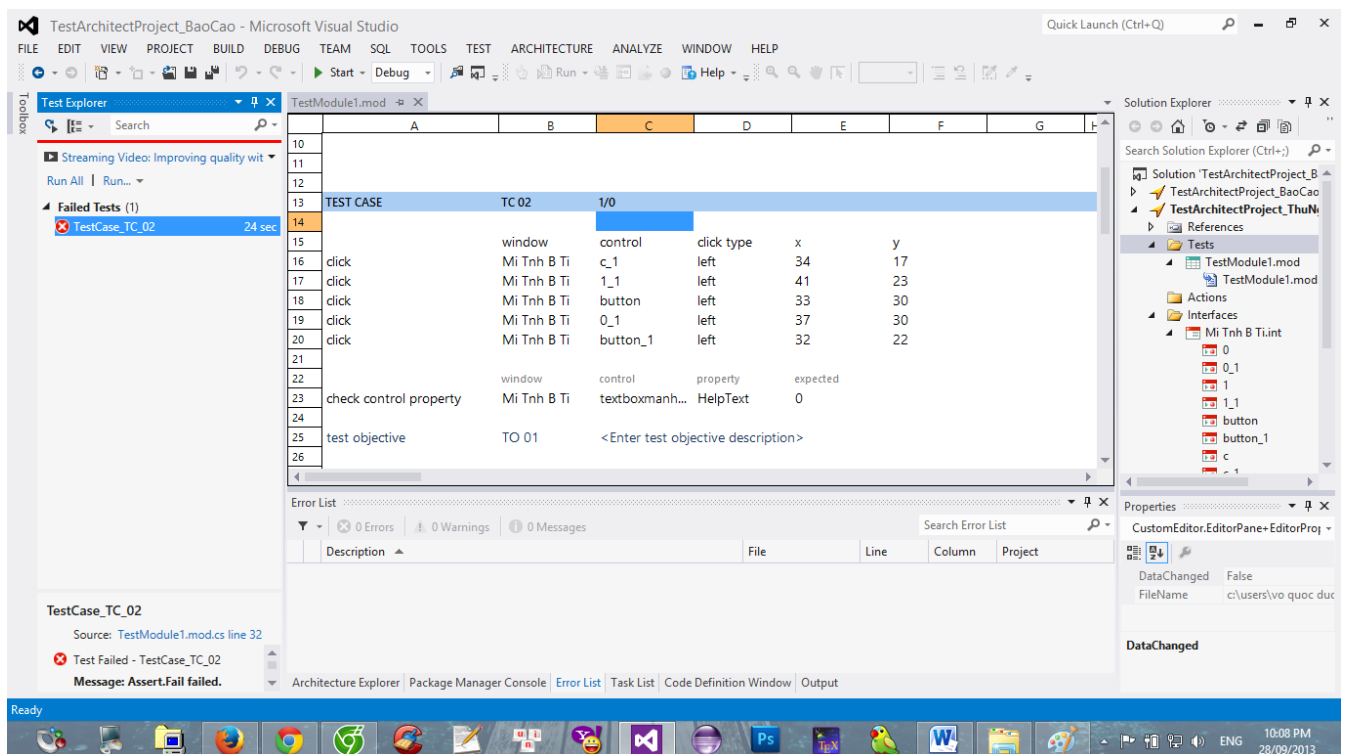
Cách 1: Viết các Script bằng cách thủ công tức là bạn soạn thảo trực tiếp các Test Case trong cửa sổ làm việc (Editor).



Cách 2: Bạn có thể Record lại các thao tác và sau khi hoàn thành quá trình Record thì chương trình sẽ xuất ra một file Script tương ứng.



Dưới đây là đoạn Test Case được sinh ra bằng thao tác Record, bạn cũng có thể chỉnh sửa lại nội dung của TestCase như viết bằng tay.



Phần quan trọng cũng không kém trong kiểm thử tự động là Report, cho biết chính xác quá trình Test có bị lỗi gì ko? Với mỗi lần chạy Test Case chương trình sẽ tự động tạo ra một file Report tương ứng như sau:

TC 02 (2013-09-28 21.55.27)

Test module	TestModule1
Start time	09/28/2013 21:55:27
Duration	00:00:04
Run machine	VoQuocDuoc

⊖ Test case summary

Total test cases executed: 1		Passed: 0	Failed: 0	
Test Case	Checks	Passed	Failed	Error
TC 02, 1/0				1
Total				1

⊖ Detailed Results Per Test Line

INITIAL

3: test objective: TO 01 - Ki??m Tra Infinity

⊖ Test case: TC 02, 1/0

Error(s) at line(s): 23

16: click [window: Mi Tinh B Ti] [control: c_1] [click type: left] [x: 34] [y: 17]

17: click [window: Mi Tinh B Ti] [control: 1_1] [click type: left] [x: 41] [y: 23]

18: click [window: Mi Tinh B Ti] [control: button] [click type: left] [x: 33] [y: 30]

19: click [window: Mi Tinh B Ti] [control: 0_1] [click type: left] [x: 37] [y: 30]

20: click [window: Mi Tinh B Ti] [control: button_1] [click type: left] [x: 32] [y: 22]

⊕ (E) 23: check control property [window: Mi Tinh B Ti] [control: textboxmanhinh_1] [property: HelpText] [expected: 0]

25: test objective: TO 01 - <Enter test objective description>

FINAL

Ưu điểm:

- Làm việc trên trình soạn thảo riêng.
- Dễ nhận ra thứ tự các Test Case.
- Có thể chuyển đổi file Script dưới dạng C# (Coded UI).
- File Script dễ dàng sửa chữa.

Nhược điểm:

- Trong quá trình Record không sao lưu lại quá trình làm việc dưới dạng hình ảnh.

Chương 3

Phân Tích Thiết Kế Hệ Thống

3.1 Tổng quan về hệ thống

Phần mềm kiểm thử tự động các ứng dụng Desktop là một công cụ chuyên dùng để kiểm tra tính năng của các ứng dụng Desktop, được viết bằng ngôn ngữ lập trình C# và Window Presentation Foundation. Hệ thống có thể hoạt động trên nhiều phiên bản của hệ điều hành Windows có cài đặt .NET Framework, với phiên bản từ 2.0 trở lên. Hệ thống không cần kiến thức lập trình chuyên sâu để sử dụng.

Kiểm thử được hầu hết các ứng dụng Windows: Phát triển bằng công nghệ của Microsoft .

Với mong muốn giao diện thân thiện đẹp mắt, dễ sử dụng được xây dựng bằng Window Presentation Foundation, và C# hệ thống giúp kiểm thử viên làm việc một cách dễ dàng, nhanh chóng, hiệu quả, có hứng thú làm việc.

3.2 Các Chức Năng Của Hệ Thống

3.2.1 Chức Năng Quản Lý

- Chức năng mở (Open): Cho phép mở một Project, Interface, Data và Script.
- Chức năng tạo mới (New): Tạo mới Project, Interface, Data và Script.
- Chức năng soạn thảo (Editor): Hệ thống cung cấp một giao diện thân thiện, dễ sử dụng để soạn thảo các Test Case, Test Script và Test Data
- Chức năng chỉnh sửa (Edit): Gồm có những chức năng như thêm, xóa, sửa, sao chép, dán một đối tượng nào đó trong chương trình.

- Chức năng hiển thị (View): Cho phép hiển thị các sổ làm việc như Project Explorer, Full Screen... Hoặc ngược lại, hệ thống cho phép ẩn các cửa sổ làm việc để không gian làm việc có thể rộng rãi, thoải mái hơn, tùy ý người dùng.

3.2.2 Chức Năng Sao Lưu Dữ Liệu

- Chức năng lưu trữ (Save): Lưu trữ các Test Case, Test Script và các hàm đã được định nghĩa. Lưu trữ dưới dạng file excel.
- Chức năng Export Report: Sau khi chương trình chạy các bộ Test Case sẽ xuất ra file Report để người dùng có thể xem và đánh giá.

3.2.3 Chức Năng Thực Thi

Chức năng thực thi: Gồm có bắt đầu thực thi, tạm ngưng, dừng.

Người dùng có thể chọn các Test Case hoặc Test Suite để chạy.

Trong đó, hệ thống cung cấp nhiều tùy chọn như:

- Chạy tuần tự nhiều Test Case hoặc Test Suite
- Chạy đồng thời nhiều Test Case hoặc Test Suite

3.2.4 Chức Năng Báo Cáo Kết Quả (Report)

Đây là một trong những chức năng quan trọng và chính yếu nhất của hệ thống, sau khi các Test Case hoặc Test Suite chạy xong, hệ thống sẽ cho ra một bản báo cáo cho biết kết quả chạy của Test Case, Test Suite là passed hay failed, có đúng với kết quả mong đợi hay không, ngoài ra còn cung cấp những thông số như: thời gian thực thi, dữ liệu thực thi, cảnh báo, lỗi phát sinh. Các báo cáo sẽ được lưu trữ dưới định dạng excel.

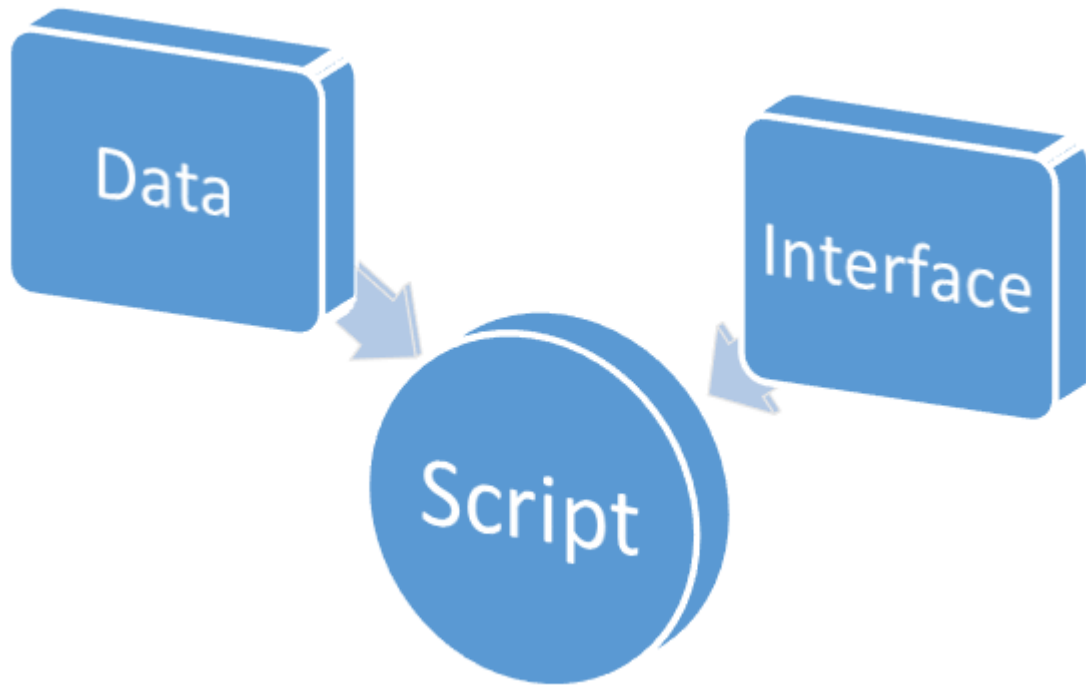
3.2.5 Chức Năng Tìm Kiếm Các Tiêu Chí (Spy)

Chức năng hỗ trợ việc tìm kiếm các tiêu chí để phân biệt và xác định các đối tượng đang chạy trên nền Windows. Khi chạy chương trình, Spy sẽ liệt kê tất cả các chương trình đang chạy trên Desktop, người dùng có nhiệm vụ chọn vào chương trình muốn Test để chọn các tiêu chí của đối tượng. Spy hỗ trợ người dùng trong việc tạo Interface (nơi lưu trữ các tiêu chí để xác nhận đối tượng cần Test).

