

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

TRẦN THỊ THÚY HẰNG

**PHƯƠNG PHÁP KIỂM THỬ TỰ ĐỘNG TƯƠNG TÁC
GIAO DIỆN NGƯỜI DÙNG CHO ỨNG DỤNG WEB**

LUẬN VĂN THẠC SĨ

Ngành: Công Nghệ Thông Tin

HÀ NỘI – 2016

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

TRẦN THỊ THÚY HẰNG

**PHƯƠNG PHÁP KIỂM THỬ TỰ ĐỘNG TƯƠNG TÁC
GIAO DIỆN NGƯỜI DÙNG CHO ỨNG DỤNG WEB**

Ngành: Công Nghệ Thông Tin

Chuyên ngành: Kỹ Thuật Phần Mềm

Mã số: 60 48 01 03

LUẬN VĂN THẠC SĨ

Ngành: Công Nghệ Thông Tin

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS. TS. Phạm Ngọc Hùng

HÀ NỘI – 2016

**VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

TRAN THI THUY HANG

**A METHOD FOR AUTOMATED GUI TESTING OF
WEB APPLICATIONS**

THE MS. THESIS

Major: Information Technology

Supervisor: Assoc. Prof. Dr. Pham Ngoc Hung

HANOI - 2016

MỤC LỤC

MỤC LỤC.....	i
LỜI CẢM ƠN	iii
TÓM TẮT	iv
ABSTRACT.....	v
LỜI CAM ĐOAN.....	vi
DANH MỤC THUẬT NGỮ VIẾT TẮT	vii
DANH MỤC HÌNH VẼ.....	viii
DANH MỤC BẢNG	x
Chương 1: Giới thiệu	1
Chương 2: Tổng quan về kiểm thử phần mềm tự động.....	3
2.1 Kiểm thử phần mềm tự động	3
2.2 Các phương pháp kiểm thử tự động	4
2.2.1 Các mức độ kiểm thử tự động	4
2.2.2 Kiểm thử tương tác giao diện người dùng	5
2.3 Kiểm thử tự động dựa trên mô hình	8
Chương 3: Phương pháp đặc tả tương tác giao diện cho các ứng dụng Web	9
3.1 Phương pháp xây dựng mô hình cho toàn bộ ứng dụng Web	9
3.2 Đặc tả tương tác giao diện của từng trang Web bằng ô-tô-mát hữu hạn trạng thái	1
3.3 Xây dựng mô hình đặc tả tương tác giao diện cho toàn bộ ứng dụng Web	3
3.4 Ví dụ minh họa cho đặc tả trang Web	3
3.3.1 Xây dựng ô-tô-mát hữu hạn trạng thái M_1	5
3.3.2 Ghép nối ô-tô-mát hữu hạn trạng thái M_1 và M_2	7
3.5 Biểu diễn mô hình đặc tả dưới dạng các tệp tin MS Excel.....	9
Chương 4: Sinh và thực thi các ca kiểm thử tự động	24
4.1 Sinh các ca kiểm thử từ mô hình đặc tả hình thức	24
4.1.1 Đường dẫn kiểm thử	24
4.1.2 Thuật toán sinh tự động các đường dẫn kiểm thử	24
4.2 Thực hiện các ca kiểm thử	27

Chương 5: Công cụ và thực nghiệm.....	28
5.1 Giới thiệu các công cụ hỗ trợ.....	28
5.1.1. Giới thiệu Selenium và một số API WebDriver được sử dụng.....	28
5.1.2. Công cụ JFLAP.....	34
5.2 Giới thiệu công cụ kiểm thử tự động tương tác giao diện cho các ứng dụng Web.....	39
5.2.1 Kiến trúc của công cụ.....	40
5.2.2 Đầu vào của công cụ.....	41
5.2.3 Giao diện và cách sử dụng công cụ ATWA.....	48
5.2.4 Đầu ra của công cụ.....	50
5.2.5 Thực nghiệm.....	53
5.2.6 Kết quả áp dụng và cải tiến công cụ.....	68
5.2.7 Ý nghĩa của công cụ thực nghiệm.....	71
Chương 6: KẾT LUẬN.....	73
TÀI LIỆU THAM KHẢO.....	75

LỜI CẢM ƠN

Trước tiên tôi xin gửi lời cảm ơn chân thành và sâu sắc đến thầy giáo PGS.TS Phạm Ngọc Hùng - người đã trực tiếp hướng dẫn, khuyến khích, chỉ bảo và đóng góp những ý kiến quý báu trong suốt quá trình tôi học tập, nghiên cứu cũng như từ khi tôi bắt đầu nghiên cứu đề tài đến khi hoàn thành luận văn này.

Tôi xin chân thành cảm ơn các thầy cô giáo khoa Công nghệ thông tin, trường Đại học Công nghệ, Đại học Quốc Gia Hà Nội đã tận tình đào tạo, cung cấp cho tôi những kiến thức vô cùng quý giá, đã tạo điều kiện tốt nhất cho tôi trong suốt quá trình học tập, nghiên cứu tại trường.

Đồng thời tôi xin chân thành cảm ơn những người thân trong gia đình cùng toàn thể bạn bè, đồng nghiệp đã luôn giúp đỡ, động viên tôi trong những lúc gặp phải khó khăn trong việc học tập và nghiên cứu.

Cuối cùng, tôi xin chân thành cảm ơn Lê Khánh Trình và Lê Thị Phượng người đã giúp đỡ, tạo điều kiện cho tôi nghiên cứu công cụ kiểm thử tự động ATWT và các đồng nghiệp của tôi tại Công ty Phần Mềm FPT đã tạo điều kiện thuận lợi cho tôi học tập và nghiên cứu chương trình thạc sĩ tại Đại học Công nghệ, ĐHQGHN.

TÓM TẮT

Luận văn tập trung nghiên cứu phương pháp kiểm thử tự động tương tác giao diện của các ứng dụng Web. Phương pháp này là một trong những phương pháp kiểm thử đang được áp dụng và sử dụng ngày càng rộng rãi trong việc kiểm thử các sản phẩm phần mềm nói chung cũng như ứng dụng Web nói riêng. Phương pháp kiểm thử tự động tương tác giao diện người dùng đã được đề xuất và cải tiến bởi một số tác giả với ý tưởng chính là đặc tả tương tác người dùng của từng trang Web bằng máy hữu hạn trạng thái. Mô hình đặc tả hành vi của cả Website sẽ được xây dựng bằng cách ghép nối các mô hình của các trang Web. Sau đó, phương pháp này sử dụng thuật toán sinh các đường dẫn kiểm thử (test paths) một cách tự động dựa trên mô hình của Website sao cho các đường dẫn kiểm thử này bao phủ mọi trường hợp của hệ thống. Tuy nhiên, phương pháp này mới được áp dụng cho phạm vi hệ thống đơn giản và còn có những hạn chế nhất định trong đó có việc xây dựng tự động cho bộ đầu vào. Luận văn tập trung nghiên cứu và áp dụng công cụ vào ứng dụng Web tại công ty và cải tiến công cụ để thực hiện tự động sinh bộ đầu vào nhằm tiến tới mục tiêu tự động hóa một cách hoàn toàn. Phương pháp đề xuất và công cụ được áp dụng tại các giai đoạn kiểm thử chấp nhận và áp dụng cho mô hình Agile. Kết quả thực nghiệm cho thấy tiềm năng ứng dụng của công cụ này trong việc kiểm thử tự động giao diện cho các ứng dụng Web.

***Từ khóa:** Kiểm thử tự động, tương tác hành vi người dùng, máy hữu hạn trạng thái, đường dẫn kiểm thử*

ABSTRACT

This thesis researches a automated GUI testing of Web applications. This method is one of the test methods are being applied and more widely in testing software product and also in testing Web application. This method has been proposed by some author with main idea is the model of each Web page of the Website under testing which describes the user interactions is represented by a finite state machine. The model that describe the behaviors of the whole Website then is constructed by composing the models of all Web pages. After that, the proposed method uses an algorithm to generates automatically test paths based on the compositional model of the Website so that these test paths cover all possible interactions of the system. However, proposed method has been developed and applied to test on a simple systems, and remains certain limited include develop automation for testing input data. This thesis have been applied successful and improve automation test tool on the Web Applications of company and generate test input automatically. Proposed methods and this tool has been applied in the acceptance testing stage of Agile model. The experimental results show the potential application of this tool for automated GUI testing of Web applications in practice.

Keywords: *Automated GUI testing, user interaction behavior, finite state machine, test paths, Web application*

LỜI CAM ĐOAN

Tôi xin cam đoan rằng luận văn thạc sĩ công nghệ thông tin “Phương pháp kiểm thử tự động tương tác giao diện người dùng cho các ứng dụng Web” là công trình nghiên cứu của riêng tôi, không sao chép lại của người khác. Trong toàn bộ nội dung của luận văn, những điều đã được trình bày hoặc là của chính cá nhân tôi hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các nguồn tài liệu tham khảo đều có xuất xứ rõ ràng và hợp pháp.

Tôi xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan này.

Hà Nội, ngày 24 tháng 03 năm 2016

Trần Thị Thúy Hằng

DANH MỤC THUẬT NGỮ VIẾT TẮT

STT	Từ viết tắt	Từ đầy đủ	Ý nghĩa
1	API	Application Programming Interface	Giao diện lập trình ứng dụng
2	FSA	Finite State Automaton	Ô-tô-mát hữu hạn trạng thái
3	FSM	Finite State Machine	Máy hữu hạn trạng thái
4	MBT	Model- base testing.	Kiểm thử dựa trên mô hình.
5	ATWA	Automation Testing Web Application	Công cụ kiểm thử tự động cho ứng dụng Web
6	GUI	Graphical User Interface	Giao diện tương tác người dùng
7	OCR	Optical Character Recognition	Nhận dạng kí tự quang học
8	UML	Unified Modeling Language	Ngôn ngữ mô hình hóa thống nhất

DANH MỤC HÌNH VẼ

Hình 3.1.	Tài liệu thiết kế màn hình của dự án.....	0
Hình 3.2.	Minh họa ô-tô-mát hữu hạn trạng thái của toàn ứng dụng Web	1
Hình 3.3.	Menu chính sau khi đăng nhập vào hệ thống	4
Hình 3.4.	Trạng thái bắt đầu của Danh sách Tổ Chức (Organisation List).....	5
Hình 3.5.	Trạng thái Chi tiết về Tổ Chức (Organisation Details)	5
Hình 3.6.	Ô-tô-mát hữu hạn trạng thái M_1	6
Hình 3.7.	Thông tin Chức Năng <i>Organisation Details - Tab Information</i>	7
Hình 3.8.	Ô-tô-mát hữu hạn trạng M_2	9
Hình 3.9.	Mô hình M sau khi thực hiện thuật toán ghép nối giữa M_1 và M_2	16
Hình 5.1.	Cấu trúc của Selenium	29
Hình 5.2.	Kiến trúc của công cụ Auto Testing Web Application (ATWA).....	40
Hình 5.3.	Ví dụ về một FA với cách định nghĩa như mục a.....	45
Hình 5.4.	Xuất ra tệp tin excel.	45
Hình 5.5.	Tệp tin excel sau khi được xuất từ công cụ JFLAP	46
Hình 5.6.	Tệp tin excel đầu vào sau khi được điền giá trị kiểm thử	47
Hình 5.7.	Lưu trữ các tệp tin đầu vào.....	48
Hình 5.8.	Giao diện nhập dữ liệu đầu vào của công cụ.....	49
Hình 5.9.	Chọn bộ kiểm thử để thực hiện.....	49
Hình 5.10.	Selenium Webdriver thực hiện kiểm thử tự động trên Web	50
Hình 5.11.	Kết quả hiển thị sau khi chạy xong bộ kiểm thử.....	51
Hình 5.12.	Ví dụ về hoạt động của công cụ ATWA.....	52
Hình 5.13.	Kết quả kiểm thử.....	54
Hình 5.14.	Ứng dụng Web quản lý thông tin cán bộ.	55
Hình 5.15.	Giao diện trang đăng nhập.....	56
Hình 5.16.	Danh sách các chức năng (Menu List).....	56
Hình 5.17.	Giao diện các chức năng của Organisation.	57
Hình 5.18.	Chức năng tìm kiếm theo bảng chữ cái.....	57
Hình 5.19.	Chức năng sắp xếp Organisation	58
Hình 5.20.	Giao diện các chức năng của Organisation.	58

Hình 5.21.	Giao diện phần Service Features	59
Hình 5.22.	Giao diện phần List Product	59
Hình 5.23.	Giao diện phần Premise Detail	59
Hình 5.24.	Giao diện phần Materials	60
Hình 5.25.	Giao diện phần Bu Derectorate	60
Hình 5.26.	Mô hình ô-tô-mát hữu hạn trạng thái trang Login	61
Hình 5.27.	Mô hình ô-tô-mát hữu hạn trạng thái chức năng Organisation Details ...	61
Hình 5.28.	Mô hình ô-tô-mát hữu hạn trạng thái chức năng Organisation Details - Tab Infomation	62
Hình 5.29.	Thư mục các tệp tin đặc tả chức năng Organisation	63
Hình 5.30.	Giao diện của công cụ	63
Hình 5.31.	Kết quả thực hiện đường dẫn kiểm thử hiển thị trong tệp tin đầu ra	67
Hình 5.32.	Thực hiện kiểm thử qua từng giai đoạn theo mô hình Agile	69
Hình 5.33.	Tạo bộ kiểm thử qua từng sprint	70
Hình 5.34.	Sinh đầu vào từ mô hình theo [3]	70
Hình 5.35.	Cải tiến sinh đầu vào từ mô hình từ đề xuất của [3]	71