3.2.6 Chức Năng Hỗ Trợ Cài Đặt

Cài đặt ngôn ngữ: Hệ thống cung cấp cho người dùng hai tùy chọn ngôn ngữ là tiếng anh và tiếng việt.

3.2.7 Cấu Hình Hệ Thống



Hình 3.1: Kiến trúc hệ thống

Dựa theo phương pháp Action Base Testing thì chúng tôi phát triển hệ thống theo mô hình sau:

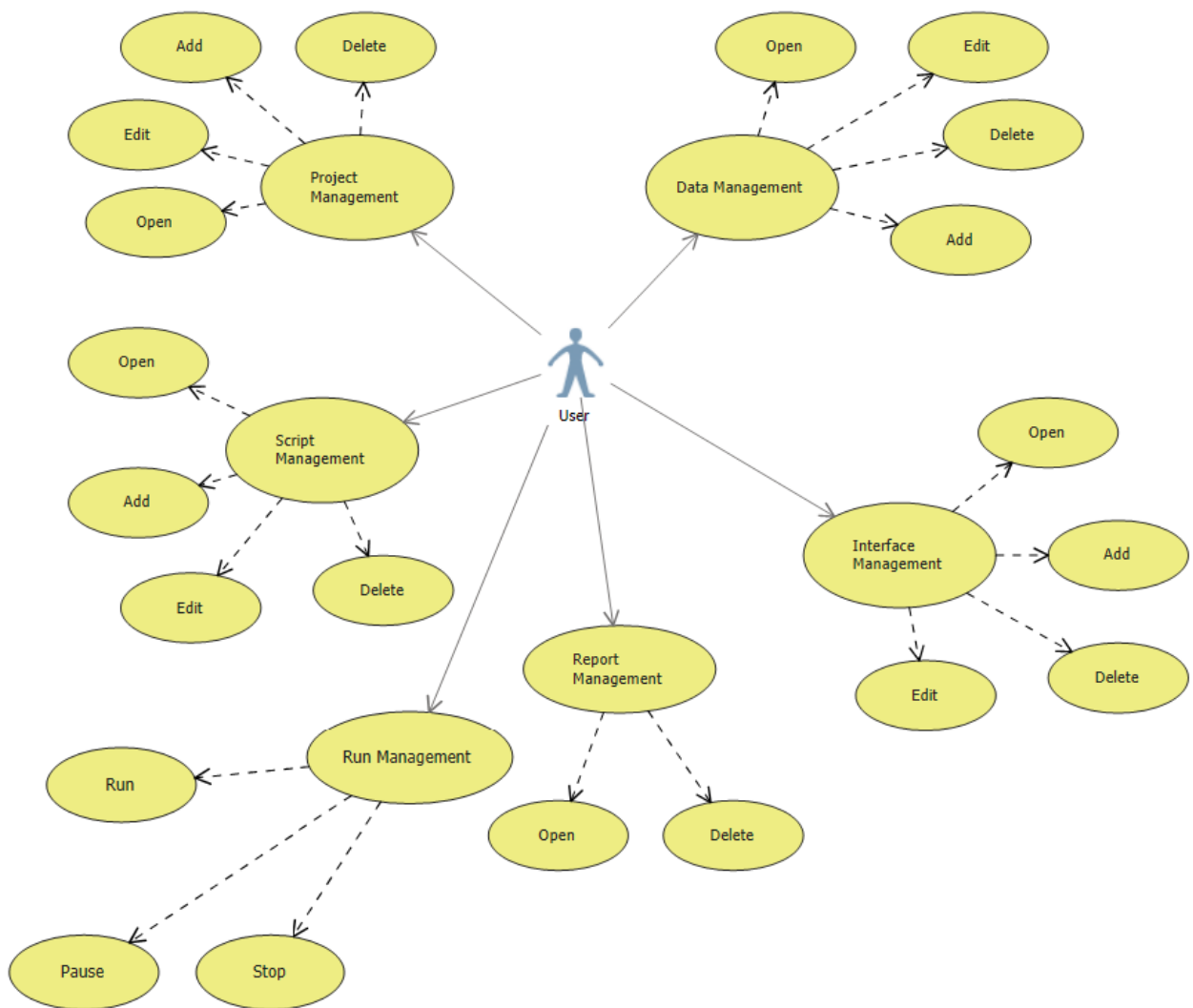
- Data : là nơi lưu trữ các bộ dữ liệu phục vụ cho quá trình Test.
- Interface: Định nghĩa các đối tượng dựa vào các tiêu chí tìm được thông qua công cụ Spy. Lưu trữ các tiêu chí để xác định các đối tượng khác nhau trong Windows nhằm hỗ trợ cho quá trình gọi đối tượng chính xác.
- Script: Lưu trữ tất cả các Test Case do người dùng soạn thảo, một Script có thể chứa một hoặc nhiều Test Case tùy thuộc vào nhu cầu Test của người dùng.

Phương thức hoạt động: Khi bắt đầu chạy các Test Case, chương trình sẽ đọc toàn bộ định nghĩa của đối tượng cần Test thông qua các tiêu chí được lưu trong Interface. Sau khi đã

xác định được đối tượng, chương trình sẽ chạy lần lượt các Test Case trong file Script, khi cần Test với bộ dữ liệu chương trình sẽ tải dữ liệu từ Data xuống cứ như vậy một quá trình Automation Test được hình thành.

3.3 Lược Đồ Use Case Và Đặc Tả

3.3.1 Sơ Đồ Tổng Quát



Hình 3.2: Sơ đồ tổng quát

3.3.1.1 Chức Năng Quản Lý Project

ID	NAME	Project Management
DESCRIPTION	Người dùng quản lý Project.	
MAIN ACTOR	Người dùng.	
PRE -CONDITIONS	Chương trình đang hoạt động.	
FLOW	1	Chọn File: New →Project: Để tạo một Project mới. Hệ thống hiển thị frmNewProject. Nhập tên Project vào form New Project, chọn Browse để chỉ đến thư mục lưu trữ Project. Chọn OK để tạo Project - Cancel để hủy thao tác.
	2	Open → Project: Để mở một Project đã có sẵn. Hệ thống hiển thị frmOpenProject. Chọn thư mục lưu trữ Project. Hệ thống hiển thị frmOpenProject. Chọn OK để mở Project - Cancel để hủy thao tác.
	3	Click chuột phải vào Project →Chọn Detele. Hệ thống hiển thị frmDeleteProject. Chọn Ok để xác nhận xóa Project - Cancel để hủy thao tác.
POST - CONDITION		

Bảng 3.1: Project Management

3.3.1.2 Chức Năng Quản Lý Data

ID	NAME	Data Management
DESCRIPTION	Cho phép người dùng thêm, xoá, sửa Data.	
MAIN ACTOR	Người dùng.	
PRE -CONDITIONS	Chương trình đang hoạt động.	
FLOW	1	Click chuột phải Data chọn: Add: Thêm một bộ dữ liệu mới. Hệ thống hiển thị frmNewData. Người dùng nhập tên Data vào form NewData. Chọn OK để tạo mới Data - Cancel để hủy thao tác.
	2	Open: Mở một Data đã có sẵn. Hệ thống hiển thị frmOpenData. Người dùng chọn file Data cần mở. Chọn OK để mở Data - Cancel để hủy thao tác.
	3	Delete: Xóa Data đã chọn. Hệ thống hiển thị frmDeleteData. Người dùng chọn OK để chấp nhận xóa Data. Chọn Cancel để hủy thao tác Delete.
	4	Edit: Chỉnh sửa nội dung Data. Người dùng chọn file Data cần chỉnh sửa. Tùy chỉnh nội dung file Data. Click vào biểu tượng Save để lưu lại Data.
POST - CONDITION		

Bảng 3.2: Data Management

3.3.1.3 Chức Năng Quản Lý Interface

ID	NAME	Interface Management
DESCRIPTION	Cho phép người dùng thêm, xoá, sửa Interface.	
MAIN ACTOR	Người dùng.	
PRE -CONDITIONS	Chương trình đang hoạt động.	
FLOW	1	Click chuột phải vào Interface: Chọn Add: để thêm mới một file Interface. Hệ thống hiển thị frmNewInterface. Người dùng nhập tên Interface vào form. Chọn OK để tạo một file Interface mới. Chọn Cancel để hủy thao tác.
	2	Chọn Open: Để mở một file Interface có sẵn. Hệ thống hiển thị frmOpenInterface. Trong form Open chọn file Interface cần mở. Chọn OK để mở file Interface. Chọn Cancel để hủy thao tác.
	3	Chọn Delete: Xóa file Interface đã chọn. Hệ thống hiển thị frmDeleteInterface. Chọn OK để xác nhận xóa Interface. Chọn Cancel để hủy thao tác Delete.
		Edit: Chỉnh sửa file Interface. Chọn file Interface cần chỉnh sửa. Người dùng mở vào chương trình soạn thảo Interface. Chỉnh sửa nội dung Interface. Chọn biểu tượng Save để lưu lại bản Interface chỉnh sửa.
	POST - CONDITION	




Bảng 3.3: Interface Management

3.3.1.4 Chức Năng Quản Lý Test Script

ID	NAME	Test Script Management
DESCRIPTION	Cho phép người dùng thêm, xóa, sửa Test Script.	
MAIN ACTOR	Người dùng.	
PRE -CONDITIONS	Chương trình đang hoạt động.	
FLOW	1	Click Chuột phải vào Script: Chọn Add: Để tạo mới một file Script. Hệ thống hiển thị frmNewScript. Người dùng nhập tên Script vào form. Chọn OK để tạo mới file Script. Chọn Cancel để hủy thao tác.
	2	Chọn Open: để mở file Script có sẵn. Hệ thống hiển thị frmOpenScript. Chọn file Script cần mở. Chọn OK để mở file Script. Chọn Cancel để hủy thao tác.
	3	Chọn Delete: Xóa file Script Hệ thống hiển thị frmDeleteScript: “Bạn có muốn xóa file Script.” Người dùng Chọn OK để đồng ý xóa. Chọn Cancel để hủy thao tác Delete.
	4	Edit: Chỉnh sửa nội dung một file Script. Chọn file Script cần chỉnh sửa bằng cách Double Click. Người dùng chỉnh sửa nội dung trong trình soạn thảo. Chọn biểu tượng Save để lưu lại file Script.
POST - CONDITION		