DANH MỤC BẢNG

Bảng 3.1.	Các trạng thái Web của trang Danh sách Tổ Chức (<i>Organisation List</i>)	6
Bảng 3.2.	Các sự kiện của trang Danh sách Tổ Chức	6
Bảng 3.3.	Bảng các phần tử Web của trang <i>Organisation Details - Tab Information</i> 17	
Bảng 3.4.	Ý nghĩa của từng cột trong Bảng <i>Element_html</i>	17
Bảng 3.5.	Bảng các trạng thái của trang <i>Organisation Details - Tab Information</i> ...	18
Bảng 3.6.	Ý nghĩa của các cột trong bảng trạng thái.....	18
Bảng 3.7.	Bảng các sự kiện của trang <i>Organisation Details - Tab Information</i>	19
Bảng 3.8.	Ý nghĩa của từng cột trong bảng Event (sự kiện).....	20
Bảng 3.9.	Bảng các transition của trang <i>Organisation Details - Tab Information</i> ..	20
Bảng 3.10.	Tệp tin Excel đặc tả trang Web <i>Organisation Details - Tab Information</i> 21	
Bảng 3.1.	Các trạng thái Web của trang Danh sách Tổ Chức.....	6
Bảng 3.2.	Các sự kiện của trang Danh sách Tổ Chức	6
Hình 3.7.	Ô-tô-mát hữu hạn trạng thái M_1	6
Hình 3.8.	Thông tin Chức Năng <i>Organisation Details - Tab Information</i>	7
Hình 3.9.	Ô-tô-mát hữu hạn trạng M_2	9
Hình 3.10.	Mô hình M sau khi thực hiện thuật toán ghép nối giữa M_1 và M_2	16
Bảng 5.1.	Các thao tác lên phần tử Web.....	32
Bảng 5.2.	Các công cụ trên JFLAP.....	35
Bảng 5.3.	Bảng <i>Element_html</i>	42
Bảng 5.4.	Bảng Sate (bảng trạng thái)	43
Bảng 5.5.	Bảng Event (bảng sự kiện)	43
Bảng 5.6.	Bảng Transition (Sự chuyển trạng thái).....	44
Bảng 5.7.	Các trường hợp thất bại FAIL	50
Bảng 5.8.	Bảng chức năng chính của trang Web FPT Services.....	53
Bảng 5.9.	Chức năng chính của Organisation.....	55
Bảng 5.10.	Các transition của trang <i>Organisation Details - Tab Information</i>	64

Bảng 5.11. Các test path (đường dẫn kiểm thử) được sinh ra từ mô hình trang <i>Organisation Details - Tab Information</i>	65
--	----

Chương 1: Giới thiệu

Hiện nay, tự động hóa được ứng dụng trong rất nhiều lĩnh vực, mục đích thường đa dạng và tùy theo nhu cầu của từng lĩnh vực. Tuy nhiên, điểm chung nhất của giải pháp này là hướng đến giảm nhân lực, thời gian và nâng cao chất lượng của quá trình kiểm thử [1]. Trong quá trình phát triển phần mềm, đã có rất nhiều các công cụ kiểm thử được áp dụng. Nhiều phương pháp cũng được áp dụng cho từng giai đoạn với mục đích chính là giảm thiểu việc sai sót do kiểm thử sản phẩm một cách thủ công, tránh việc lặp đi lặp lại một động tác, gây nhàm chán từ đó xảy ra việc kiểm thử thiếu.

Các ứng dụng Web được phát triển một cách mạnh mẽ nhưng yêu cầu của khách hàng đặt ra dường như chưa đáp ứng được. Câu hỏi quan trọng trong phát triển phần mềm đặt ra là: “Làm thế nào để đảm bảo được chất lượng của các ứng dụng Web”. Kiểm thử gần như là phương pháp duy nhất để đảm bảo chất lượng cho các sản phẩm phần mềm trong các công ty phần mềm hiện nay. Giai đoạn kiểm thử là giai đoạn chiếm mất rất nhiều công sức và tiền của, chi phí cho giai đoạn này cũng thường chiếm tới 40% tổng các nỗ lực dành cho một dự án phát triển phần mềm. Các công việc kiểm thử thường làm một cách thủ công nhưng vẫn không thể đảm bảo phát hiện ra được hết tất cả các lỗi. Kiểm thử thủ công thường gây cảm giác nhàm chán đặc biệt tại giai đoạn kiểm thử hồi quy. Mỗi khi thực hiện sửa một lỗi nhỏ trong hệ thống, kiểm thử viên phải thực hiện kiểm thử lại toàn bộ hệ thống, công việc lặp đi lặp lại một cách nhàm chán. Các kỹ thuật hay phương pháp kiểm thử tự động được biết đến như là các giải pháp tiềm năng để giải quyết các vấn đề nêu trên. Việc áp dụng kiểm thử tự động khiến chúng ta có thể cắt giảm đi rất nhiều thời gian và chi phí không cần thiết.

Phương pháp kiểm thử tự động tương tác người dùng của ứng dụng Web là một trong những phương pháp kiểm thử tự động đang phổ biến hiện nay. Phương pháp kiểm thử tự động tương tác giao diện người dùng được chia thành ba phương pháp kiểm thử chính: kiểm thử thủ công, kiểm thử bằng cách chụp và phát lại (capture and replay), kiểm thử dựa trên mô hình [4]. Trên thực tế, có rất công cụ kiểm thử tự động áp dụng cho công việc kiểm thử như [5] đã mô tả, tuy nhiên hướng nghiên cứu về kiểm thử dựa trên mô hình cho ứng dụng Web đang là hướng nghiên cứu được nhiều tác giả đề cập và đưa ra nhiều phương pháp nhưng dường như việc kiểm thử chức năng (theo luồng tương tác giao diện người dùng) vẫn chưa có giải pháp thỏa đáng. Một số nghiên cứu trước đã đề xuất các phương pháp và công cụ thực hiện việc kiểm thử chức năng ứng dụng Web [2,3]. Công cụ và phương pháp được đề xuất bởi [3] và cải tiến bởi [2] có ý tưởng chính là đặc tả một cách thủ công máy hữu hạn các trạng thái tương tác người dùng của từng trang Web, sau đó các bản đặc tả máy hữu hạn trạng thái này thông qua JFLAP [8] để sinh ra tệp tin excel làm đầu vào cho công cụ kiểm thử tự động. Phương pháp đưa ra áp dụng thuật toán sinh các đường dẫn kiểm

thử (test paths) một cách tự động dựa trên mô hình của trang Web sao cho các đường dẫn kiểm thử bao phủ mọi trường hợp của hệ thống. Công cụ xây dựng tích hợp Java và Selenium [7,9,10,11] để thực hiện kiểm thử một cách tự động tất cả các đường dẫn kiểm thử được sinh ra. Tuy nhiên, phương pháp này còn có những hạn chế nhất định trong đó có việc xây dựng tự động cho bộ đầu vào. Các dữ liệu đầu vào cho công cụ phải thực hiện thủ công hơn 70% khối lượng công việc, và thường xảy ra sai sót dẫn tới việc thực hiện lặp đi lặp lại nhiều lần một công việc. Một hạn chế trong phương pháp [2,3], phương pháp mới chỉ thực hiện kiểm thử tự động cho một số hệ thống đơn giản, chưa áp dụng thực tế vào dự án theo quy trình phát triển phần mềm, đặc biệt quy trình phát triển phần mềm theo xu hướng hiện nay, quy trình phát triển theo mô hình nhanh - Agile.

Luận văn tập trung nghiên cứu và đưa ra giải pháp để giải quyết vấn đề nêu trên. Từ công cụ đã được đề xuất bởi [3], luận văn nghiên cứu áp dụng và cải tiến công cụ thực hiện tự động sinh bộ đầu vào, sử dụng JFLAP hỗ trợ, nhằm tiến tới mục tiêu tự động hóa một cách hoàn toàn. Ngoài việc cải tiến bộ đầu vào cho công cụ kiểm thử được đề xuất, luận văn áp dụng phương pháp đặc tả mô hình [12] và áp dụng cho hệ thống phức tạp với nhiều loại phần tử Web. Ngoài ra, điểm đặc biệt trong nghiên cứu này, luận văn đã thực hiện áp dụng vào giai đoạn kiểm thử chấp nhận trong mô hình Agile [13] tại công ty FPT Software.

Nội dung của luận văn được trình bày trong năm chương và phần kết luận. Chương 1 giới thiệu về đề tài, chương này trình bày các ngữ cảnh, những lý do chọn đề tài, mục tiêu của đề tài và cấu trúc của luận văn. Chương 2 trình bày về kiểm thử tự động, so sánh giữa kiểm thử tự động và kiểm thử thủ công, các kiểu kiểm thử tự động, nêu lý thuyết của kiểm thử tự động dựa trên mô hình. Chương 3 mô tả phương pháp đặc tả tương tác giao diện cho các ứng dụng web, trong chương này người viết có nêu cách đặc tả, xây dựng mô hình đặc tả, và cách biểu diễn mô hình đặc tả như thế nào. Chương 4 mô tả về việc sinh và thực thi các ca kiểm thử tự động và ví dụ áp dụng. Chương 5 giới thiệu công cụ và trình bày kết quả thực nghiệm vào một ứng dụng Web tại Công Ty Cổ Phần Phần Mềm FPT. Cuối cùng là phần kết luận, định hướng mở rộng và tài liệu tham khảo.

Chương 2: Tổng quan về kiểm thử phần mềm tự động

2.1 Kiểm thử phần mềm tự động

Như chúng ta đã biết, sản phẩm phần mềm được kiểm thử tất nhiên sẽ luôn luôn đảm bảo được chất lượng khi đưa ra môi trường sử dụng. Kiểm thử phần mềm là thực hiện tìm ra lỗi tiềm ẩn trong phần mềm. Nhưng không hẳn là chỉ tìm lỗi. Việc kiểm thử cần phải xem xét đến mức độ hiệu quả, hiệu suất của công việc kiểm thử, phải đảm bảo vừa nhanh chóng nhưng lại tốn ít chi phí nhất có thể.

Kiểm thử tự động đã giải quyết được phần nào đó vấn đề này. Kiểm thử tự động có thể giảm công sức được yêu cầu để kiểm thử, hoặc có thể tăng việc kiểm thử trong một giới hạn cho phép. Kiểm thử tự động có thể chỉ cần thực hiện trong vài phút mà kiểm thử thủ công phải thực hiện trong vài giờ.

Trong thực tế việc kiểm thử tự động và kiểm thử phần mềm thủ công có những ưu và nhược điểm là khác nhau. Kiểm thử phần mềm không phải là công việc dễ dàng và cũng cần phải có kỹ năng. Người kiểm thử chỉ cần tạo một bộ các ca kiểm thử (test cases), thực hiện chúng, quan sát và so sánh với tài liệu chuẩn và bạn đã có thể thực hiện kiểm thử. Tuy nhiên, công việc quan trọng nhất trong kiểm thử đó là cần phải lựa chọn ra bộ các ca kiểm thử nào để có thể tìm ra được nhiều lỗi nhất có thể.

Đối với kiểm thử, chúng ta có thể đưa ra bốn điều quan trọng cần quan tâm để việc kiểm thử đạt được kết quả tốt nhất đó là: Chất lượng của bộ các ca kiểm thử, hiệu quả của việc tìm lỗi, tiết kiệm chi phí, dễ dàng bảo trì. Vì vậy việc kiểm thử chỉ là tìm lỗi là chưa đủ, chúng ta cần phải quan tâm đến các yếu tố làm cách nào để tìm được nhiều lỗi nhưng lại đảm bảo chi phí không vượt quá mức [4].

Kiểm thử tự động cũng cần phải có kỹ năng nhưng kỹ năng để dành cho việc kiểm thử tự động hoàn toàn khác với kiểm thử thủ công. Điều nhận thấy đầu tiên và sự khác biệt đầu tiên đó là về mặt chi phí. Để cảm thấy được lợi ích thực sự của kiểm thử tự động, người dùng cần phải lựa chọn và cài đặt một cách cẩn thận. Chất lượng của kiểm thử tự động độc lập với chất lượng của kiểm thử. Tự động kiểm thử ảnh hưởng duy nhất đến việc làm cách nào để tiết kiệm chi phí và cải tiến hoặc bảo trì. Tuy nhiên, mỗi một lần cài đặt kiểm thử tự động chi phí thường vượt và trội hơn so với việc tạo và bảo trì. Nếu chỉ sử dụng duy nhất một lần công cụ kiểm thử tự động chi phí sẽ rất cao và như vậy kiểm thử tự động lại không mang lại lợi ích. Do đó, đối với công việc phải thực hiện lại nhiều lần, và kéo dài trong nhiều thời gian thì kiểm thử tự động là một lựa chọn tốt và mang lại lợi ích cũng như hiệu quả công việc cao.

2.2 Các phương pháp kiểm thử tự động

2.2.1 Các mức độ kiểm thử tự động

Nói đến kiểm thử, chúng ta sẽ đề cập đến quy trình, đến phương pháp kiểm thử, đến các kiểu kiểm thử. Tuy nhiên, tại nghiên cứu này người viết muốn đưa ra các kiểu Kiểm thử tự động.

Mô hình V-Model đã quá quen thuộc đối với những người làm trong công việc kiểm thử. Mô hình thể hiện một cách rõ nhất các hoạt động của kiểm thử từ giai đoạn kiểm thử mức đơn vị, kiểm thử tích hợp, kiểm thử hệ thống cho đến kiểm thử chấp nhận. Tương ứng với mỗi một công đoạn trong việc phát triển phần mềm là một hoạt động kiểm thử. Với mỗi một mức kiểm thử, chúng ta đều có thể áp dụng kiểm thử tự động. Theo tài liệu [5], dựa trên mô hình vừa nêu chúng ta có thể chia kiểm thử tự động thành ba kiểu:

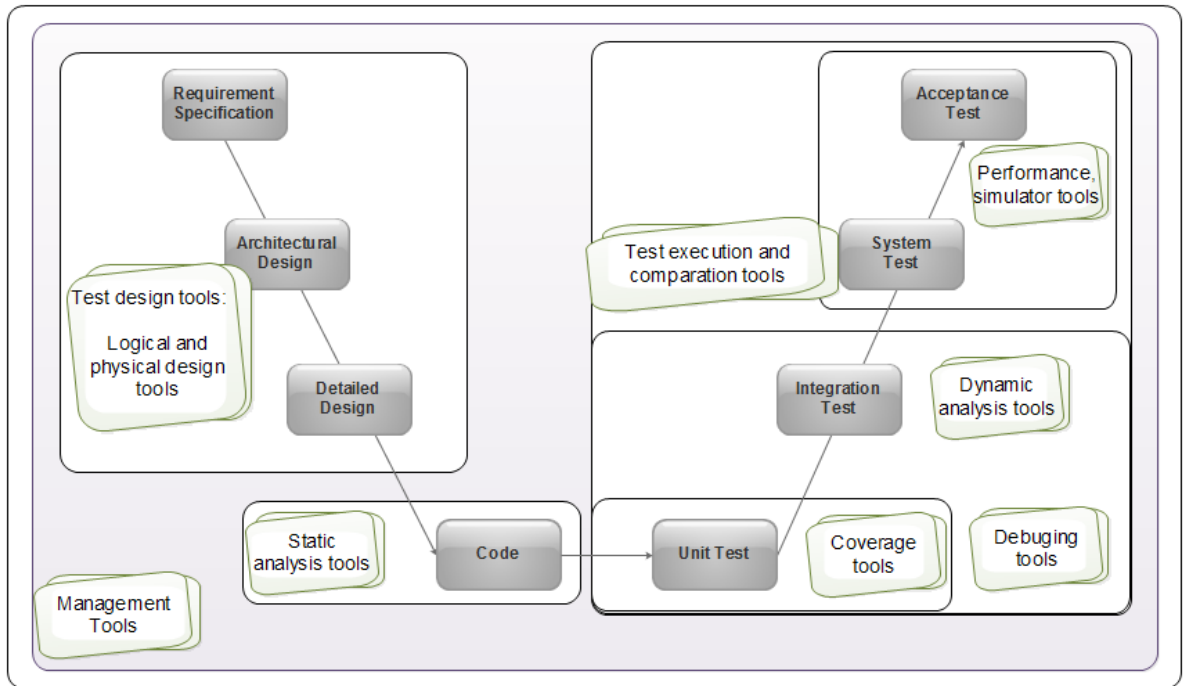
- Kiểm thử mức đơn vị (Unit test)
- Kiểm thử tích hợp (có thể là tích hợp các mô đun hoặc tích hợp hệ thống)
- Kiểm thử giao diện người dùng (kiểm thử chức năng, kiểm thử luồng)

Kiểm thử tự động mức đơn vị (Unit test): Kiểm thử mức đơn vị là kiểm thử mức đầu tiên được thực hiện khi phát triển sản phẩm phần mềm. Công việc kiểm thử mức đơn vị được thực hiện nhanh, tin cậy vì chỉ thực hiện với một phần nhỏ trong mã chương trình, mức đơn vị. Kiểm thử viên thường là lập trình viên, khi tạo ra một hàm hoặc một thủ tục, ngay sau đó lập trình viên thường tạo luôn các ca kiểm thử để thực hiện kiểm thử ngay lập tức mà không cần phải chờ cho đến khi lập trình hoàn thiện sản phẩm. Kiểm thử tự động mức đơn vị chiếm khối lượng các bộ kiểm nhiều nhất và các công cụ thường đa dạng nhất. Các công cụ kiểm thử tự động áp dụng cho mức kiểm thử này: công cụ phân tích mã nguồn, hay công cụ đã đánh giá mức độ bao phủ v.v. (Hình 2.1)

Kiểm thử tự động mức tích hợp (Integration test and System Test): Kiểm thử tích hợp thực hiện những phần mà kiểm thử mức đơn vị chưa bao phủ và chưa được kiểm thử. Kiểm thử tích hợp thường mất nhiều thời gian, chậm, và để thực hiện tự động thường khó, yêu cầu cần phải thực hiện lập trình nhiều hơn so với kiểm thử mức đơn vị. Các công cụ kiểm thử tự động áp dụng cho mức kiểm thử này: phân tích động (dynamic analysis) tự động dò tìm các lỗi do tràn bộ nhớ v.v. (Hình 2.1)

Kiểu kiểm thử tự động tương tác giao diện người dùng: Kiểm thử tương tác giao diện người dùng là kiểm thử tất cả các chức năng và các đường dẫn kiểm thử của ứng dụng. Công việc thực hiện kiểm thử tương tác giao diện thường khó khăn vì có nhiều ràng buộc giữa các thành phần, chức năng. Ví dụ, một số chức năng của ứng dụng phải thực hiện theo một trình tự phức tạp của các sự kiện của giao diện người dùng. Kiểm

thử tự động tương tác giao diện người dùng thường chiếm phần nhỏ hơn so với kiểm thử tự động mức đơn vị và mức kiểm thử tích hợp nhưng không vì thế mà kiểm thử tự động tương tác giao diện người dùng lại không hữu ích, kiểm thử tương tác giao diện người dùng đã có những lợi ích đáng kể.



Hình 2.1. Phân loại các công cụ kiểm thử tự động tương ứng trong vòng đời phát triển phần mềm [4]

2.2.2 Kiểm thử tương tác giao diện người dùng

Giao diện người dùng là phần trung gian giữa người sử dụng và hệ thống. Người sử dụng tương tác với giao diện người dùng để thực hiện các nhiệm vụ. Một giao diện người dùng là một phần quan trọng của tương tác hệ thống. Người dùng có thể dựa trên các giao diện để có thể hiểu hệ thống như thế nào và quyết định sử dụng hay không sử dụng. Có nhiều phương pháp khác nhau mà yếu tố đầu vào và yếu tố đầu ra sử dụng các kiểu cảm biến như hình ảnh và âm thanh. Các phương pháp khác nhau này có thể sử dụng một cách kết hợp trong cùng một hệ thống và cho cùng một nhiệm vụ. Người sử dụng có thể nhìn thấy các dữ liệu đầu ra của hệ thống bằng màn hình máy tính và có thể nhập các giá trị đầu vào từ các thiết bị như chuột, bàn phím, cảm biến bằng chạm vào màn hình. Các cách mà những phương pháp này được thực hiện chỉ thể hiện được các kiểu tương tác khác nhau [6].

Theo tài liệu [6] tương tác giao diện người dùng được chia ra làm hai kiểu chính: dòng lệnh (Command-line) và tương tác giao diện người dùng (GUIs).

Giao diện dòng lệnh (Command-line): là những ví dụ của giao diện người dùng đồng bộ và tuần tự. Các hộp thoại giữa hệ thống và người dùng được thiết lập theo một

trình tự các câu hỏi và câu trả lời. Tại mỗi một bước, hệ thống sẽ chờ cho đến khi người dùng gửi lệnh. Sau đó hệ thống sẽ thực hiện xử lý, gửi kết quả đầu ra, và thực hiện bước tiếp theo. Một kiểu ví dụ cho dạng giao diện này là Unix Shell

Giao diện tương tác người dùng (GUIs) hỗ trợ đa dạng và phong phú hơn giao diện dòng lệnh (command-line). GUIs có các biểu mẫu để điền (form fill-in), lựa chọn thực đơn (menu selection), hay thao tác trực tiếp. Một giao diện có thể có nhiều cửa sổ và màn hình tương tác với nhiều đối tượng khác nhau như: thực đơn, nút ấn v.v. Tất cả bao gồm cả văn bản được trộn lẫn trong giao diện giúp tạo ra một bản giao diện hoàn chỉnh, bắt mắt hơn giao diện chỉ dùng một mình văn bản để thể hiện. Giao diện tương tác người dùng còn đa dạng phong phú hơn trong việc tương tác nhờ hỗ trợ: chuyển đổi giữa các cửa sổ, kéo thả, nhấp chuột v.v. Đặc biệt, người dùng có thể làm gián đoạn một nhiệm vụ để tương tác với một cửa sổ hay hộp thoại. Giao diện tương tác người dùng có các kiểu khác nhau: siêu văn bản (hyper-text), dựa trên web (web-based), dựa trên biểu mẫu (form-based), thao tác trực tiếp, đa nhánh (rick client), đa phương thức (multi-modal), và thực tại ảo (virtual reality).

Phương pháp kiểm thử tự động tương tác giao diện người dùng

Đối với kiểm thử tương tác giao diện người dùng, chúng ta có thể thực hiện thủ công, phân tích tĩnh, hoặc sử dụng các công cụ kiểm thử tự động như: kiểm thử bằng cách chụp và phát lại (Capture/Replay Testing), kiểm thử đầu vào ngẫu nhiên (Random input testing), kiểm thử đơn vị (Unit Testing Frameworks), kiểm thử dựa trên mô hình (Model-based Testing) [tr39-50, 6].

Kiểm thử bằng cách chụp và phát lại (Capture/Replay): Đối với công cụ này, kịch bản kiểm thử sẽ được kiểm thử viên thực hiện bằng cách tương tác với giao diện thông qua các hành vi như: di chuột, nhấp chuột, nhập dữ liệu đầu vào, chọn thực đơn, v.v. Các tương tác này sẽ được công cụ hỗ trợ chụp lại với mục đích để phát lại trình tự này cho các lần sau. Lợi ích của công cụ là có thể phân tích và nhận dạng được các loại kí tự Otica (OCR), có khả năng quan sát tốt, tạo ra được các kịch bản kiểm thử với nhiều ngôn ngữ kịch bản khác nhau. Tuy nhiên, trên thực tế công cụ chụp và phát lại chưa thực sự là một công cụ kiểm thử tự động tốt và hữu ích. Công cụ còn những hạn chế: chỉ thực hiện được khi đã có sản phẩm, bị phụ thuộc vào kiểm thử viên đặc biệt khi đưa các giá trị đầu vào, không hỗ trợ thiết kế ca kiểm thử và đánh giá mức độ bao phủ của ca kiểm thử, và phải thực hiện lại kịch bản kiểm thử từ đầu khi thay đổi cài đặt. Ví dụ cho một số công cụ chụp và phát lại: Win Runner (www.mercury.com) , Rational Robot (www.ibm.com) v.v.

Kiểm thử đầu vào ngẫu nhiên (Random Input Testing): Kiểm thử đầu vào ngẫu nhiên Random input testing còn được gọi là kiểm thử stochastic hay kiểm thử monkey. Việc kiểm thử được thực hiện chỉ dựa trên giá trị bất kì được đưa vào mà giá trị đó không cần phải mang ý nghĩa. Kiểm thử đầu vào ngẫu nhiên được thực hiện bởi bất kì ai ngay cả khi họ không hiểu về chương trình, họ chỉ cần đưa dữ liệu bất kì, hoặc đơn giản chỉ là thao tác chuột và bàn phím bất kì với chương trình. Theo đánh giá của Microsoft thì 10-20% số lỗi chương trình được tìm ra theo cách này. Tuy nhiên, phương pháp này có những hạn chế không thể bao phủ tất cả các trường hợp. Ngoài ra, các lỗi tìm thấy có thể không nằm trong phạm vi của chương trình, lỗi được tìm ra khó tái hiện và khó điều tra. Có hai kiểu công cụ áp dụng phương pháp kiểm thử này: Dumb monkeys, Smart monkeys.

Kiểm thử đơn vị (Unit Testing Frameworks): Kiểm thử tương tác giao diện người dùng hỗ trợ cho giai đoạn kiểm thử mức đơn vị thường được áp dụng phổ biến nhất, điển hình là JUnit (www.junit.org), NUnit (www.nunit.org). Công cụ này hỗ trợ tốt trong việc tổ chức và thực thi các ca kiểm thử, cụ thể là cho việc kiểm thử các API. Cách thực hiện phổ biến nhất là lập trình các ca kiểm thử một cách thủ công dựa trên các hành vi tương tác với giao diện. Đối với công cụ này, kiểm thử tương tác người dùng được coi như là kiểm thử các API. Các ca kiểm thử phải được lập trình để mô phỏng hành vi tương tác của người dùng với giao diện, sau đó bằng cách quan sát đầu ra để kiểm tra kết quả có thực sự như mong đợi. Hạn chế của công cụ đó là các trường hợp hoặc các ca kiểm thử đều ngắn, như vậy dễ dàng bỏ qua nhiều trường hợp và dẫn đến bỏ sót lỗi. Ngoài ra, việc thực hiện lập trình các ca kiểm thử đòi hỏi kiểm thử viên phải có kinh nghiệm trong công việc lập trình và luôn luôn phải nâng cao kỹ năng lập trình. Một số công cụ áp dụng phương pháp này: Abbot (abbot.sourceforge.net/), Jemmy (jemmy.netbeans.org).

Kiểm thử dựa trên mô hình: Kiểm thử dựa trên mô hình được xem như là một công cụ kiểm thử giúp tự động sinh ca kiểm thử tương tác giao diện người dùng. Công cụ này áp dụng bằng cách kiểm tra tự động sản phẩm hoặc chương trình có đúng như mô hình được thiết kế hay không. Mức cao hơn công cụ đó là có thể thực hiện sinh các ca kiểm thử cùng kết quả thực tế so với mong đợi. Hạn chế của công cụ kiểm thử tự động này đó là khó áp dụng cũng như xây dựng với một số loại giao diện tương tác người dùng. Một số công cụ áp dụng phương pháp này: TGV ([www-verimag.imag.fr/~async/TGV](http://www.verimag.imag.fr/~async/TGV)), AGEDIS (www.agedis.de), Autofocus (autofocus.informatik.tu-muenchen.de), QuickCheck (www.md.chalmers.se/~rjmh/QuickCheck), and Spec Explorer (research.microsoft.com/SpecExplorer).

2.3 Kiểm thử tự động dựa trên mô hình

Có nhiều khái niệm khác nhau về kiểm thử dựa trên mô hình. Tựu trung lại, chúng ta có thể hiểu kiểm thử dựa trên mô hình là một phương pháp kiểm thử nơi mà các ca kiểm thử được sinh ra từ mô hình đặc tả hành vi của hệ thống đang được kiểm thử. Mô hình này được biểu diễn bằng máy hữu hạn trạng thái, ô tômat, đặc tả đại số, biểu đồ trạng thái bằng UML, v.v. [1].

Để kiểm thử một ứng dụng một cách tự động dựa trên mô hình. Trước hết kiểm thử viên sẽ phải tạo ra một kịch bản bằng cách ghi lại các hành vi hoặc các hoạt động của hệ thống hay ứng dụng đó. Bộ kịch bản này được tạo ra dựa trên yêu cầu, chức năng của hệ thống. Thông thường, bộ kịch bản này kiểm thử viên sẽ tạo ra bằng cách thủ công, việc kiểm thử thủ công này tiềm ẩn nhiều rủi ro, tốn thời gian và mất công sức. Kiểm thử tự động dựa trên mô hình chính là việc tạo tự động các kịch bản kiểm thử từ mô hình được xây dựng.