Bảng 3.4: Script Management

3.3.1.5 Chức Năng Thực Thi Chương Trình

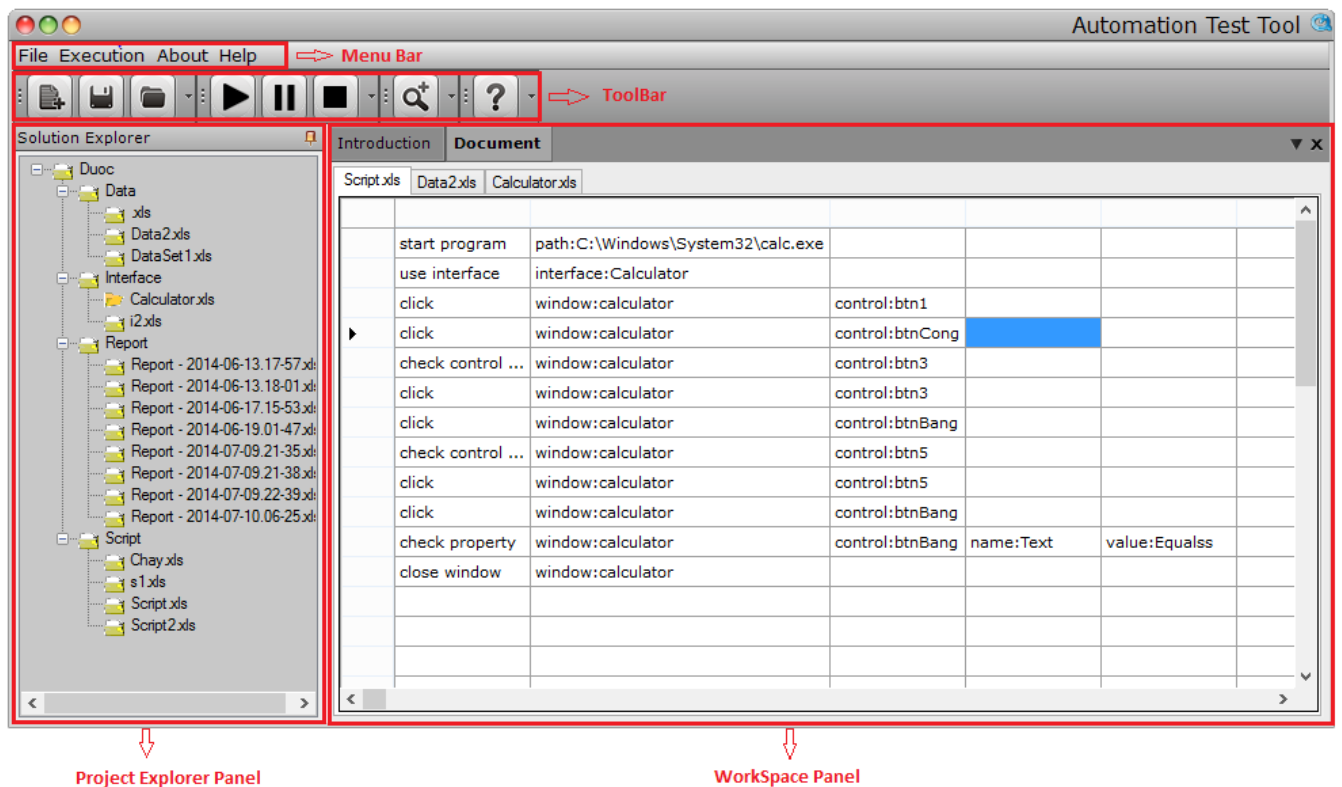
ID	NAME	Run Management
DESCRIPTION	Cho phép người dùng chạy, tạm dừng và ngừng quá trình Test.	
MAIN ACTOR	Người dùng.	
PRE -CONDITIONS	Chương trình đang chạy các Test Case.	
	1	<p>Run:</p> <p>Sau khi đã nhập đầy đủ các Test Case.</p> <p>Người dùng muốn chạy chương trình thì thực hiện bằng cách:</p> <p>Chọn vào biểu tượng Run trên thanh ToolBar .</p>
	2	<p>Pause:</p> <p>Chương trình đang chạy các bộ Test Case.</p> <p>Người dùng muốn tạm dừng chương trình thì thực hiện như sau:</p> <p>Chọn vào biểu tượng Pause trên thanh ToolBar .</p> <p>Chọn Cancel để hủy thao tác.</p>
	3	<p>Stop:</p> <p>Chương trình đang chạy các bộ Test Case</p> <p>Người dùng muốn dừng chương trình và thoát ra.</p> <p>Chọn vào biểu tượng Stop trên thanh Toolbar .</p>
POST - CONDITION		

3.3.1.6 Chức Năng Quản Lý Báo cáo

ID	NAME	Report Management
DESCRIPTION	Cho phép người dùng xem và xóa báo cáo.	
MAIN ACTOR	Admin.	
PRE -CONDITIONS	Chương trình đang hoạt động.	
FLOW	1	Click chuột phải vào Report: Chọn Open: để xem kết quả file Report. Người dùng chọn file báo cáo cần xem bằng cách chuột phải chọn Open hoặc Double Click vào file báo cáo cần xem.
	2	Chọn Delete: Xóa một file Report. Hệ thống hiển thị frmDeleteReport với nội dung: “ Bạn có muốn xóa file báo cáo này không” Chọn OK để xác nhận xóa file. Chọn Cancel để hủy thao tác Delete.
POST - CONDITION		

Bảng 3.5: Report Management

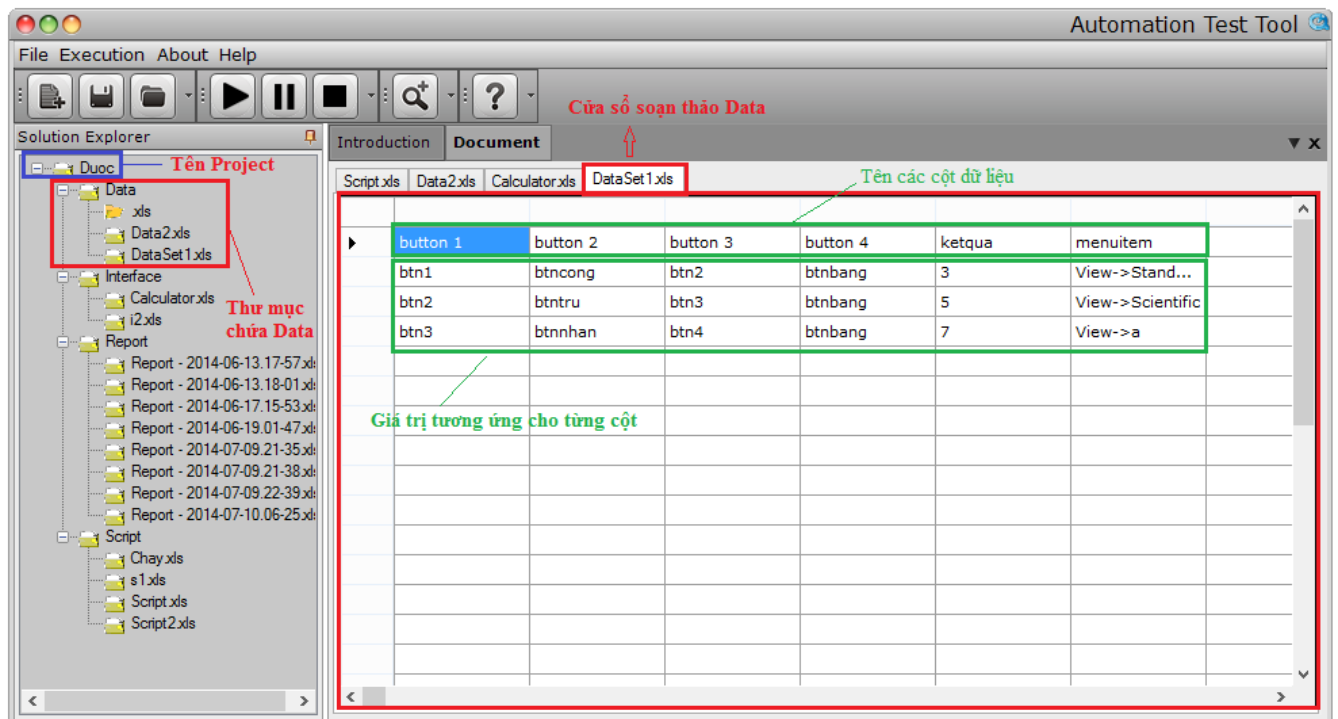
3.4 Giao Diện Của Hệ Thống



Hình 3.3: Giao diện hệ thống

- MenuBar: Hiển thị tất cả các Menu công cụ như: File (New, Open, Save, Close một Project or File), Execution (Thực hiện các chức năng thực thi như Run, Pause, Stop quá trình chạy Test Case, ngoài ra người dùng có thể sử dụng các tổ hợp phím tắt tương ứng), About (Hiển thị thông tin chi tiết của chương trình), Help (Hướng dẫn sử dụng chương trình cho người dùng).
- ToolBar: Chứa các button để giúp bạn quản lý và thực thi các Test Case.
- Project Explorer Panel: Hiển thị cấu trúc của một Project theo dạng cây thư mục. Khi người dùng tạo một Project thì chương trình sẽ tự sinh ra 4 thư mục chính. Data: Chứa các bộ dữ liệu Test, Interface: chứa các tiêu chí để xác định đối tượng, Script: Thư mục lưu trữ các Test Script và cuối cùng là Report: nơi lưu trữ các file Report.
- Workspace Panel: Là khu vực làm việc chính giúp người dùng soạn thảo Data, Interface và Test Script, cho phép người dùng xem, chỉnh sửa nội dung trong Project. Trình soạn thảo hiển thị theo dạng lưới nên người dùng rất dễ sử dụng.