Bản kịch bản được sinh ra từ mô hình chính là đầu vào cho các ca kiểm thử. Tương ứng với bộ đầu vào ta sẽ sinh các giá trị đầu ra mong muốn. Kết thúc bước này, chúng ta có ca kiểm thử. Theo quy trình kiểm thử tự động dựa trên mô hình được định nghĩa trong [1], chúng ta có năm bước cần phải thực hiện: Sinh mô hình, sinh ca kiểm thử, chạy các kịch bản kiểm thử, so sánh kết quả đầu ra thực tế với kết quả đầu ra dự kiến, và cuối cùng từ đó quyết định có hành động tiếp theo là sửa đổi mô hình, tạo thêm ca kiểm thử, dừng kiểm thử hay đánh giá chất lượng của phần mềm. Công việc đầu tiên là sinh mô hình đặc tả hành vi của hệ thống. Công việc đặc tả hành vi của hệ thống thường được sử dụng bằng các công cụ mô hình hóa. Từ mô hình đó, giúp cho việc hiểu hệ thống một cách tổng quan và rõ ràng, thuận lợi cho việc kiểm thử cũng như phát triển sản phẩm.

Chương 3: Phương pháp đặc tả tương tác giao diện cho các ứng dụng Web

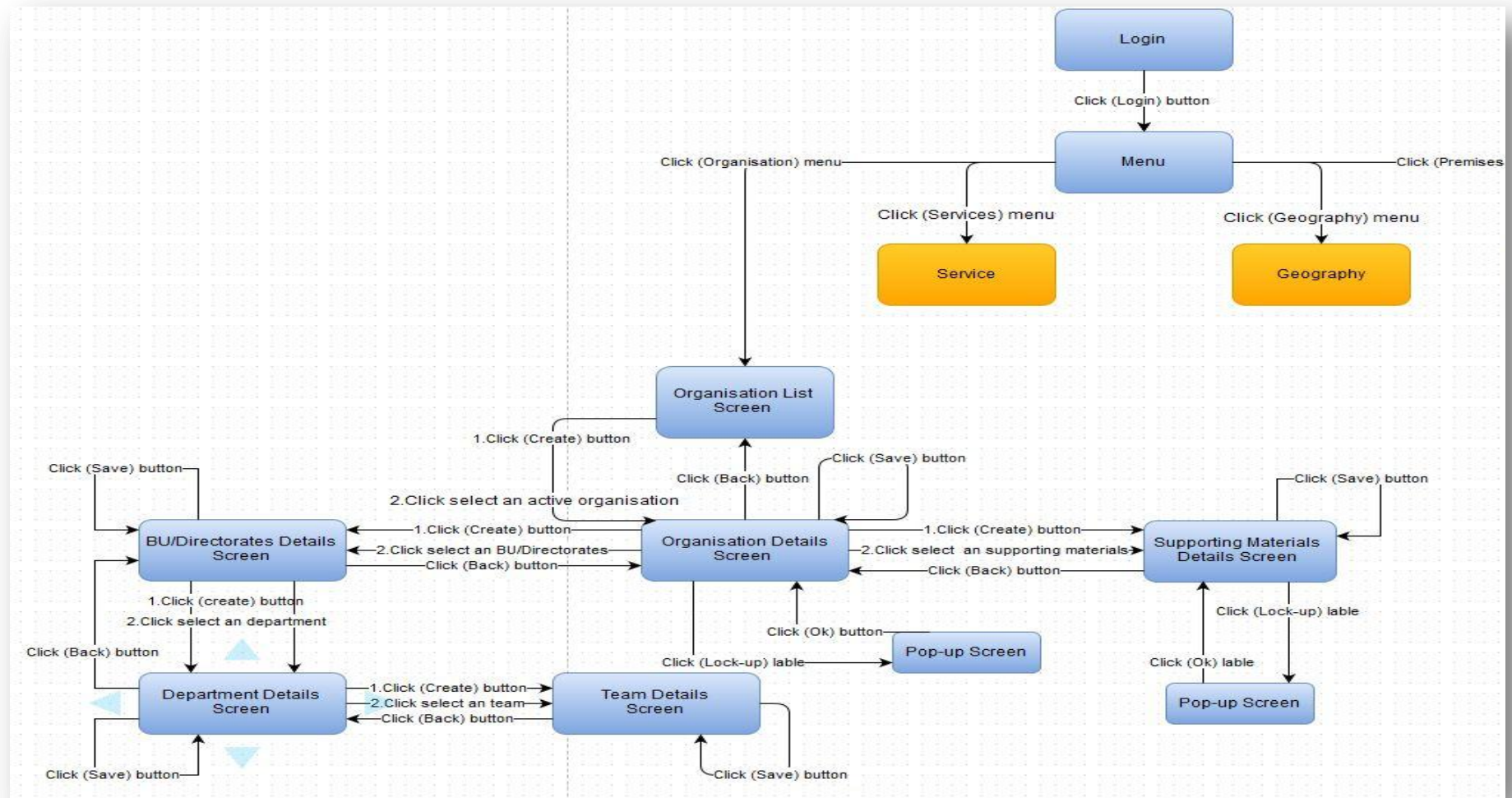
Như chương 2 đã đề cập, phương pháp kiểm thử tự động tương tác giao diện cho các ứng dụng Web là phương pháp kiểm thử được sử dụng rộng rãi và phổ biến. Để có thể kiểm thử tự động được một ứng dụng Web bằng phương pháp này, trước hết chúng ta cần phải xây dựng hoặc đặc tả tương tác giao diện. Việc đặc tả cần áp dụng kiểm thử dựa trên mô hình. Lý thuyết về kiểm thử dựa trên mô hình cũng đã được chúng tôi đề cập trong chương 2.

Mô hình là một sự biểu đồ hóa, mô tả chi tiết hệ thống, đồng thời mô tả chi tiết các khía cạnh, các đặc tính của hệ thống. Mô hình cần phải đủ chi tiết để giúp ta hiểu và đoán nhận được hành vi của hệ thống. Có nhiều phương pháp đặc tả mô hình như: máy hữu hạn trạng thái, ô-tô-mát trạng thái, máy trạng thái UML, chuỗi Markov, văn phạm, bảng quyết định, v.v. [1]. Phụ thuộc vào phương pháp và công cụ kiểm thử, chúng ta sẽ lựa chọn phương pháp đặc tả hệ thống tương ứng. Trong chương 3, chúng tôi chỉ trình bày một phương pháp đặc tả tương tác giao diện ứng dụng Web được sử dụng cho nghiên cứu này.

3.1 Phương pháp xây dựng mô hình cho toàn bộ ứng dụng Web

Khi thực hiện đặc tả tương tác giao diện cho một ứng dụng Web áp dụng máy trạng thái hữu hạn ô-tô-mát, người dùng thường gặp khó khăn trong việc xác định các trạng thái, có quá nhiều trạng thái, có quá nhiều đường chuyển trạng thái, nhưng có thể là thừa hoặc có thể là thiếu khi xác định trạng thái để đưa vào mô hình. Công việc đầu tiên khi thực hiện kiểm thử tự động cho ứng dụng Web là cần phải phân tích ứng dụng, tập trung đưa ra một mô hình tổng thể về chức năng của ứng dụng cần kiểm thử. Hình 3.1 là một ví dụ mô tả luồng hoạt động của một trang Web. Trang Web được phân cấp theo dạng hình cây từ lớn tới bé. Đầu tiên là màn hình *Login*, sau khi thực hiện *Login* thành công, *Menu* sẽ được hiển thị để người dùng lựa chọn. Các chức năng chính của trang Web sẽ được phân cấp tiếp thành ba phần: *Organisation List*, *Services*, *Geography*.

Trong hình 3.1 là một nhánh phân cấp chức năng của *Organisation List* bao gồm: *Organisation Details*, *Bu/Directorates Details*, *Support Materials Details*. Ngoài chức năng chính của *Organisation List*, còn có các chức năng phụ hỗ trợ là: *Popup*, *Departement Details*, *Team Details*.



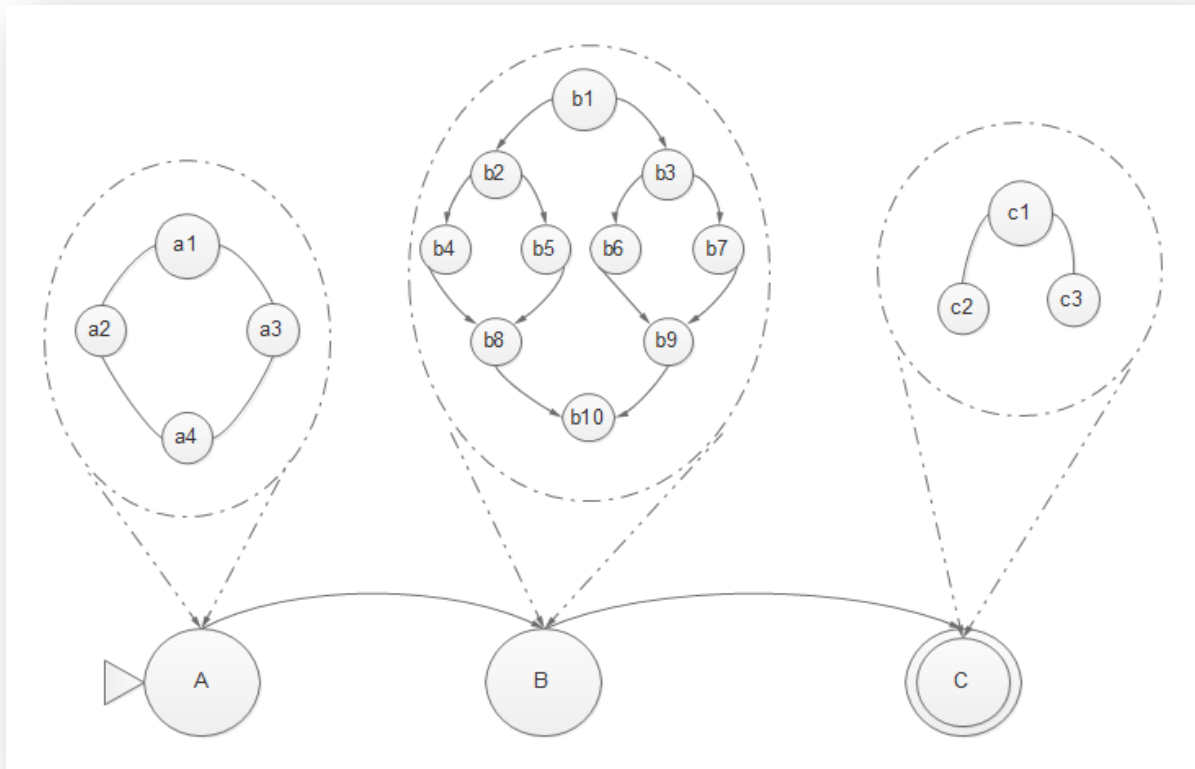
Hình 3.1. Tài liệu thiết kế màn hình của dự án

Theo tài liệu [12] có chỉ ra cách tiếp cận và phương pháp xây dựng mô hình hiệu quả cho toàn bộ ứng dụng Web tuân tự theo các bước như sau:

- 1) Các ứng dụng Web được phân chia thành các cụm
- 2) Phân tách, xác định được từng trang Web cần kiểm thử.
- 3) Xây dựng mô hình ô-tô-mát hữu hạn trạng thái cho từng cụm đã được xác định tại bước 1
- 4) Ghép nối các mô hình ô-tô-mát hữu hạn trạng thái tại bước 3 thành một ô-tô-mát hữu hạn trạng thái cho toàn bộ ứng dụng Web.

Minh họa ô-tô-mát hữu hạn của toàn ứng dụng Web được mô tả như hình 3.2. Tại bước 1, trang Web được phân chia thành các cụm lớn, như hình 3.1. Sau đó, tới bước 2, trang Web sẽ được phân tách thành các trạng thái nhỏ hơn là A, B và C. Sang bước 3, thực hiện xây dựng mô hình ô-tô-mát hữu hạn cho từng trạng thái lớn A, B, và C, đó là các mô hình ô-tô-mát a_i tương ứng trạng thái A và các mô hình ô-tô-mát b_i, c_i tương ứng trạng thái B, C. Bước 4, thực hiện ghép nối các mô hình A, B, C ta được một ô-tô-mát hữu hạn trạng thái cho toàn bộ ứng dụng Web.

Hình 3.2.



Hình 3.3. Minh họa ô-tô-mát hữu hạn trạng thái của toàn ứng dụng Web

3.2 Đặc tả tương tác giao diện của từng trang Web bằng ô-tô-mát hữu hạn trạng thái

Phương pháp đặc tả tương tác giao diện ứng dụng Web được chúng tôi sử dụng cho luận văn này là phương pháp đặc tả tương tác giao diện bằng ô-tô-mát hữu hạn trạng thái. Máy hữu hạn trạng thái (Finite State Machine - FSM) hay thường được gọi là ô-tô-mát hữu hạn trạng thái (Finite State Automaton) được biết đến như là phương pháp đặc tả phổ biến nhất cho thiết kế và kiểm thử phần mềm nói riêng và các hệ thống nói chung. FSM rất hiệu quả trong việc đặc tả hành vi dựa trên việc chuyển trạng thái của các hệ thống [1].

Dựa trên các yêu cầu và chức năng của hệ thống, có thể là các tài liệu thiết kế, kiểm thử viên đưa ra được mô hình dựa vào ô-tô-mát hữu hạn trạng thái. Với mục đích chính của luận văn mong muốn, kiểm tra tính đúng đắn của thiết kế so với chương trình. Để vẽ được máy hữu hạn trạng thái, chúng ta cần hiểu các thành phần của ô-tô-mát, ô-tô-mát hữu hạn trạng thái được thể hiện qua các thành phần như sau:

- Tập các đầu vào, tập các đầu ra, và tập các trạng thái
- Dữ liệu đầu ra là kết quả của cặp dữ liệu đầu vào và trạng thái

- Trạng thái tiếp sau là kết quả của cặp đầu vào và trạng thái tới bước tiếp theo

Để có cách nhìn một cách hệ thống và toán học, tài liệu [3] đã đưa ra định nghĩa 3.1 và 3.2 như sau:

Định nghĩa 3.1: Hành vi tương tác giao diện của một trang Web được đặc tả bằng ô-tô-mát trạng thái (Finite State Automaton - FSA) $M = \langle S, s_0, \Sigma, \delta, F \rangle$, trong đó:

- S là tập hữu hạn khác rỗng các trạng thái của trang Web
- $s_0 \in S$ là trạng thái đầu tiên được tải lên của trang Web
- Σ ứng với tập sự kiện có dạng $\langle \text{điều kiện} \rangle \text{sự kiện}$
- δ là hàm chuyển trạng thái: $\delta: S \times \Sigma \rightarrow S$
- $F \subseteq S$ là tập các trạng thái kết thúc, tương ứng với các giao diện xuất hiện cuối cùng sau một chuỗi các sự kiện liên tiếp.

Chú ý 3.1: Dạng $\langle \text{điều kiện} \rangle \text{sự kiện}$ có nghĩa là $\langle \text{sự kiện} \rangle$ chỉ xảy ra khi $\langle \text{điều kiện} \rangle$ được thỏa mãn. Ô-tô-mát hữu hạn trạng thái rỗng, ký hiệu là $M = \square$ là ô-tô-mát hữu hạn trạng thái và có tập các trạng thái $S = \emptyset$ [3].

Với định nghĩa nêu trên, áp dụng cho việc đặc tả tương tác giao diện của từng trang Web, ta coi mỗi một trang Web như một ô-tô-mát hữu hạn trạng thái. Mỗi một giao diện của một trang Web, tại một thời điểm được mô hình hóa như một trạng thái. Mỗi yêu cầu của người dùng được mô hình hóa như một hành động tạo ra hàm chuyển trạng thái. Để thực hiện đặc tả tương tác giao diện cho cả một ứng dụng Web, trước hết cần đặc tả cho một giao diện Web.

Sau đây là một số khái niệm căn bản giúp chúng ta đặc tả một trang Web như là một ô-tô-mát hữu hạn trạng thái.

Phần tử Web (Web Element): Phần tử Web là các thành phần cơ bản cấu tạo nên một trang Web. Mỗi phần tử Web có nhiều thành phần nhưng để đặc tả một trang Web chúng ta cần mô tả một số các thành phần cơ bản như: thẻ mở để khai báo bắt đầu một phần tử Web; các thuộc tính, trong đó thuộc tính *id*, *class*, *name* thường được sử dụng để xác định vị trí của phần tử Web trên trang Web; nội dung của phần tử Web thường được sử dụng để xác định trạng thái của phần tử Web và thẻ đóng nằm ở vị trí cuối cùng của phần tử Web.

Trạng thái trang Web: Trạng thái trang Web là giao diện của trang Web tại một thời điểm, mỗi trạng thái Web là một tập hợp các trạng thái của từng phần tử trong một trang Web.

Sự kiện: Sự kiện bao gồm các hành động (action) tác động lên các phần tử Web tại một thời điểm, làm thay đổi trạng thái của trang Web đó. Có nhiều sự kiện tương tác người dùng trên giao diện Web, trong đó có 4 sự kiện chính được chúng tôi sử dụng để đặc tả hành vi Web như sau: sự kiện *adddtext* dùng để nhập một đoạn ký tự vào

ô *textbox*, sự kiện *delttext* dùng để xóa đoạn ký tự ở ô *textbox*, sự kiện *click* là sự kiện nhấp chuột vào một nút (*button*) hoặc một đường dẫn, sự kiện *select* dùng để chọn một hoặc nhiều, được sử dụng cho các phần tử *combobox*, *checkbox*, *listbox*, *dropdown list*, *radio list*.

3.3 Xây dựng mô hình đặc tả tương tác giao diện cho toàn bộ ứng dụng Web

Như mục 3.1 đưa ra là phương pháp đặc tả cho từng trang Web. Nhưng trên thực tế, toàn bộ ứng dụng Web là sự kết hợp của nhiều trang Web. Nếu chúng ta đặc tả ngay mô hình cho toàn bộ ứng dụng Web thì điều này vô cùng khó khăn, thậm chí xảy ra nhiều lỗi, đặc biệt là mô hình cho hệ thống lớn và phức tạp. Trong nghiên cứu của [3] đã đưa ra cách ghép nối giữa nhiều trang Web hay ghép nối lần lượt từng ô-tô-mát hữu hạn trạng thái nhỏ lại với nhau thành một ô-tô-mát hữu hạn trạng thái lớn cho cả ứng dụng Web.

Trang Web gốc: Một trang Web $M_i = \langle S_i, s_{0i}, \Sigma_i, \delta_i, F_i \rangle$ được gọi là trang Web gốc, nếu trang Web được dùng làm gốc để các trang Web khác ghép nối vào, hay nó là trang Web khởi đầu của ứng dụng Web. $M = \langle S, s_0, \Sigma, \delta, F \rangle$ là trang Web sau khi ghép nối, với $s_0 = s_{0i}$.

Mô hình của toàn bộ ứng dụng Web được xây dựng bằng cách ghép nối mô hình của tất cả các trang Web lại với nhau bằng phép toán ghép nối được định nghĩa tại [3] như sau:

Định nghĩa 3.2: Giả sử $M_1 = \langle S_1, s_{01}, \Sigma_1, \delta_1, F_1 \rangle$ và $M_2 = \langle S_2, s_{02}, \Sigma_2, \delta_2, F_2 \rangle$ lần lượt là các ô-tô-mát hữu hạn trạng thái của 2 trang Web. Phép ghép nối của M_1 và M_2 là ô-tô-mát $M = \langle S, s_0, \Sigma, \delta, F \rangle$, kí hiệu là $M = M_1 \parallel M_2$ (M_1 là trang Web gốc). Nếu $M_1 = \Pi$, $M_2 = \Pi$, hoặc $s_{01} \notin F_2$ và $s_{02} \notin F_1$ thì $M = M_1 \parallel M_2 = \Pi$. Các trường hợp khác $M = M_1 \parallel M_2$, với $S = S_1 \cup S_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $\delta = \delta_1 \cup \delta_2$ và $F = F_1 \cup F_2$ và các trạng thái bắt đầu được xác định theo quy tắc:

- Nếu $s_{01} \notin F_2$ và $s_{02} \notin F_1$ thì $M = \Pi$
- Nếu $s_{02} \in F_1$ thì M_1 được đặt làm trang Web gốc và $s_0 = s_{01}$
- Các trường hợp còn lại thì $s_0 = s_{02}$

Chú ý 3.2: $M = \Pi$ là ký hiệu ô-tô-mát hữu hạn trạng thái rỗng, tức nó có tập các trạng thái là rỗng.

3.4 Ví dụ minh họa cho đặc tả trang Web

Đầu vào của việc đặc tả tương tác giao diện cho ứng dụng Web là tài liệu thiết kế như hình 3.1. Dựa trên tài liệu thiết kế, ta có cái nhìn tổng quan về ứng dụng Web, từ

đó xây dựng đặc tả cho từng trang Web. Theo như mô tả tại hình 3.1 ứng dụng Web bao gồm các trang sau:

- Login: Trang thực hiện việc đăng nhập của ứng dụng Web, có 3 kiểu người dùng.
- Organisation List Screen, Service, Geography: ba trang trang Web thể hiện cho ba chức năng chính của ứng dụng Web.
- Organisation List Screen: Hiển thị danh sách tất cả các tổ chức
- Organisation Details Screen: Trong trường hợp người dùng muốn tạo mới hoặc cập nhật thông tin một tổ chức nào đó.
- Bu/Directorate Details Screen, Department Details, Team Details, Support Materials Details: Các trang thuộc về một tổ chức. Người dùng có thể từ Organisation Details để đi đến các trang Web này.
- Pop-up screen: Đây là một pop-up xuất hiện trong trang Organisation Details để có thể lựa chọn người chịu trách nhiệm cho tổ chức.

Để xây dựng mô hình cho ứng dụng Web này, kiểm thử viên cần phải xác định được chính xác các trạng thái, các phần tử element của Web và các sự kiện của trang Web. Từ đó, kiểm thử viên có thể xây dựng được máy trạng thái đặc tả cho từng trang Web. Theo như tài liệu thiết kế Hình 3.1 trang đăng nhập (Login) là trạng thái đầu tiên và sẽ là máy hữu hạn trạng thái đầu tiên của ứng dụng Web.

Sau khi hoàn thành việc đăng nhập, trang chính được hiển thị như hình 3.3. Để vào chức năng Organisation, người dùng chọn menu “Organisation”, tương tự như vậy với các trang Services và Geography. Ô-tô-mát hữu hạn trạng thái thứ hai sẽ là một trong những chức năng trên. Trong phạm vi luận văn, chúng tôi quan tâm đặc tả cho chức năng Organisation.



Hình 3.4. Menu chính sau khi đăng nhập vào hệ thống

Danh sách tổ chức (*Organisation List*) được coi là một trạng thái của Web. Giao diện trong Hình 3.4 là một trạng thái thứ hai của trang Web (trạng thái *Organisation List*). Người dùng chọn một mục trong danh sách tổ chức để cập nhật, hệ thống sẽ

chuyển từ trạng thái bắt đầu sang trạng thái tiếp theo, như trong Hình 3.5 là trạng thái người dùng đã chọn một Tổ Chức để cập nhật (trạng thái *Organisation Details*).

ORGANISATION SERVICES GEOGRAPHY PREMISES				
Organisation List				
All 0-9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Create <input type="checkbox"/> Include In-active				
Organisation Name	Head Office Address Line1	Postcode	Contact	Is Active?
Công ty TNHH FPT Information Service	17 North Central Expwy1	SE15 OSS	Linh19 nguyen manh	Yes
Kiến trúc và nội thất Thời Đại	19 Maurer Court Greenwich1	SE15 OSS	Desiree Hector	Yes
Đại học FPT	17 North Central Expwy1	SE15 OSS	Linh19 nguyen manh	Yes
Cao đẳng Thực Hành FPT	10 Maurer Court Greenwich	SE10 OSS	Nicole Angie	Yes
Công ty TNHH Cung ứng Kiến Vang	Đường Lê Thánh Tông (NR Nguyễn Tiến Việt), Khu Khà Lễ, Phường Võ Cường, Thành phố Bắc Ninh, Bắc Ninh	SE10 OSS	Nicole Angie	Yes
Công Ty TNHH Dịch vụ BVTQ Quê An	Số 32A, Khu 1, Thị trấn Phố Mới, Huyện Quê Võ, Bắc Ninh	SE10 OSS	Linh18 nguyen manh	Yes
Data Solution IT Company	123 Holyhood Drive, Los Angeles	SE11 OSS	NganLTI Carey	Yes
Total System Services	13 North Central Expwy1	SE15 OSS	Desiree Hector	Yes
Ban quản lý chương trình hợp tác với WHO	138A Giảng Võ, Phường Kim Mã, Quận Ba Đình, Hà Nội	SE14 OSS	Nicole Angie	Yes
Hitachi Solution	Toukyou-to, Bunkyou-ku, Hongou, 7-choume	SE15 OSS	Darnell Deanna	Yes
Fujixerox Company	7-3-1 Hongou, Bunkyo-ku, Tokyo-to, 113-0033 JAPAN	SE15 OSS	Son Bui hong	Yes
Công ty Cung ứng máy in Phú Long	10452 North Central Expwy	SE14 OSS	Nicole Angie	Yes
Công ty TNHH Phát triển Quốc Lâm	One Washington Square	SE15 OSS	Minh nguyen ngoc	Yes
Cooper Industry	50-11, Motoyogicho - Shibuyaky Tokyo	PC106	Nicole Angie	Yes

Hình 3.5. Trạng thái bắt đầu của Danh sách Tổ Chức (Organisation List)

ORGANISATION SERVICES GEOGRAPHY PREMISES	
Organisation Details	
Information Service Feature Premise Detail Materials BU/Directorate In-active Save Back	
Organisation Name *	Kiến trúc và nội thất Thời Đại
Organisation Short Description *	Cung cấp các đồ nội thất gia đình và khách sạn
Lead Contact *	Desiree Hector
Address Line 1 *	19 Maurer Court Greenwich1
Address Line 2	Số 3/2 - Trần Phú - Hà Nội
Address Line 3	
Post Code *	SE15 OSS
City/Town	Hà Nội
County	Việt Nam
Nation/Country	United Kingdom
Preferred Organisation	<input type="checkbox"/>
Expression of Interest	<input type="checkbox"/>
Type of Business *	abattoir1 (manufacture)
SIC Code	10111
Organisation Full Description	Tư vấn nội thất văn phòng hiện đại Văn phòng bài trí nội thất lịch lãm chuyên nghiệp không chỉ thể hiện
Phone Number *	0438636976
Fax	0438636977
Email	thoidaicompany@gmail.com.vn
Web Address	http://thoidai.com.vn
Charity Number	5423
Company Number	0384

Hình 3.6. Trạng thái Chi tiết về Tổ Chức (Organisation Details)

3.3.1 Xây dựng ô-tô-mát hữu hạn trạng thái MI

Để xây dựng được mô hình cho trang Web này, chúng ta cần xác định chính xác các trạng thái và sự kiện của nó. Các trạng thái của trang Danh Sách Tổ Chức

(*Organisation List*) (Bảng 3.1.) được xác định dựa trên các trạng thái của các phần tử Web sau: tiêu đề trang Web (*Lable*), link của tổ chức (*linkText*)

Bảng 3.1. Các trạng thái Web của trang Danh sách Tổ Chức (*Organisation List*)

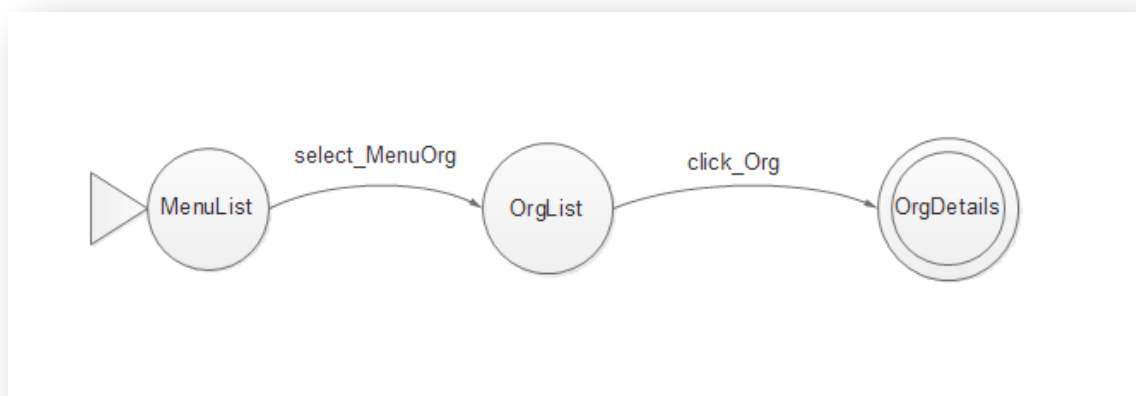
Stt	Tên trạng thái	Ý nghĩa
1	MenuList	Trạng thái ban đầu trước khi chọn <i>Organisation List</i>
2	OrgList	Khi người dùng chọn một tổ chức trong danh sách tổ chức để hiển thị thông tin
3	OrgDetails	Trạng thái trang Web khi trả về thông tin chi tiết của một tổ chức, và cũng là trạng thái kết thúc

Các sự kiện người dùng tương tác lên trang Web Danh sách Tổ Chức bảng 3.2

Bảng 3.2. Các sự kiện của trang Danh sách Tổ Chức

Stt	Tên sự kiện	Ý nghĩa
1	select_MenuOrg	Người dùng chọn menu <i>Organisation</i> để hiển thị
2	click_Org	Người dùng lựa chọn 1 trong những <i>Organisation</i> trong danh sách để hiển thị chi tiết

Trang *Organisation List* (Danh sách Tổ chức) được đặc tả bằng ô-tô-mát hữu hạn trạng thái $M_1 = \langle S_1, s_{01}, \Sigma_1, \delta_1, F_1 \rangle$ (Hình 3.6), dựa trên tập trạng thái và tập sự kiện đã được nêu trên.