Giao diện soạn thảo Data:

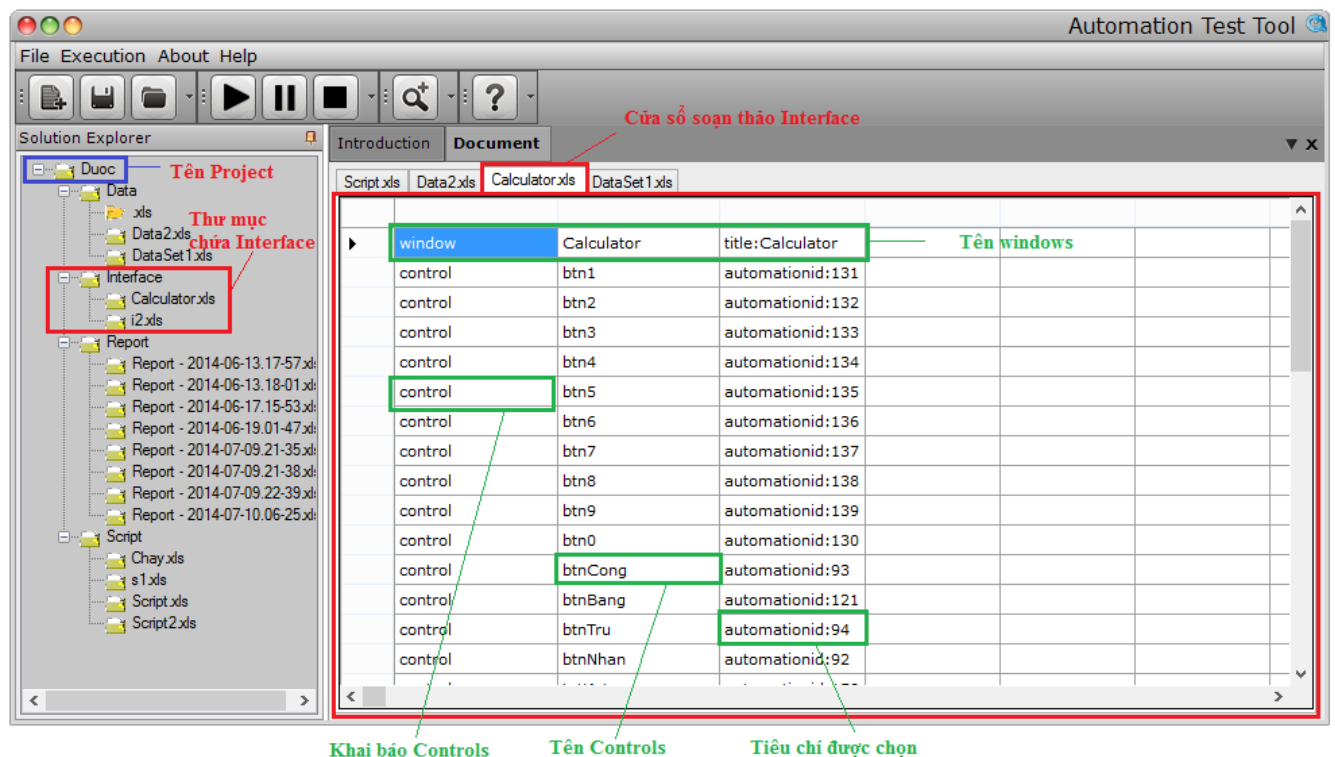


Hình 3.4: Giao diện soạn thảo Data

Đối với việc Test với một bộ dữ liệu người dùng phải nhập các thông số cần thiết trong file Data. Dòng đầu tiên là tên các cột dữ liệu và cũng là tên biến để truy xuất dữ liệu trong quá trình Test. Tùy thuộc vào số lượng biến truyền vào Test Case thì người dùng khai báo bấy nhiêu cột tương ứng. Dòng thứ 2 trở đi chính là các giá trị của biến.

Test Data dùng trong trường hợp người dùng phải lặp đi lặp lại một bộ Test Case với những dữ liệu khác nhau. Dùng Test Data sẽ tiết kiệm được thời gian và công sức của kiểm thử viên.

Giao diện soạn thảo Interface:



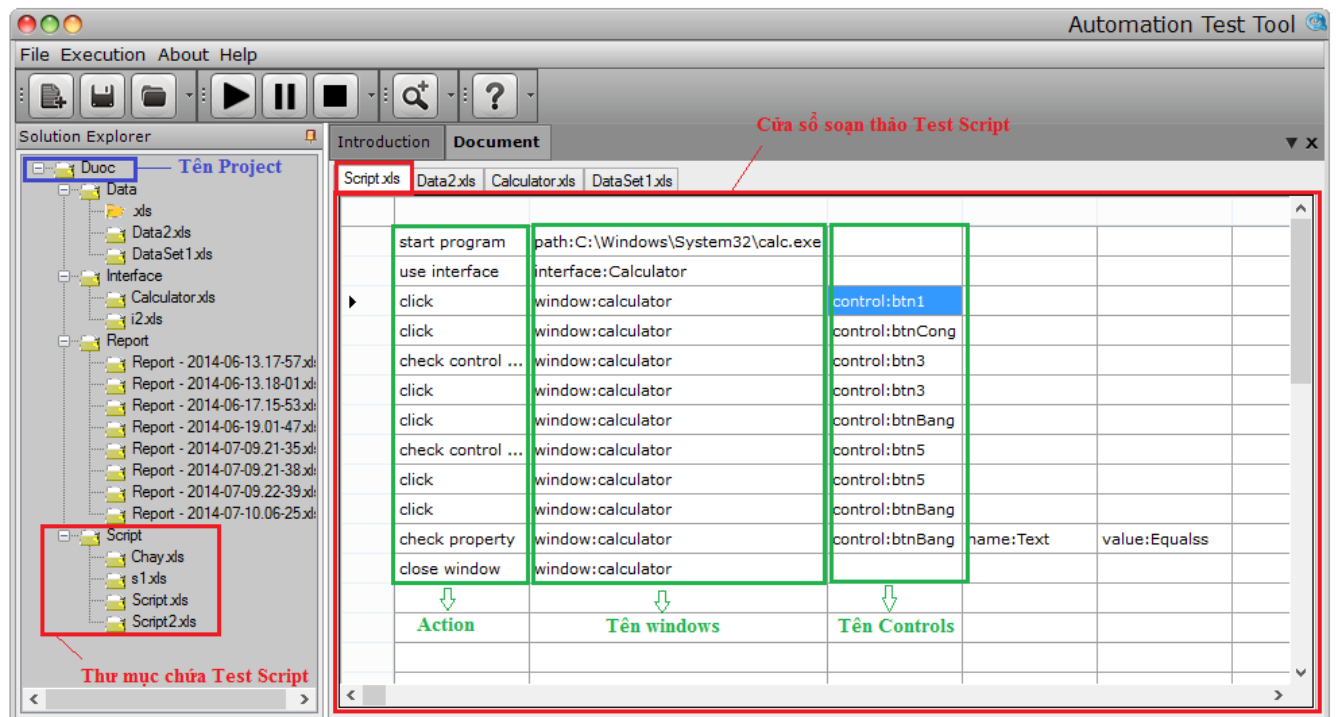
Hình 3.5: Giao diện soạn thảo Interface

Khai báo Interface là công việc quan trọng trong chương trình. Khi người dùng muốn Test một chương trình thì phải khai báo tất cả những thuộc tính cần thiết cho quá trình Test thông qua công cụ Spy, việc làm này nhằm giúp chương trình xác định đúng đối tượng. Dòng đầu tiên sẽ khai báo tên windows với tiêu chí của nó.

Ví dụ: Như hình trên tên Windows: Caculator và tiêu chí để xác định là title: Caculator.

Các dòng tiếp theo là định nghĩa các Controls trong Windows tìm được. Cột thứ nhất là khai báo Controls, cột thứ hai là tên Controls do người dùng đặt tên và cuối cùng là tiêu chí xác định nó, người dùng có thể chọn nhiều tiêu chí để làm rõ hơn đối tượng cần Test.

Giao diện soạn thảo Test Script:

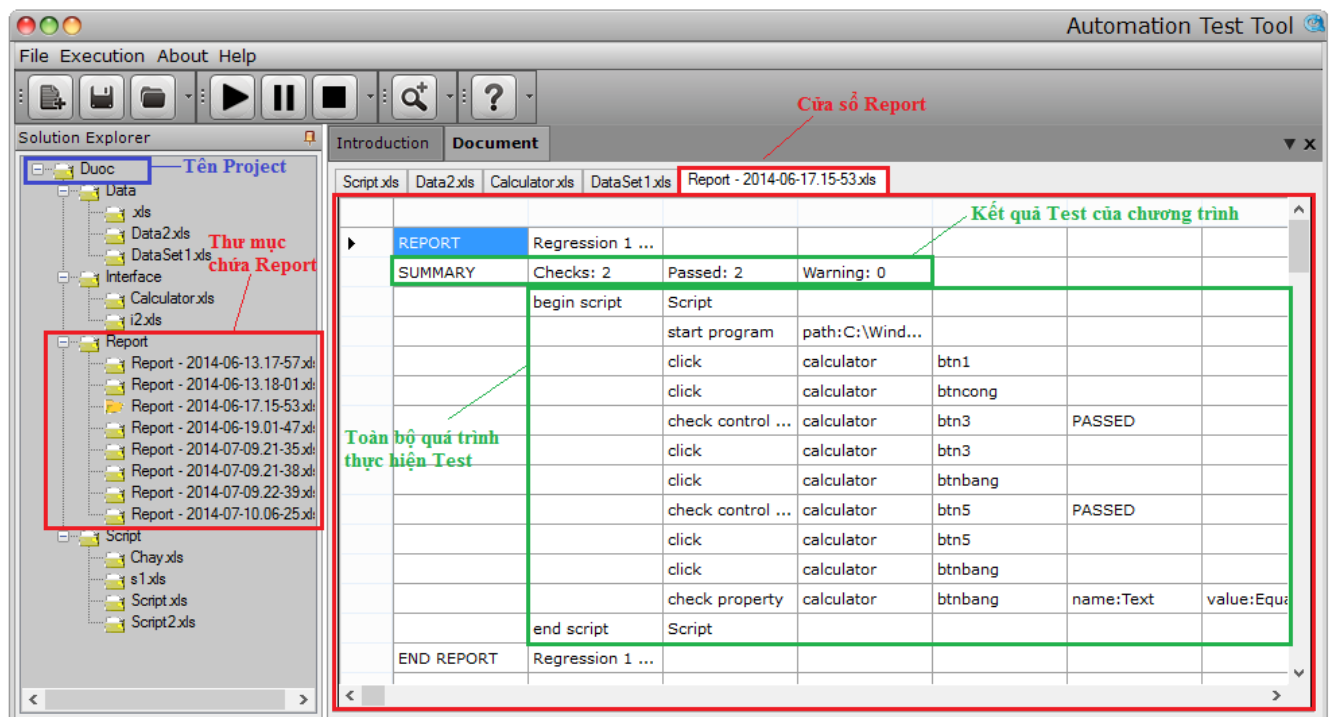


Hình 3.6: Giao diện soạn thảo Test Script

Đây là cửa sổ soạn thảo giúp quản lý và nhập nội dung Test Case. Người dùng có thể nhập một hay nhiều bộ Test Case tùy thuộc yêu cầu của công việc.

Cấu trúc của Test Script: Cột đầu tiên người dùng nhập tên Action, tùy thuộc vào những Action được gọi ra thì các cột tiếp theo người dùng truyền vào lần lượt tên Windows, tên Controls và những tham số tương ứng. Để biết thêm nhiều về các Action được hỗ trợ, có thể xem trong phần Help.

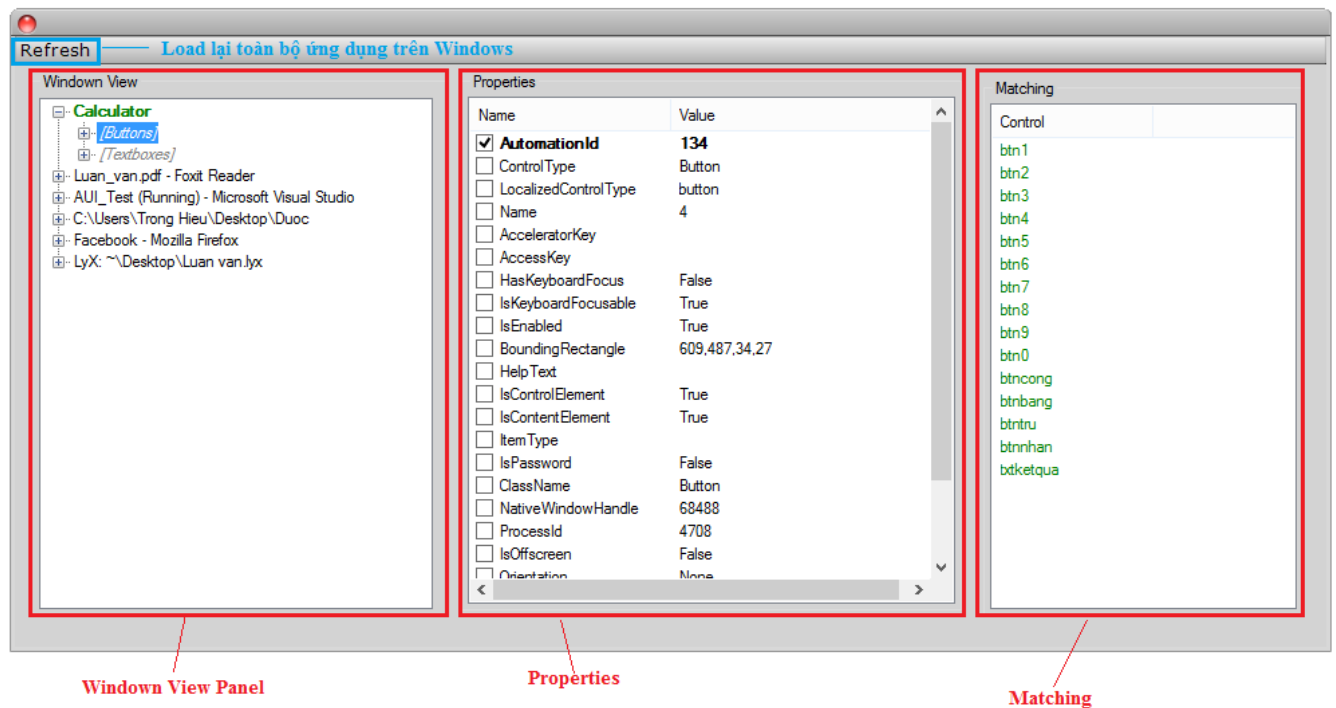
Giao diện của file báo cáo:



Hình 3.7: Giao diện Report

Report: Xuất ra kết quả sau khi chạy các bộ Test Case, giúp người dùng xác định rõ vấn đề của chương trình được Test. Kết quả trả về là Passed nếu quá trình Test đạt được những yêu cầu mà người dùng đưa ra và Failed nếu không thỏa mãn bất kỳ một tiêu chí nào đó. Trong file báo cáo dòng thứ 2 sẽ liệt kê lần lượt Checks (số lượng những tiêu chí cần Test), Passed (Số lượng tiêu chí đạt yêu cầu) và Warning. Các dòng tiếp theo là quá trình Test chi tiết để người dùng có thể tìm hiểu rõ hơn vấn đề.

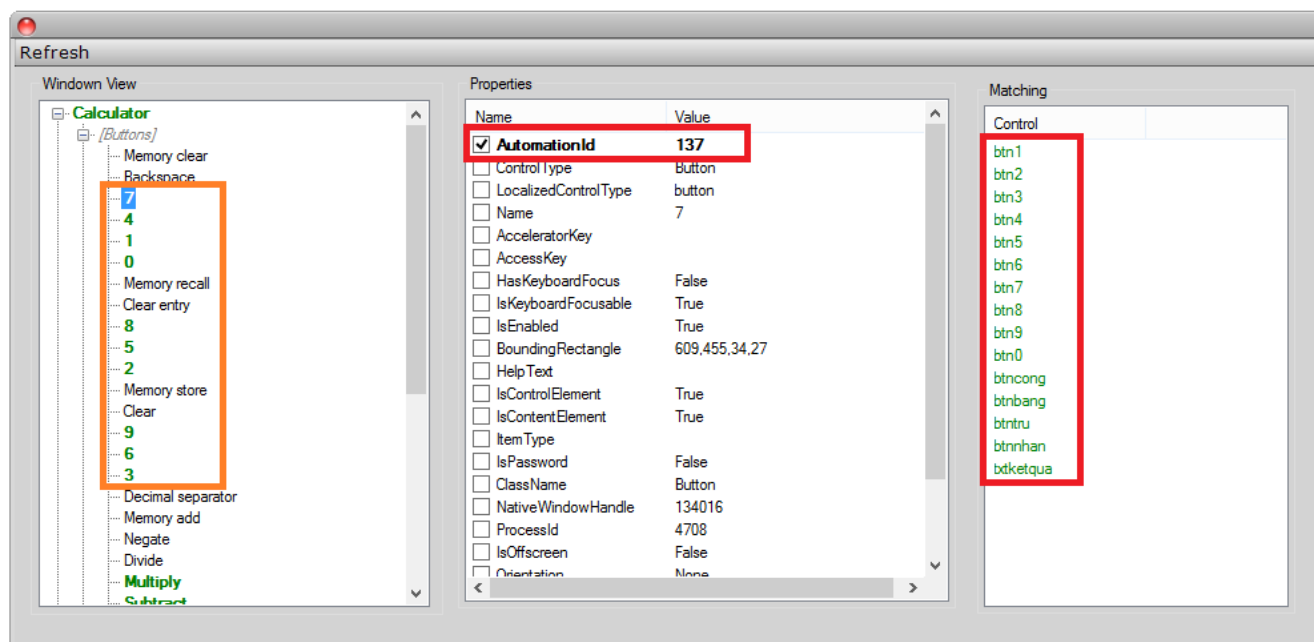
Giao diện Spy:



Hình 3.8: Giao diện Spy

- **Windows View Panel:** Liệt kê tất cả các ứng dụng đang chạy trên Desktop hiện tại, giúp người dùng dễ dàng chọn được đối tượng cần Test.
- **Properties:** Hiển thị những thuộc tính sẵn có của chương trình được Test, ở cửa sổ này người dùng sẽ chọn những tiêu chí riêng biệt để xác định rõ đối tượng và lưu vào Interface.
- **Matching:** Tất cả các Control có trong cửa sổ này được Load lên từ Interface nhằm mục đích đối chiếu với những tiêu chí của đối tượng. Nếu Control đó thực sự tồn tại thì sẽ được tô màu xanh. Mục đích nhằm giúp người dùng tránh sai sót khi nhập Interface, đây được xem là bước kiểm tra Interface có hợp lệ hay không

Để có thể hiểu rõ hơn quá trình đối chiếu trong Matching thì bạn hãy nhìn vào hình bên dưới:



Hình 3.9: Kiểm tra tính hợp lệ của Interface

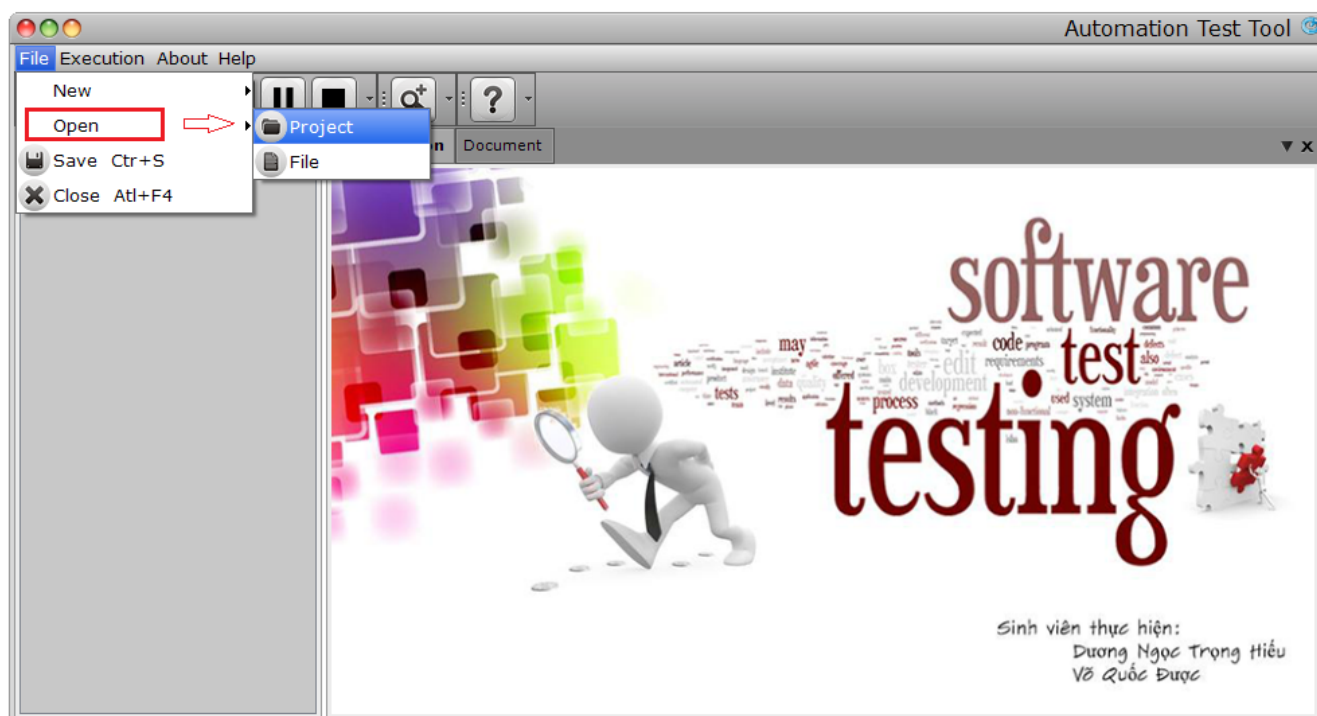
Ở cửa sổ Windows View là đối tượng được chọn, cụ thể là Caculator. Cửa sổ Matching là những Control được định nghĩa trong Interface. Đối với button 7 của Caculator được xác định thông qua tiêu chí là AutomatonId : 137 thì tương ứng ở cửa sổ Macthing cũng định nghĩa btn7 như vậy. Nếu đúng cả 2 sẽ được tô màu xanh là các controls khác tương tự. Vì vậy người dùng có thể dễ dàng kiểm tra tính hợp lệ của Interface một cách dễ dàng.