Hình 3.7. Ô-tô-mát hữu hạn trạng thái M_1

Trong đó:

$S_1 = \{MenuList, OrgList, OrgDetails\}$ là tập các trạng thái của trang,

$s_{01} = MenuList$ là trạng thái bắt đầu của trang Web,

$\Sigma_1 = \{select_MenuOrg, click_Org\}$ là tập các sự kiện,

Hàm chuyển trạng thái δ_1 gồm các đường chuyển trạng thái như trong Hình 3.6, và

$F_1 = \{OrgDetails\}$ là tập các trạng thái kết thúc.

Sau khi người dùng chọn một tổ chức để hiển thị, giao diện của trang Web Danh sách Tổ chức (*Organisation List*) sẽ chuyển sang trạng thái thông tin chi tiết của tổ chức (*OrgDetails*).

3.3.2 Ghép nối ô-tô-mát hữu hạn trạng thái M_1 và M_2

Trong màn hình *Organisation Details*, người dùng có thể cập nhật các thông tin của tổ chức vừa được chọn bằng cách chọn vào các tab, Tab “Information” được hiển thị như Hình 3.7.

Trang *Organisation Details* được đặc tả bằng ô-tô-mát hữu hạn trạng thái $M_2 = \langle S_2, s_{02}, \Sigma_2, \delta_2, F_2 \rangle$ như trong hình 3.7. Tại giao diện của trang Web *Organisation Details*, người dùng có thể sửa các thông tin liên quan đến *Organisation*. Như trong hình 3.7, mô hình được thực hiện cho việc sửa ba thông tin là : *Organisation Address1* , *Nation/Country*, và *Preffered*.

ORGANISATION SERVICES GEOGRAPHY PREMISES			
Organisation Details			
Information Service Feature Premise Detail Materials BU/Directorates		In-active Save Back	
Organisation Name *	Kiến trúc và nội thất Thời Đại	Preffered Organisation	<input type="checkbox"/>
Organisation Short Description *	Cung cấp các đồ nội thất gia đình và khách sạn	Expression of Interest	<input type="checkbox"/>
Lead Contact *	Desiree Hector	Type of Business *	abattoir1 (manufacture) ...
Address Line 1 *	19 Maurer Court Greenwich1	SIC Code	10111
Address Line 2	Số 3/2 - Trần Phú - Hà Nội	Organisation Full Description	Tư vấn nội thất văn phòng hiện đại Văn phòng bài trí nội thất lịch lãm chuyên nghiệp không chỉ thể hiện
Address Line 3		Phone Number *	0438636976
Post Code *	SE15 OSS	Fax	0438636977
City/Town	Hà Nội	Email	thoidaicompany@gmail.com.vn
County	Việt Nam	Web Address	http://thoidai.com.vn
Nation/Country	United Kingdom	Charity Number	5423
		Company Number	0384

Hình 3.8. Thông tin Chức Năng *Organisation Details* - Tab *Information*

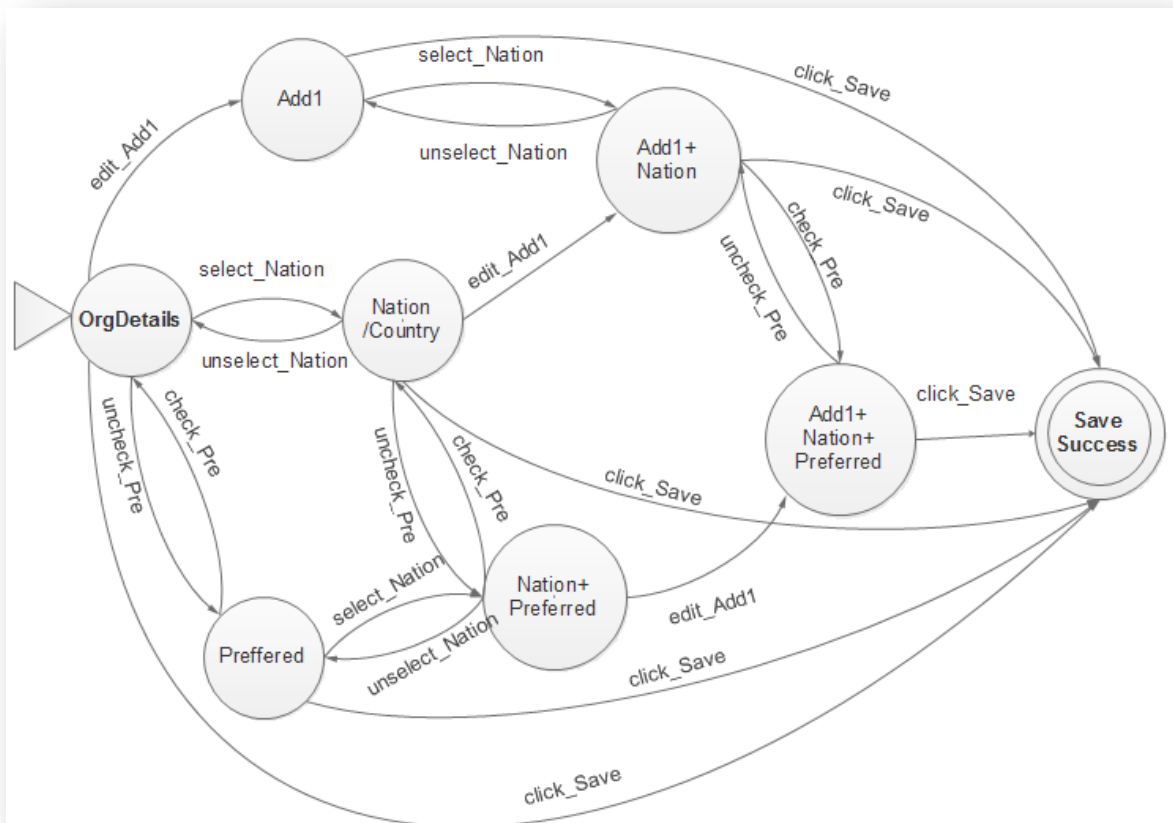
Để minh họa cụ thể cho phép toán ghép nối, chúng ta tiến hành ghép nối hai ô-tô-mát hữu hạn trạng thái của trang *Organisation List* (Danh sách tổ chức) và trang *Organisation Details* (Chi tiết tổ chức) (như hình 3.7) thành ô-tô-mát hữu hạn trạng thái $M = M_1 \parallel M_2 = \langle S, s_0, \Sigma, \delta, F \rangle$, trong đó:

$M_1 = \langle S_1, s_{01}, \Sigma_1, \delta_1, F_1 \rangle$ với $S_1 = \{ MenuList, OrgList, OrgDetails \}$; $s_{01} = MenuList$; $F_1 = \{ OrgDetails \}$ là mô hình của trang *Organisation Details*.

$M_2 = \langle S_2, s_{02}, \Sigma_2, \delta_2, F_2 \rangle$ với $S_2 = \{ OrgDetails, Add1, Nation/Country, Preferred, Add1+Nation, Nation+Preferred, Add1+Nation+Preferred, SaveSuccess \}$, $s_{02} = OrgDetails$; $F_2 = \{ SaveSuccess \}$ là mô hình của trang *Organisation Details - Tab Information*.

Thuật toán thực hiện các bước sau:

- Thuật toán tiến hành kiểm tra xem $s_{01} \in F_2$ và $s_{02} \in F_1$ hay không? Kết quả cho thấy $s_{02} = OrgDetails \in F_1$ suy ra $s_0 = s_{01} = MenuList$
- Tập trạng thái kết thúc $F = F_1 \cup F_2 = \{ OrgDetails, SaveSuccess \}$ và $S = S_1 \cup S_2 = \{ MenuList, OrgList, OrgDetails, Add1, Nation/Country, Preferred, Add1+Nation, Nation+Preferred, Add1+Nation+Preferred, SaveSuccess \}$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $\delta = \delta_1 \cup \delta_2$.



Hình 3.9. Ô-tô-mát hữu hạn trạng M_2

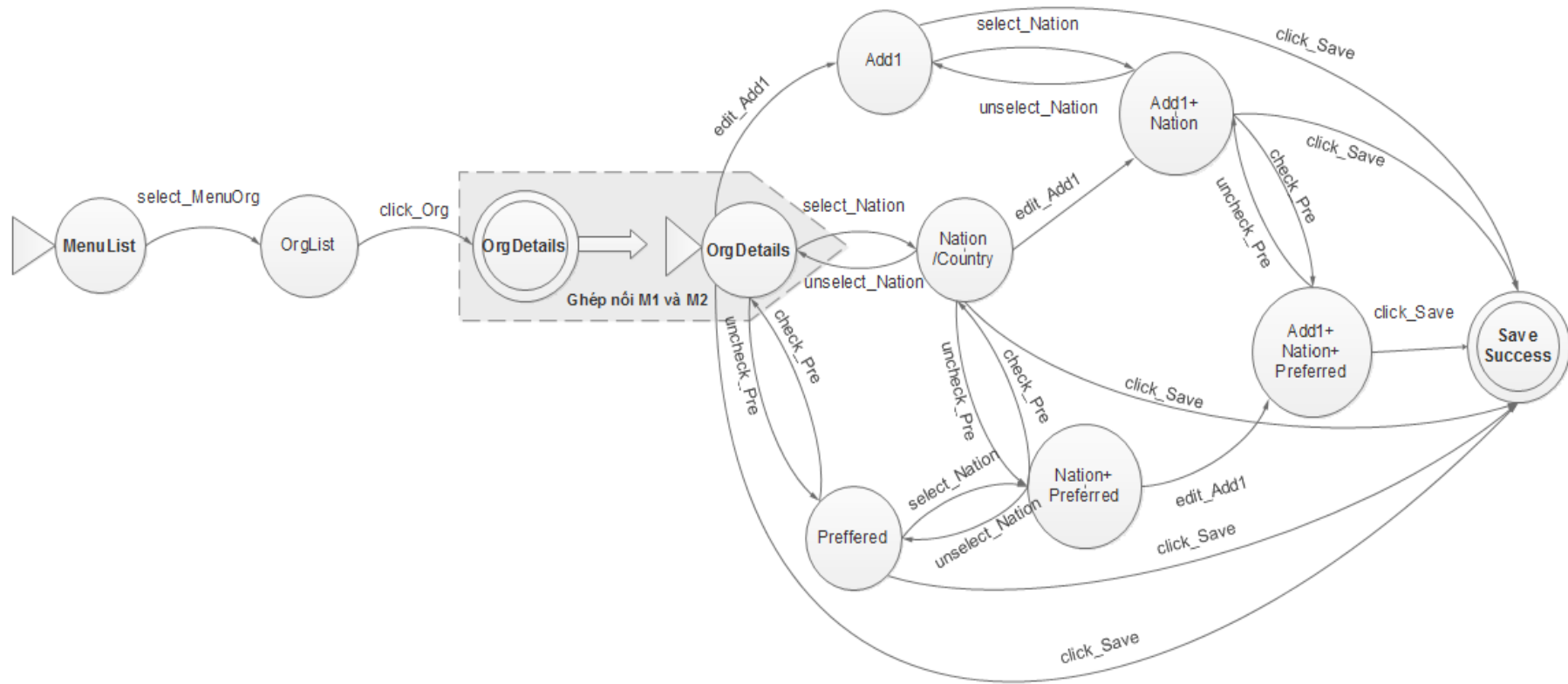
Ta có $M_1 = \langle S_1, s_{01}, \Sigma_1, \delta_1, F_1 \rangle$, $M_2 = \langle S_2, s_{02}, \Sigma_2, \delta_2, F_2 \rangle, \dots, M_n = \langle S_n, s_{0n}, \Sigma_n, \delta_n, F_n \rangle$ lần lượt là các ô-tô-mát hữu hạn của n trang Web với M_1 là trang Web gốc. Đầu tiên, M_1 ghép nối với M_i ($2 \leq i \leq n$) thành M_{1i} , thuật toán sau đó sẽ tiến hành ghép nối M_{1i} với M_j ($2 \leq j \leq n$ và $i \neq j$). Thuật toán ghép nối kết thúc sau khi duyệt qua tất cả các ô-tô-mát hữu hạn trạng thái.

Nhận xét: Phép ghép nối như định nghĩa trên có tính kết hợp nhưng không có tính giao hoán [3].

3.5 Biểu diễn mô hình đặc tả dưới dạng các tệp tin MS Excel

Công cụ để thực hiện sinh ca kiểm thử tự động cần phải có đầu vào là mô hình, nhưng công cụ không thể (hoặc chưa thể) đọc được trực tiếp từ mô hình ô-tô-mát hữu hạn. Do đó, trong đề tài luận văn này, người viết đã chuyển đổi mô hình ô-tô-mát dưới dạng các tệp tin Excel. Định dạng của tệp tin Excel được chia thành bốn bảng như sau:

- 1) *Bảng Element_html*: Mô tả các phần tử Web của trang Web
- 2) *Bảng State*: Mô tả các trạng thái
- 3) *Bảng Event*: Mô tả các sự kiện



Hình 3.10. Mô hình M sau khi thực hiện thuật toán ghép nối giữa M_1 và M_2

- 4) *Bảng Transition*: Mô tả các hàm chuyển trạng thái giữa các trạng thái trong bảng *State*.