3.5 Hướng Dẫn Sử Dụng

Mở, tạo một Project

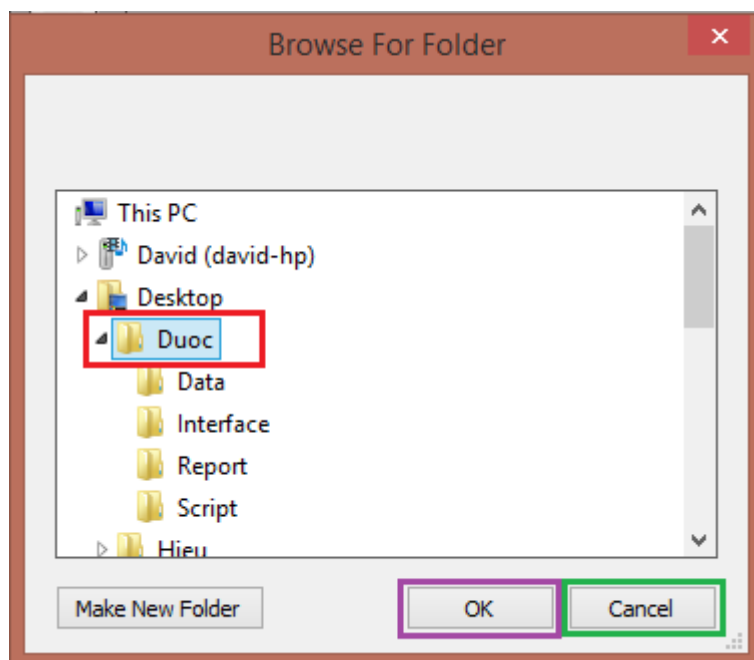
Để mở một Project, người dùng làm theo các bước sau:

Mở chương trình lên, Chọn File →Open→Project.



Hình 3.10: Open Project

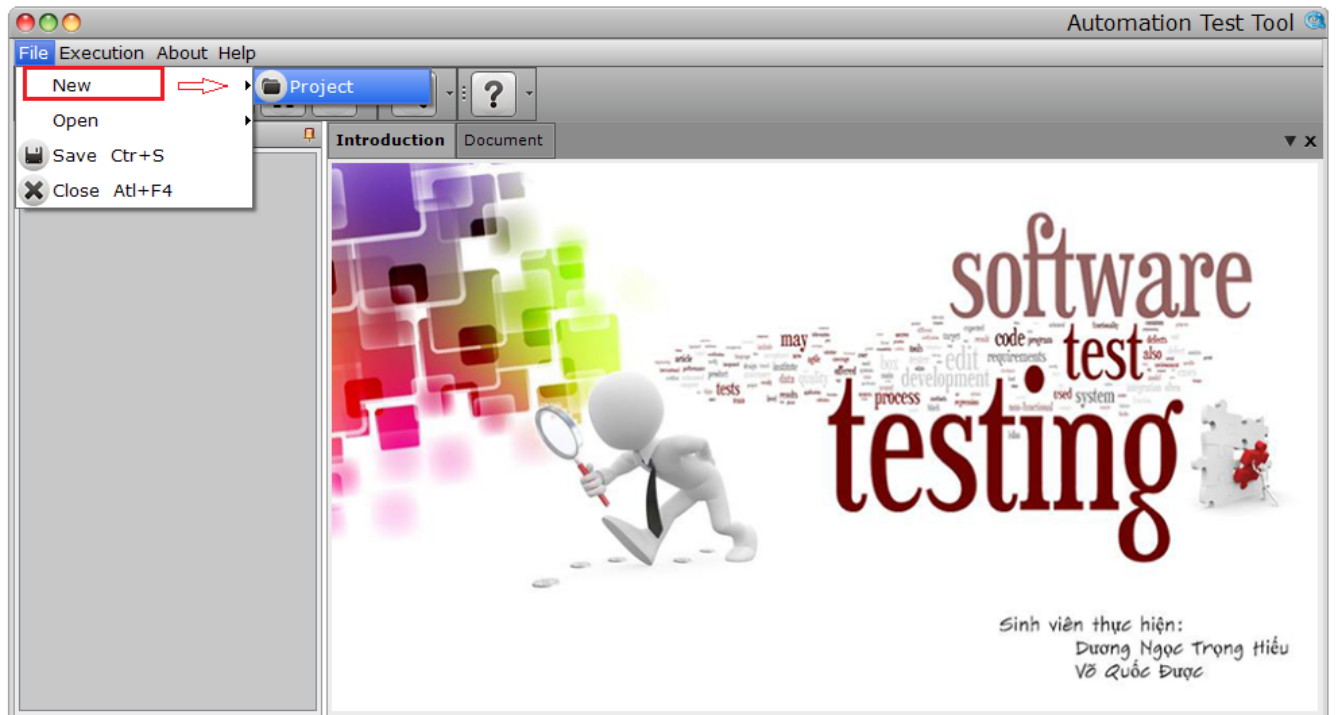
Chọn thư mục chứa Project cần mở chọn OK để xác nhận và Cancel để hủy chọn:



Hình 3.11: Browse for Folder

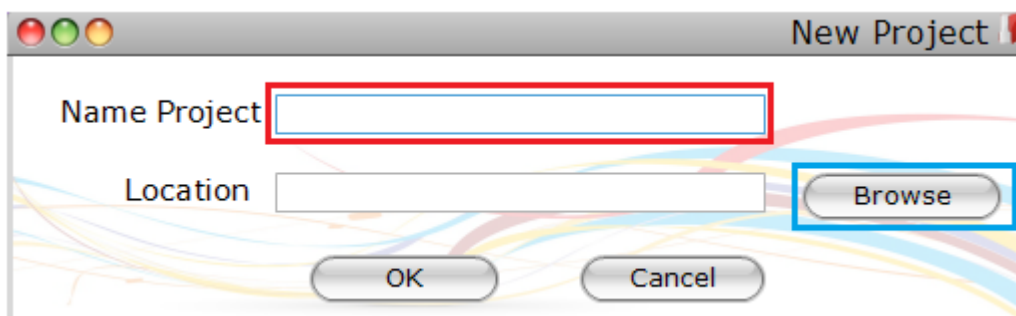
Để tạo mới một Project, người dùng làm theo các bước sau:

Chọn File → Open → Project.



Hình 3.12: Create Project

Chương trình sẽ hiển thị form New Project, người dùng nhập tên Project và Browse đến thư mục làm việc của Project, Chọn OK để đồng ý tạo Project mới và Cancel để hủy thao tác.



Hình 3.13: New Project

Thực thi chương trình

Sau khi tạo xong Project, bạn sẽ được giao diện chương trình như sau:



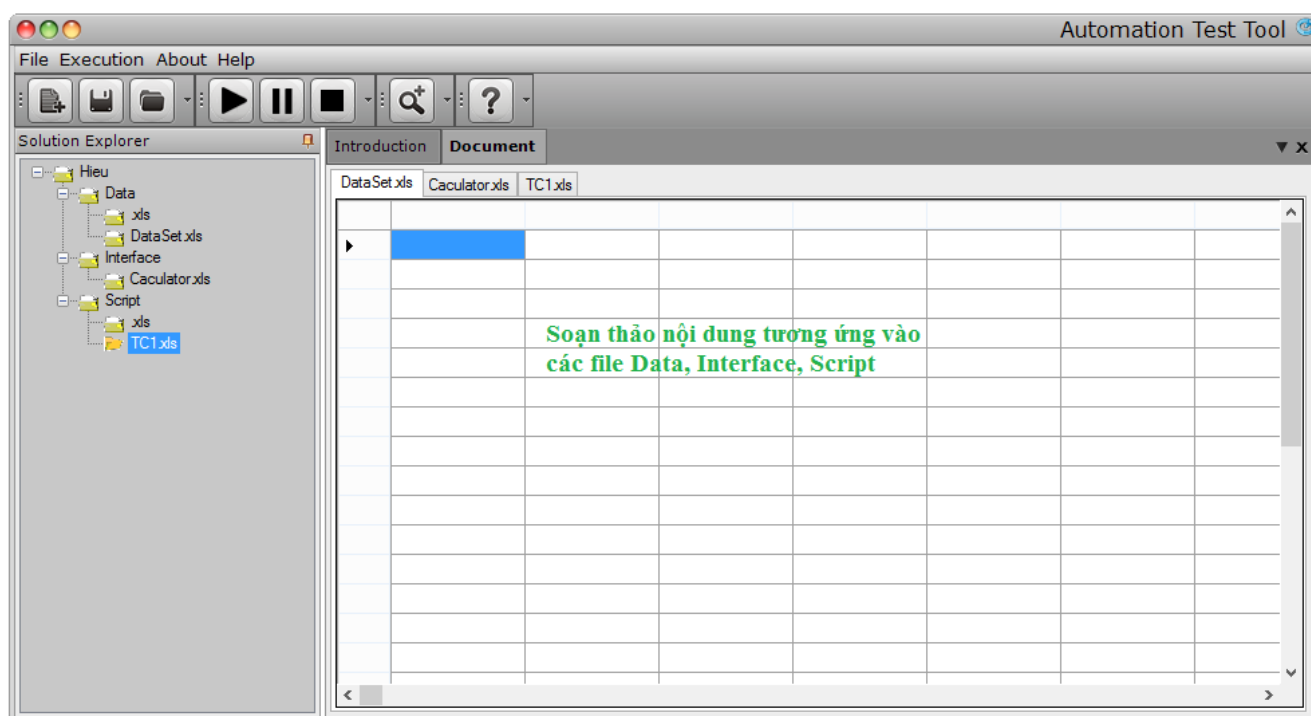
Hình 3.14: Giao diện chương trình

Click chuột phải vào Data →Add: Để tạo mới một file Data.

Click chuột phải vào Interface →Add: Để tạo mới một file Interface.

Click chuột phải vào Script →Add: Để tạo mới một file Script.