Sau đây, chúng tôi sẽ mô tả chi tiết từng bảng trong tệp tin Excel của trang chi tiết tổ chức *Organisation Details - Tab Information*.

Bảng Element_html (bảng các phần tử Web): Bảng này được thiết kế để đặc tả các phần tử Web của trang Web cần kiểm thử. Bảng 3.3 là bảng mô tả các phần tử Web của trang *Organisation Details - Tab Information*. Cấu trúc của bảng này gồm có bốn cột chính: cột *id*, *html*, *type* và *value*, mỗi cột đều có những ý nghĩa khác nhau và được mô tả trong Bảng 3.4.

Bảng 3.3. Bảng các phần tử Web của trang *Organisation Details - Tab Information*

4			
id	html	type	value
0	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_txtLine1	textbox	19 Maurer Court Greenwich1
1	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_ddlCountry	dropdown	France
2	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_chkOrg	cbox	
3	ctl00_ContentPlaceHolder1_iblInform	texthtml	Save Successful Organisation

Bảng 3.4. Ý nghĩa của từng cột trong Bảng Element_html

Tên bảng	Mô tả
<i>id</i>	Định danh phần tử Web, được đánh số thứ tự tăng dần
<i>html</i>	Lưu các thuộc tính của phần tử Web. Các thuộc tính này được lấy từ các thẻ HTML của trang Web. Nếu phần tử Web được đặc tả có thuộc tính định danh là <i>id</i> , <i>classname</i> , <i>name</i> thì cột này ghi giá trị định danh của phần tử Web đó.
<i>type</i>	Cột lưu kiểu của phần tử Web, công cụ có thể tiến hành kiểm thử thành công với đa số các kiểu phần tử Web như: <i>textarea</i> , <i>textbox</i> , <i>checkbox</i> , <i>combobox</i> , <i>radiobox (radio)</i> , <i>listbox</i> , <i>button (btn)</i> , <i>texthtml</i> , <i>datagrid (grid)</i> , v.v. Giá trị của cột này được công cụ sử dụng để lấy nội dung

	của phần tử Web tương ứng
<i>value</i>	Cột lưu nội dung đầu ra mong muốn của phần tử Web hoặc nội dung cần đưa vào đối với phần tử Web kiểu nhập thông tin vào, chẳng hạn như phần tử Web dạng <i>textbox</i> .
<i>Số phần tử</i>	Trên mỗi bảng đều có một ô đầu tiên số các phần tử của bảng

Bảng State (bảng trạng thái): là bảng mô tả các trạng thái của trang Web. Bảng 3.5 là bảng đặc tả chi tiết của từng trạng thái và giá trị tương ứng đối với từng trạng thái của trang *Organisation Details - Tab Information*. Ý nghĩa của từng cột được nêu tại Bảng 3.6.

Bảng 3.5. Bảng các trạng thái của trang *Organisation Details - Tab Information*

8					
State	name	0	1	2	3
0	OrgDetails				
5	Add1	o		o	
5	NationCountry		o		o
5	PreferredOrg				
5	Name+Nation	o	o	o	o
5	Nation+Preferred				
5	Name+Nation+Preferred	o	o	o	o
1	SaveSuccess				

Bảng 3.6. Ý nghĩa của các cột trong bảng trạng thái

Tên bảng	Mô tả
<i>Cột đầu tiên</i>	Phân loại trạng thái: số 0 nếu trạng thái đó là trạng thái bắt đầu, số 1 nếu đó là trạng thái kết thúc và số 5 biểu diễn các trạng thái bình thường. Theo Bảng 3.4 :

	Trạng thái đầu tiên là OrgDetails có giá trị 0 Trạng thái kết thúc SaveSuccess có giá trị 1
<i>name</i>	Tên trạng thái
<i>Các cột sau cột name</i>	Số cột còn lại là số các phần tử Web (bằng số phần tử Web của bảng <i>Element_html</i>). Ở mỗi cột này, ô đầu mỗi cột là các <i>id</i> định danh được đánh như bảng <i>Element_html</i> , các ô tiếp theo của mỗi cột này là trạng thái của phần tử Web tương ứng với từng trạng thái của trang Web. Chúng tôi đặc tả ba trạng thái của một phần tử Web như sau: <ul style="list-style-type: none"> ○ Ký hiệu "o": Phần tử Web có một giá trị bất kỳ khác rỗng. ○ Ký hiệu "null": Phần tử Web có một giá trị rỗng. ○ Ký hiệu " ": Phần tử Web không nhận giá trị nào cả.

Bảng Event (Sự kiện): Bảng 3.7 liệt kê tất cả các sự kiện diễn ra của trang Web *Organisation Details - Tab Information*. Ý nghĩa của từng cột trong bảng sự kiện được nêu tại bảng 3.8.

Cách thực hiện của công cụ đối với từng sự kiện, ví dụ cho sự kiện *edit_Add1*. Sự kiện *edit_Add1* xảy ra khi người dùng muốn sửa thông tin của trường *Add1 (Organisation Address 1)* có html tương ứng là *ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_txtLine1*.

Đối chiếu với bảng *Element_html*, với *id* *ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_txtLine1* ta có *value* "19 Maurer Court Greenwich1". Hành vi của sự kiện là xóa chuỗi *Add1 (Organisation Address 1)* trước và thay bằng một chuỗi *Add1 (Organisation Address 1)* mới là "19 Maurer Court Greenwich1". Tương tự với các sự kiện còn lại, tuy nhiên với các sự kiện *click* sẽ không có *value*.

Bảng 3.7. Bảng các sự kiện của trang *Organisation Details - Tab Information*

7			
Event	name	html	action
1	check_PREFERRED	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_chkOrg	click
2	click_Save	ctl00_ContentPlaceHolder1_btnSave	click

SaveSuccess	edit_Add1	select_Nation	check_PREFERRED				
-------------	-----------	---------------	-----------------	--	--	--	--

Số cột trong bảng *transition* tương ứng với số các trạng thái ở bảng *State*. Ở đầu mỗi hàng của cột đầu tiên là tên các trạng thái bắt đầu và mỗi cột từ cột thứ hai trở đi là tên các trạng thái kết thúc của mỗi *transition*, các ô ở giữa có công thức *[guard]event* ([điều kiện] tên sự kiện). Có ba kiểu giá trị của một ô trong bảng *Transition*:

- Chỉ có tên sự kiện (*event*): tồn tại một *transition*.
- Gồm cả điều kiện và sự kiện (*[guard]event*): khi điều kiện là đúng thì *transition* được thực hiện.
- Ô trống: Không có *transition* giữa hai trạng thái.

Mối liên kết giữa các bảng: Tập tin Excel đặc tả trang Web *Organisation Details - Tab Information* gồm bốn bảng trên, bốn bảng này được nối lại như bảng 3.10.

Bảng 3.10. Tập tin Excel đặc tả trang Web *Organisation Details - Tab Information*

4						
id	html	type	Value			
0	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_txtLine1	textbox	19 Maurer Court Greenwich1			
1	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_ddlCountry	dropdown	France			
2	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_chkOrg	cbox				
3	ctl00_ContentPlaceHolder1_lblInform	texthtml	Save Successful Organisation			
8						
State	name	0	1	2	3	
0	OrgDetails					
5	Add1	o		o		
5	NationCountry		o		o	
5	PreferredOrg					
5	Name+Nation	o	o	o	o	

5	Nation+Preferred						
5	Name+Nation+Preferred			o	o	o	o
1	SaveSuccess						
7							
Event	name	html					action
1	check_PREFERRED	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_chkOrg					click
2	click_Save	ctl00_ContentPlaceHolder1_btnSave					click
3	edit_Add1	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_txtLine1					edittext
4	select_Nation	ctl00_ContentPlaceHolder1_tabContainer1_tabPanel1_ddlCountry					select
9							
name	Add1	Nation Country	Preferred Org	Name +Nation	Nation +Preferred	Name +Nation +Preferred	Save Success
OrgDetails	edit_Add1	select_Nation	check_PREFERRED				
Add1				select_Nation			click_Save
NationCountry				edit_Add1	check_PREFERRED		click_Save
PreferredOrg					select_Nation		click_Save
Name+Nation						check_PREFERRED	click_Save
Nation+Preferred						edit_Add1	click_Save
Name+Nation+Preferred							click_Save
SaveSuccess	edit_Add1	select_Nation	check_PREFERRED				

Từ các mô tả chi tiết về các bảng trong tệp tin *Excel* trên cho chúng ta thấy mối liên kết giữa các bảng trong một tệp tin như sau: Bảng *Element_html* liên kết với bảng *State* thông qua cột *id* (liên kết qua các phần tử Web). Bởi vì, mỗi trạng thái trang Web

được mô tả bằng tập hợp các trạng thái của tất cả các phần tử Web. Bảng *Event* liên kết với bảng *Element_html* thông qua cột *html* (các sự kiện được thực hiện trên các phần tử Web). Bảng *Transition* liên kết với bảng *Event* thông qua các sự kiện, liên kết với bảng *State* thông qua các trạng thái. Các mối liên kết này được sử dụng để xây dựng ô-tô-mát hữu hạn trạng thái và thực thi các sự kiện đối với trang Web tương ứng.

Ứng dụng Web được xây dựng gồm nhiều trang Web. Các trang Web này được thực hiện theo thứ tự tương tác người dùng. Các tệp tin Excel được đánh số lần lượt theo thứ tự đó. Chúng tôi dựa trên thứ tự các tệp tin Excel, các trạng thái bắt đầu, trạng thái kết thúc của từng tệp tin, áp dụng thuật toán ghép nối được đề xuất ở phần trước để nối các ô-tô-mát hữu hạn trạng thái lại thành một bản đặc tả cho toàn bộ ứng dụng Web.

Việc xây dựng các tệp tin đầu vào yêu cầu người thiết kế phải có kỹ năng về phân tích, thiết kế hệ thống và phải hiểu rõ chi tiết hoạt động của hệ thống thì mới xây dựng được các tệp tin tốt (có khả năng tìm được tối đa lỗi của ứng dụng Web). Trong thực tế, các giao diện trang Web thường có quá nhiều các phần tử Web cần đặc tả và số lượng các trạng thái Web là rất lớn, không thể biểu diễn hết trong một trang Excel. Vì vậy, công việc này rất mất thời gian và tiềm ẩn nhiều nguy cơ sai sót trong quá trình thực hiện. Người thiết kế cần sử dụng các tiện ích hỗ trợ biểu diễn ô-tô-mát hữu hạn trạng thái và tạo tự động các bảng *Transition* để giải quyết những khó khăn trên. Một tiện ích hỗ trợ đã được giới thiệu tại [3]. Ngoài ra còn một số yêu cầu về thiết kế trang Web nhằm phục vụ cho việc đặc tả ứng dụng Web bằng phương pháp đã nêu trên như sau.

Chương 4: Sinh và thực thi các ca kiểm thử tự động

Từ mô hình ô-tô-mát hữu hạn trạng thái được đặc tả trong Chương 3, các ca kiểm thử tự động sẽ được sinh tự động. Việc sinh các ca kiểm thử tự động được dựa trên hai thuật toán. Thuật toán thứ nhất là thuật toán mở rộng của thuật toán duyệt đồ thị theo chiều sâu. Thuật toán thứ hai là thuật toán bổ sung cho thuật toán thứ nhất. Trong các mục bên dưới, chúng tôi sẽ trình bày chi tiết về thuật toán sinh các ca kiểm thử từ mô hình đặc tả hình thức.

4.1 Sinh các ca kiểm thử từ mô hình đặc tả hình thức

4.1.1 Đường dẫn kiểm thử

Trong nghiên cứu này, các đường dẫn kiểm thử được sinh ra từ thuật toán có cấu trúc như sau: $\langle \text{trạng thái bắt đầu} \rangle * \langle \text{sự kiện}_i \rangle = \langle \text{trạng thái}_i \rangle * \dots = \langle \text{trạng thái kết thúc} \rangle$.

Mỗi đường dẫn kiểm thử là một chuỗi các hành động mà người dùng tương tác với ứng dụng Web. Thuật toán sinh đường dẫn kiểm thử phải thỏa mãn các yêu cầu [3] sau:

- Đảm bảo tất cả các đỉnh và các cạnh của đồ thị đều phải được duyệt qua.
- Kết quả trả về là các đường dẫn không trùng nhau.
- Mỗi đường dẫn kiểm thử có trạng thái bắt đầu chính là trạng thái bắt đầu của ô-tô-mát hữu hạn trạng thái đặc tả trang Web gốc, và có trạng thái kết thúc là một trong những các trạng thái kết thúc của ô-tô-mát hữu hạn trạng thái. Giữa hai trạng thái là một sự kiện.

4.1.2 Thuật toán sinh tự động các đường dẫn kiểm thử

Như đã trình bày tại đầu chương, trong mục này chúng tôi sẽ đề cập để hai thuật toán được đề xuất bởi [3]. Thuật toán thứ nhất như hình 4.1 và được gọi là thuật toán 4.1 và thuật toán 4.2 (hình 4.2).

4.1.2.1. Thuật toán 4.1

Mô hình đặc tả hành vi của hệ thống được đặc tả dựa trên ô-tô-mát hữu hạn trạng thái và được coi như một đồ thị có hướng. Việc xác định được tất cả các đường đi và sinh các ca kiểm thử dựa trên mô hình hay đồ thị này có thể áp dụng các phương pháp: duyệt ngẫu nhiên, duyệt theo chiều sâu, duyệt theo chiều rộng. Trong nghiên cứu này, [3] đã áp dụng thuật toán duyệt theo chiều sâu và có mở rộng nhằm liệt kê tất cả các đường đi có thể trong FSA. Chi tiết các bước của thuật toán tìm kiếm theo chiều sâu (Depth First Search) được trình bày trong thuật toán 4.1 [3].