Bây giờ, bạn sẽ nhập nội dung vào Data, Interface, Script để hoàn thành quá trình Test một phần mềm:



Hình 3.15: Soạn thảo nội dung

Để chạy Test Case bạn có hai cách sau đây:

- Chọn file Test Case cần chạy → Execution → Run

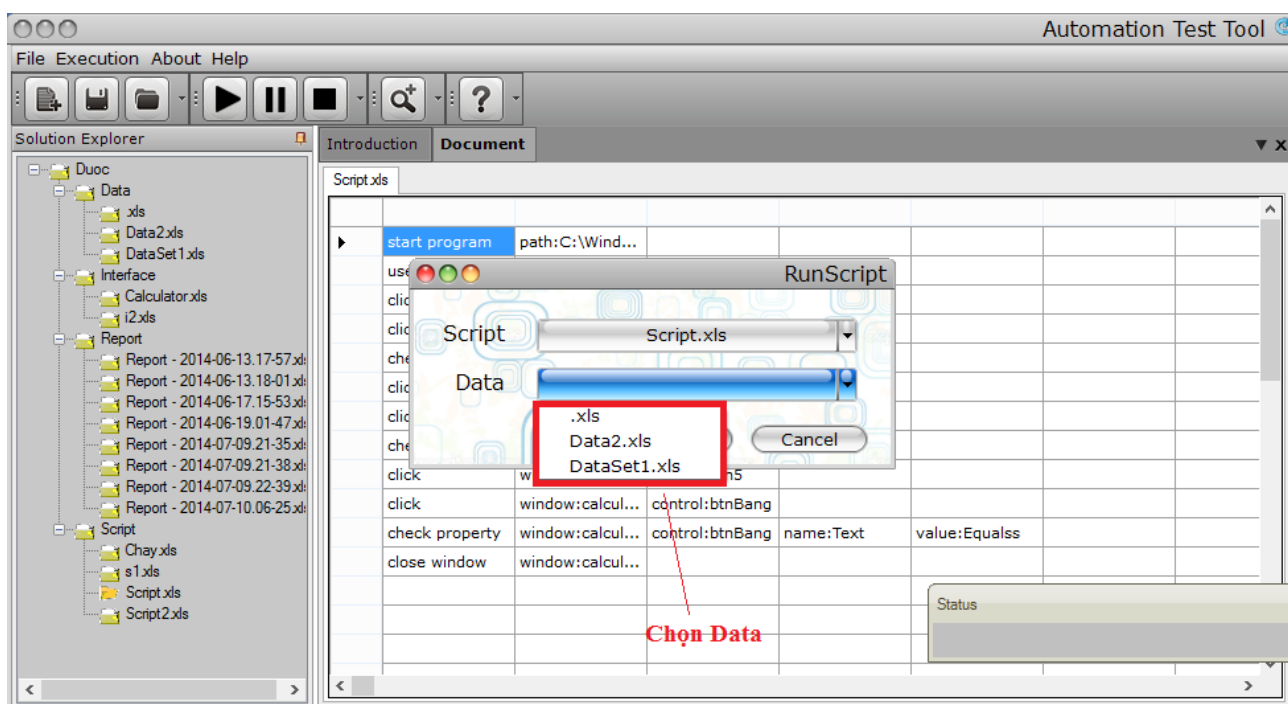
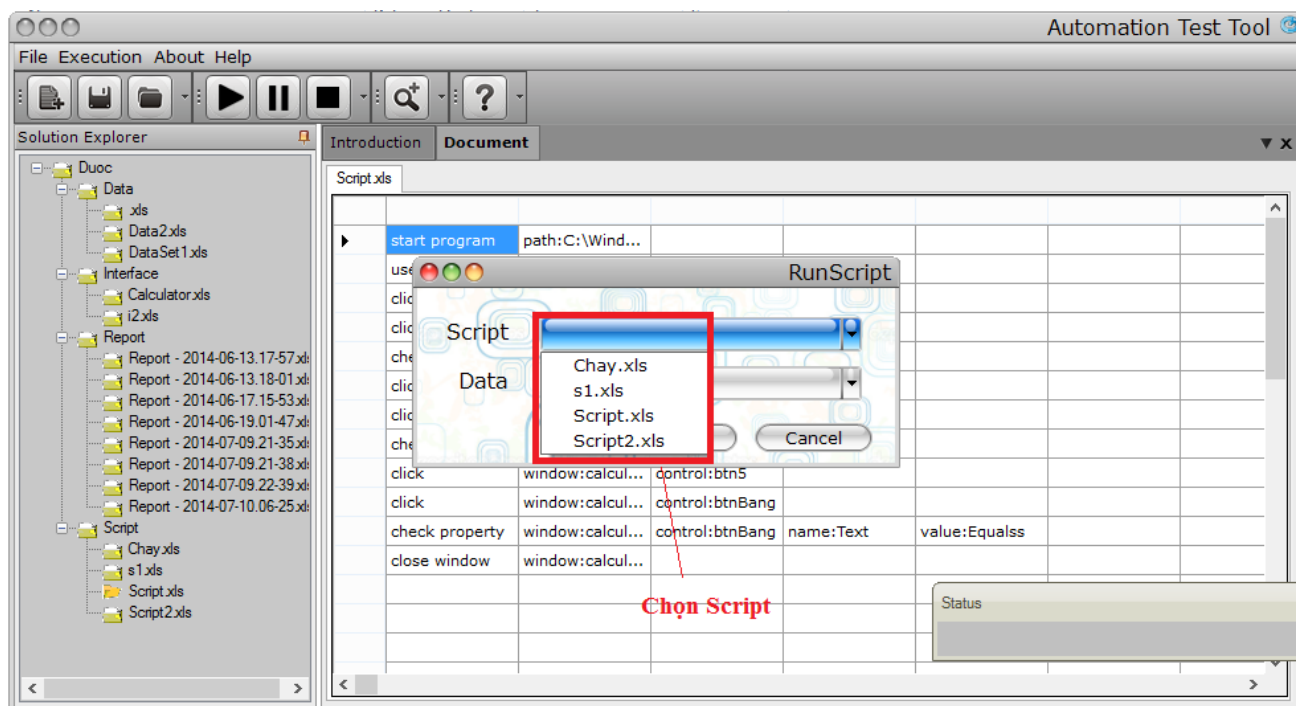
- Chọn file Test Case cần chạy → Click vào biểu tượng Run



Ngoài ra còn có các chức năng Pause, Stop để tạm dừng và kết thúc quá trình Test, bạn cũng thực hiện như trên.

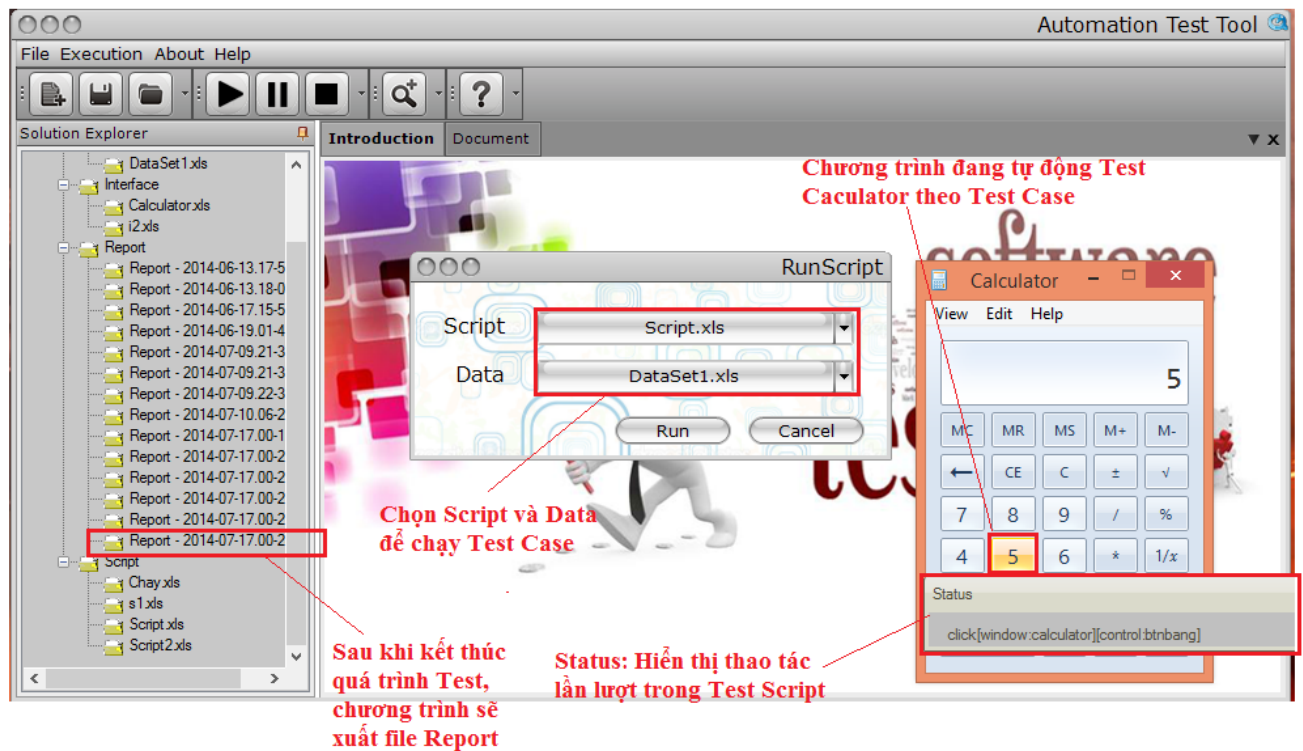
Sau khi chọn Run → Chương trình sẽ hiện form Run, ở đây lần lượt bạn chọn Test Script và Data (nếu cần Test với bộ dữ liệu).

Hình 3.16: Chọn file Script để chạy chương trình



Hình 3.17: Chọn file Data cho quá trình Test

Làm theo các bước hướng dẫn trên, bạn chọn OK để chạy chương trình và Cancel để hủy thao tác. Và đây là hình ảnh chương trình chạy quá trình Test.

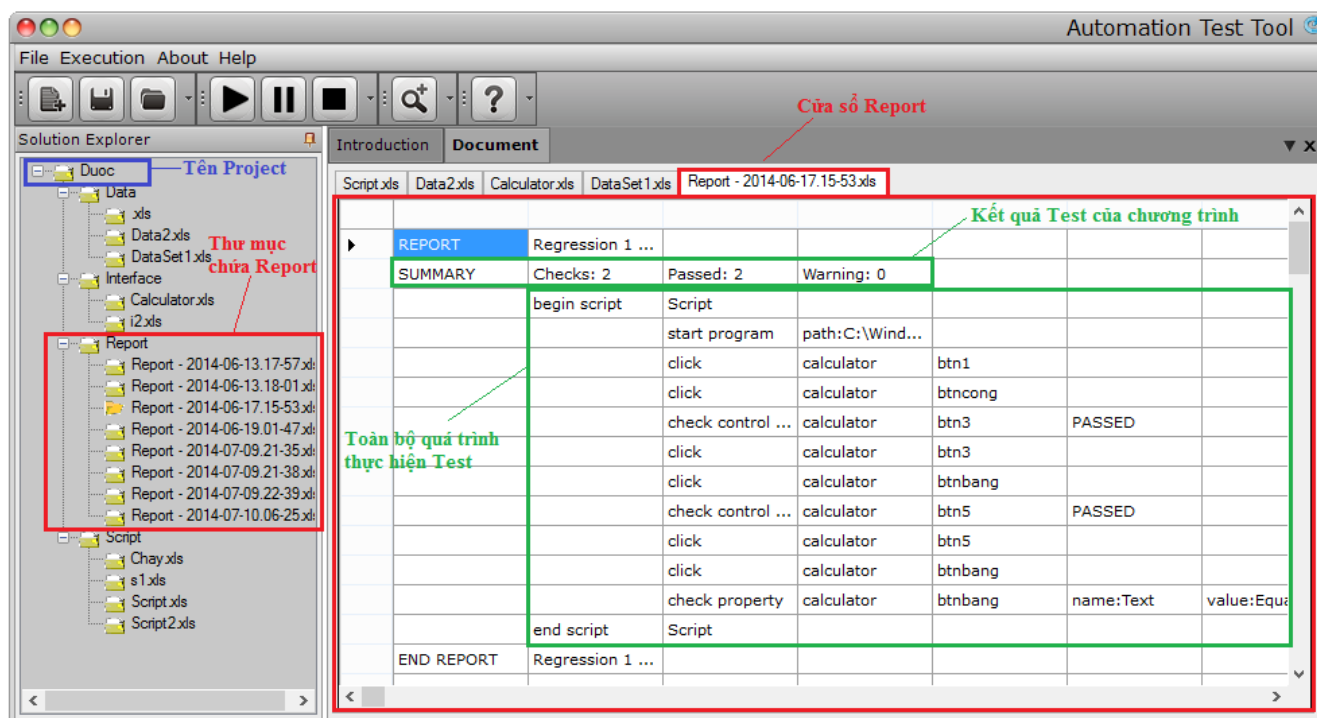


Hình 3.18: Chạy chương trình

Kết quả cuối cùng là file Report, bạn có thể xem và đánh giá quá trình kiểm thử có đạt yêu cầu hay không.

Report

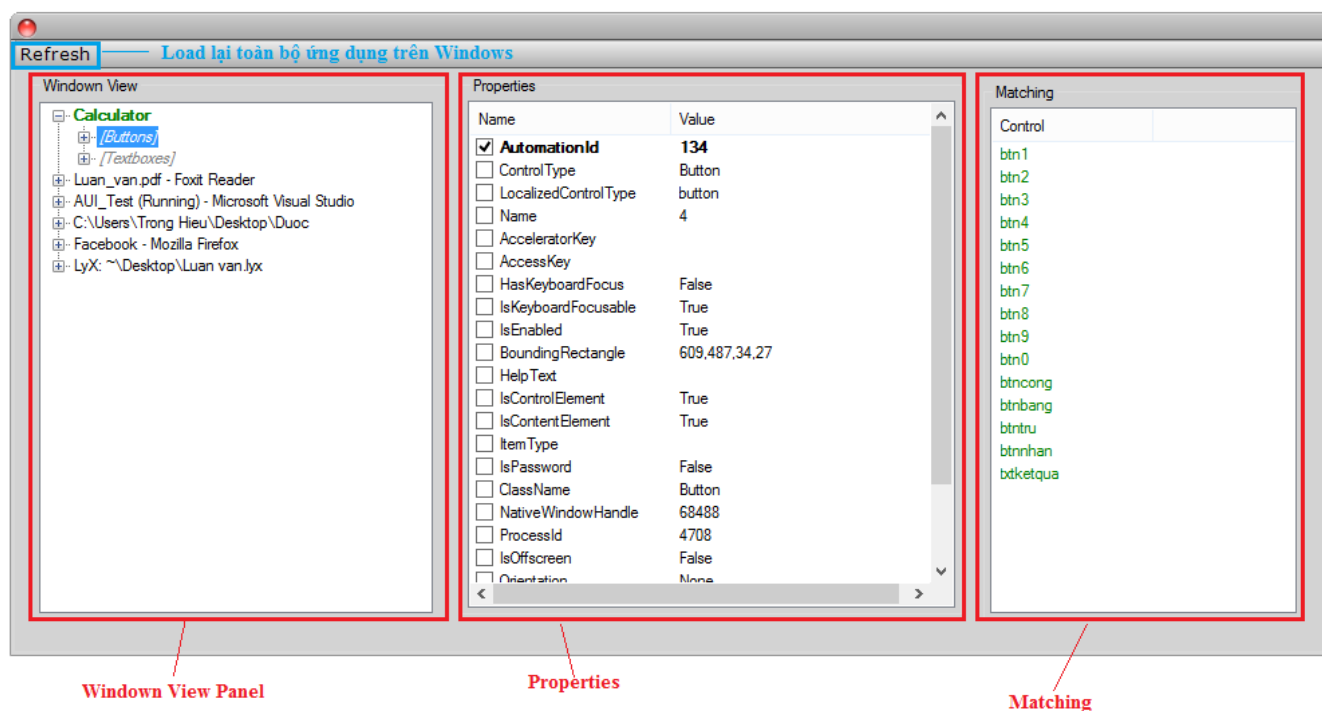
Mục đích cuối cùng của chương trình là đem đến cho người dùng một bản báo cáo kết quả của quá trình thực thi. Dựa vào bản báo cáo để có thể đánh giá đúng tính chất của quá trình Test có thật sự hiệu quả hay không. Dưới đây là một ví dụ về file báo cáo mà chúng tôi thực hiện.



Sử dụng công cụ Spy

Để dùng công cụ Spy bạn click vào biểu tượng Spy trên thanh ToolBar.

Chương trình sẽ mở thêm cửa sổ mới của Spy.



Nhấp chọn chương trình cần Test trong khung Window View Panel, cửa sổ Properties sẽ

hiện tất cả các thuộc tính của chương trình đó đây chính là các tiêu chí bạn có thể chọn để định nghĩa trong file Interface.

Chương 4

Tổng Kết

Đề tài: "Phần mềm kiểm thử các ứng dụng Desktop chạy trên .Net" đã phần như đã đáp ứng được những yêu cầu chính yếu nhằm phục cho nhu cầu kiểm thử. Trải qua thời gian dài nghiên cứu, phát triển và sự hỗ trợ hết sức tận tình của thầy Nguyễn Đạt Thông, Chúng tôi đã xây dựng được những chức năng cơ bản sau:

- Chức năng soạn thảo: Với giao diện soạn thảo đơn giản tiện lợi, hiển thị dạng lưới hỗ trợ cho các kiểm thử viên nhanh chóng soạn thảo Test Case.
- Chức năng thực thi: Bao gồm Run, Pause, Stop cho phép người dùng chạy một Test Script bất kì sau khi được chọn. Ngoài ra kiểm thử viên còn có thể can thiệp vào quá trình Test bằng cách chọn Pause để tạm dừng, Stop dừng hẳn quá trình kiểm thử. Chương trình có hỗ trợ các phím tắt cho các tổ hợp chức năng này, vào phần Help để biết thêm chi tiết.
- Chức năng báo cáo kết quả: Một trong những chức năng quan trọng của kiểm thử, dựa vào file báo cáo kiểm thử viên có thể biết được chi tiết quá trình Test và những thống kê Checked, Passed tương ứng với từng tiêu chí được chọn Test. File báo cáo được định dạng ".xls".
- Công cụ tìm kiếm tiêu chí (Spy): Giúp tìm kiếm những tiêu chí đặc trưng để xác định được đối tượng cần Test, giúp người dùng dễ dàng định nghĩa và đối chiếu đối tượng trong Interface. Spy là công cụ hỗ trợ đắc lực cho quá trình kiểm thử.
- Chức năng quản lý tác vụ: Cho phép thêm, xóa, sửa những file quan trọng của chương trình là Data, Interface, Script. Đây là tính năng cơ bản mà chương trình phải có.
- Chức năng sao lưu dữ liệu: Cho phép người dùng lưu trữ file Data, Interface, Script dưới dạng chuẩn ".xls"

Ngoài ra chương trình còn một số chức năng khác hỗ trợ tốt cho người dùng trong quá trình kiểm thử.

Những ưu nhược điểm của chương trình:

Chương trình có giao diện thân thiện giúp người dùng dễ dàng tương tác. Hỗ trợ kiểm thử được cho hầu hết các ứng dụng chạy trên Desktop. Được phát triển dựa trên công nghệ kiểm thử tự động Coded UI kết hợp phương pháp Action Base Testing giúp người dùng dễ quản lý trong mô hình : Data, Interface, Script. Ngoài những điểm mạnh thì bên cạnh đó có những nhược điểm và chức năng chúng tôi chưa kịp phát triển. Chương trình chỉ mới là sản phẩm nhỏ chưa thể ứng dụng lên một hệ thống lớn như Server. Chưa thể cho người dùng định nghĩa các Action và tạo hàm để gọi sử dụng cho những lần kiểm thử kế tiếp, và còn nhiều hạn chế khác nữa. Trong tương lai, chúng tôi sẽ còn tiếp tục nuôi dưỡng và hiện thực hóa những ý tưởng của mình. Chúng tôi sẽ hoàn thiện dần những tính năng còn thiếu và sẽ xây dựng thành một hệ thống lớn có thể ứng dụng lên Server.

Chúng tôi mong muốn rằng, với “Phần mềm kiểm thử các ứng dụng Desktop chạy trên .Net” sẽ khiến người dùng có đánh giá và cách nhìn khác về phần mềm kiểm thử mã nguồn mở và quan trọng hơn hết đó là giúp người dùng tiết kiệm thời gian tiền bạc cho kiểm thử các ứng dụng trên Desktop. Đặc biệt là cung cấp cho các bạn sinh viên đang có ý định theo chuyên ngành kiểm thử một công cụ học tập hiệu quả, chất lượng, thực tế, miễn phí trong thời buổi kinh tế khó khăn này.

Cuối cùng, chúng tôi hi vọng phần mềm sẽ nhận được sự quan tâm, góp ý, xây dựng của mọi người.

Tài Liệu Tham Khảo

- [1] Pekka Laukkanen, Data-Driven and Keyword-Driven Test Automation Framework, 2006.
- [2] Hans Buwalda, Experiences with Action Base Testing, 2002.
- [3] LogiGear Conglomerate, Action base Testing, 2006.
- [4] Luận văn: Hệ thống quản lí dự án kiểm thử tự động các ứng dụng web - Hồ Hoài Công Nguyễn & Ngọc Minh Khương 2013
- [5] Cùng một số thông tin có được trên website: Wikipedia và <http://google.com.vn>.