Thuật toán 4.1 *Depth_First_Search*: int *i*, path *PATH*

1. // *PATH* là biến lưu các test path
2. // *arr* lưu test path tạm thời
3. **Bool** backTrack = true;
4. **for** tất cả các cạnh **do**
5. **if** cạnh chưa được duyệt **then**
6. backTrack = false;
7. Thông báo cạnh đã duyệt qua;
8. Thêm đỉnh vào *arr*
9. *Depth_First_Search*: *j*, *PATH*;
10. Bỏ đỉnh khỏi *arr*
11. **end if**
12. **end for**
13. **if** backTrack = true **then**
14. Thêm *arr* vào *PATH*
15. **end if**

Biến đầu vào là trạng thái bắt đầu *i* và các đường dẫn kiểm thử được lưu trong biến *PATH*. Đầu ra là các đường dẫn kiểm thử mà có khả năng bao phủ hết tất cả các trường hợp có thể xảy ra khi người dùng tương tác giao diện ứng dụng Web. Đầu tiên, biến *backTrack* được khởi tạo với giá trị ban đầu là *true* (dòng 3). Sau đó dòng 4 sẽ kiểm tra lần lượt từng *transition* - hàm chuyển trạng thái thông qua một sự kiện, nếu *transition* chưa được duyệt (dòng 5) thì biến *backTrack* được gán giá trị *false* (dòng 6) và trạng thái bắt đầu của *transition* được thêm vào chuỗi *arr* - biến lưu các đường dẫn kiểm thử tạm thời (dòng 8) rồi thông báo cạnh đã được duyệt (dòng 7). Thuật toán thực hiện đệ quy với đầu vào là trạng thái cuối của *transition* vừa duyệt (dòng 9) cho đến khi không thêm được *transition* nào vào đường dẫn kiểm thử *arr*, khi đó biến *backTrack* có giá trị *true* (dòng 13), *arr* được thêm vào *PATH* (dòng 14). Cuối cùng, các trạng thái

trong *arr* được loại bỏ để bắt đầu tạo một đường dẫn kiểm thử mới (dòng 10).

Thuật toán 4.1 là sự mở rộng của thuật toán duyệt đồ thị theo chiều sâu. Vì thế, tất cả các cạnh của đồ thị sẽ được duyệt qua sau khi áp dụng thuật toán tìm kiếm theo chiều sâu. Tuy nhiên, trong một số trường hợp sẽ có đường dẫn kiểm thử mà trạng thái bắt đầu là trạng thái bắt đầu của đồ thị còn trạng thái kết thúc của nó không phải là trạng thái kết thúc của đồ thị. Do đó, phương pháp dựa vào việc sử dụng thuật toán 2 [3] để hoàn thiện các đường dẫn kiểm thử theo tiêu chí: trạng thái cuối cùng của mỗi đường dẫn kiểm thử là một trạng thái kết thúc của đồ thị.

4.1.2.2. Thuật toán 4.2

Đầu vào của thuật toán 2 là tập các đường dẫn kiểm thử *PATH* được sinh ra từ thuật toán 4.1. Thuật toán duyệt từng đường dẫn kiểm thử trong *PATH*, giả sử đường dẫn kiểm thử *i* có trạng thái cuối không phải là trạng thái kết thúc của đồ thị thì thuật toán kiểm tra chừng nào điều kiện trên còn thỏa mãn (dòng 4) thì thuật toán sẽ tìm cạnh của đồ thị có đỉnh đầu giống với trạng thái cuối của đường dẫn kiểm thử đó (dòng 6) và thêm đỉnh cuối của cạnh vào đường dẫn kiểm thử (dòng 7).

Thuật toán 4.2 *Add_path*: path *PATH*

```

1. loop
2.   for duyệt các test path của PATH do
3.     if trạng thái cuối cùng của test path i không là trạng thái kết thúc then
4.       while trạng thái cuối cùng của test path i không là trạng thái kết thúc do
5.         for duyệt tất cả các cạnh của đồ thị do
5.           if cạnh đồ thị có đỉnh bắt đầu là đỉnh cuối của PATH then
6.             Thêm đỉnh cuối của cạnh vào test path i;
7.           end if
8.         end for
9.       end while
10.    end if
12.  end for
13. end loop

```

Như vậy, bằng cách áp dụng hai thuật toán được trình bày ở trên, chúng ta có thể sinh ra tất cả các đường dẫn kiểm thử từ mô hình tương tác của người dùng trên giao diện Web. Chất lượng của các đường dẫn kiểm thử này phụ thuộc vào mô hình được đặc tả trước đó.

4.2 Thực hiện các ca kiểm thử

Từ bản mô hình đặc tả và tệp excel đầu vào, các đường dẫn kiểm thử được sinh ra một cách tự động. Để xác định tính chính xác của chương trình so với mô hình đặc tả xây dựng ban đầu, đầu tiên, chúng ta cần cài đặt hệ thống dựa trên mô hình đặc tả. Khi đã cài đặt xong chương trình, chúng ta sẽ tiến hành thực thi các ca kiểm thử. Việc thực hiện các ca kiểm thử một cách tự động theo các bước sau:

- Bước 1: Tách đường dẫn kiểm thử thành các đường dẫn kiểm thử nhỏ (*transition*). Mỗi đường dẫn kiểm thử nhỏ chỉ bao gồm: $\langle \text{trạng thái}_i \rangle * \langle \text{sự kiện} \rangle = \langle \text{trạng thái}_j \rangle$.

- Bước 2: Thực hiện từng đường dẫn nhỏ. Phương pháp sẽ kết nối với trang Web thông qua Selenium, dùng Selenium để xác định trạng thái_i của trang Web. Nếu xác định trạng thái thành công, tiếp tục xác định vị trí phần tử Web và thực hiện sự kiện. Cuối cùng, xác định trạng thái của trang Web sau khi đã thực hiện sự kiện, bằng cách lấy các giá trị của các phần tử Web tương ứng với trạng thái này (trạng thái Web thực tế), so sánh với giá trị trạng thái của trang Web trong bản đặc tả (trạng thái Web mong muốn) và chuyển sang bước 3. Ngược lại thì dừng thực hiện và chuyển sang bước 4.

- Bước 3: Nếu kết quả so sánh trên trùng nhau thì tiếp tục lặp lại hai bước trên cho đường dẫn con tiếp theo. Ngược lại, thì kết thúc việc thực hiện đường dẫn lớn và chuyển sang bước 4.

- Bước 4: Kết luận về kết quả thực hiện đường dẫn kiểm thử.

Trong mục 5.2.3, luận văn đã nêu một cách chi tiết cụ thể cho việc thực hiện các ca kiểm thử áp dụng bốn bước vừa nêu trên.

Chương 5: Công cụ và thực nghiệm

Chương 3, chương 4 đã trình bày phương pháp đặc tả hình thức giao diện ứng dụng Web bằng ô-tô-mát hữu hạn trạng thái, phương pháp sinh tự động các ca kiểm thử bằng cách duyệt đồ thị trạng thái. Nội dung của chương 5 sẽ trình bày về công cụ kiểm thử tự động tương tác giao diện các ứng dụng Web được xây dựng từ các phương pháp trong chương 2, 3 và trình bày kết quả thực nghiệm. Trước tiên, chúng tôi sẽ giới thiệu về hai công cụ hỗ trợ chính: một là, giới thiệu về Selenium, một số API WebDriver chính của Selenium được sử dụng để thực hiện tương tác với các phần tử Web; hai là, giới thiệu JFLAP - công cụ hỗ trợ tạo tệp tin đầu vào. Tiếp theo, chúng tôi sẽ trình bày về công cụ kiểm thử tự động tương tác giao diện ứng dụng Web. Cuối chương sẽ đưa ra một ứng dụng Web và kết quả kiểm thử ứng dụng này bằng công cụ kiểm thử tự động.

5.1 Giới thiệu các công cụ hỗ trợ

Luận văn dựa trên cơ sở phương pháp và công cụ được đề xuất bởi [3] để cài đặt, nghiên cứu và nâng cấp công cụ kiểm thử tự động tương tác giao diện cho các ứng dụng Web. Công cụ được xây dựng bằng ngôn ngữ lập trình Java, sử dụng trình soạn thảo Eclipse. Ngoài ra, luận văn chúng tôi còn tích hợp, sử dụng hai thư viện hỗ trợ là Selenium phiên bản 2.32.0 và sử dụng công cụ mã nguồn mở JFLAP để tạo đầu vào cho công cụ kiểm thử tự động.

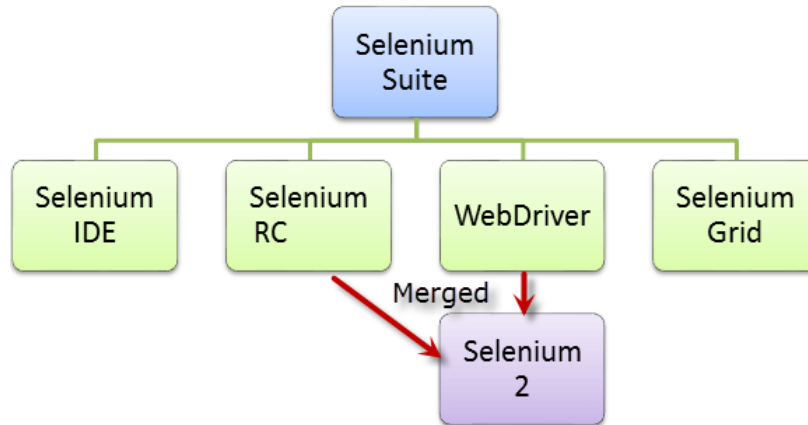
5.1.1. Giới thiệu Selenium và một số API WebDriver được sử dụng

Selenium là một trong những công cụ kiểm thử phần mềm tự động mã nguồn mở (open source test automation tool) mạnh mẽ nhất hiện nay cho việc kiểm thử ứng dụng Web. Selenium script có thể chạy được trên hầu hết các trình duyệt như IE, Mozilla FireFox, Chrome, Safari, Opera; và hầu hết các hệ điều hành như Windows, Mac, Linux [7].

Selenium được chia làm bốn phần:

- Selenium IDE
- Selenium RC (Selenium 1 – Selenium Remote Control)
- Selenium Gird
- Selenium WebDriver (Selenium 2)

Hiện nay, Selenium được sử dụng khá nhiều trong việc kiểm thử tự động các ứng dụng Web. Theo tài liệu [7] Selenium được giới thiệu với các đặc tính nổi trội và linh hoạt như bên dưới:



Hình 5.1. Cấu trúc của Selenium ¹

- 1) Mã nguồn mở: Điểm mạnh của Selenium chính là mã nguồn mở, vì người sử dụng không cần phải trả bất cứ phí nào hay phải lo lắng về hạn sử dụng.
- 2) Cộng đồng hỗ trợ: Khi là một phần mềm mã nguồn mở thì công cụ Selenium sẽ có một cộng đồng hỗ trợ mạnh mẽ. Hơn nữa, Google là nơi phát triển Selenium nên chúng ta có thể yên tâm về sự hỗ trợ khi có vấn đề về công cụ. Nhưng mặt khác đây cũng là điểm yếu, vì khi cần sự hỗ trợ gấp thường sẽ không được, và cũng có quá nhiều giải pháp, và nhiều ý kiến đối với một vấn đề cần giải quyết khi đưa ra cộng đồng hỗ trợ cần phải lựa chọn.
- 3) Selenium hỗ trợ nhiều ngôn ngữ lập trình.
- 4) Selenium hỗ trợ chạy trên nhiều hệ điều hành khác nhau với mức độ chỉnh sửa script hầu như là không có. Tuy nhiên, để thực thi trên nhiều hệ điều hành điều đó phụ thuộc vào khả năng viết script của từng kiểm thử viên.
- 5) Chạy test case ở nền: Việc thực hiện trên nền có nghĩa là chúng ta có thể thực thi nhiều công việc khác trên cùng máy tính cá nhân khi đang thực hiện chạy script của Selenium. Điều đó giúp cho chúng ta không bị mất quá nhiều tài nguyên và máy móc.
- 6) Không hỗ trợ các ứng dụng dành cho Windows: Selenium không hỗ trợ các ứng dụng dành cho Windows mà chỉ tương tác và hỗ trợ với các trình duyệt, và các cảnh báo của trình duyệt. Vậy nên, để xử lý các trường hợp cần tương tác với hệ thống hay một ứng dụng thứ ba, chúng ta cần một hay nhiều thư viện khác như AutoIt hay Coded UI.

¹ Trích dẫn từ <http://career.guru99.com/top-30-selenium-interview-questions-answers/>

* **Selenium IDE:** Khi bắt đầu với Selenium, người dùng thường lựa chọn Selenium IDE vì đó là cách bắt đầu dễ dàng nhất. Công cụ Selenium IDE cho phép chúng ta Record/Playback một kịch bản kiểm thử. Đây là tiện ích mở rộng hỗ trợ cho FireFox. Chúng ta chỉ có thể Record trên trình duyệt FireFox, nhưng bù lại, chúng ta có thể Playback trên các trình duyệt khác như Internet Explorer, Chrome... Trong luận văn nghiên cứu, để thực hiện lấy các phần tử `html_id` trên một ứng dụng Web, theo cách thông thường, người dùng có thể sử dụng nháy chuột phải và chọn *Inspect Element* rồi tìm từng phần tử của ứng dụng Web. Tuy nhiên, đối với Selenium IDE, người dùng có thể Record/Playback, hành động này có thể truy xuất và lấy ra được các phần tử của ứng dụng Web (`id`, `class`,...).

* **Selenium RC:** Selenium IDE rất dễ sử dụng cho người mới bắt đầu, tuy nhiên đối với các nhiệm vụ và công việc phức tạp, yêu cầu cần phải có tính lô-gic cao, người dùng nên chọn Selenium RC để thực hiện. Ví dụ như: các câu lệnh có điều kiện, công việc lặp đi lặp lại, ghi và báo cáo kết quả kiểm thử, xử lý lỗi, xử lý lỗi không mong muốn, chụp các kết quả kiểm thử lỗi v.v. [14, tr.49]

* **Selenium Grid:** là một hệ thống hỗ trợ người dùng thực thi kịch bản kiểm thử trên nhiều trình duyệt một cách song song mà không cần phải chỉnh sửa kịch bản kiểm thử.

* **Selenium WebDriver (Selenium 2):** Về cơ bản, Selenium WebDriver không phải là một công cụ kiểm thử tự động. Selenium WebDriver là một bộ thư viện (API) hỗ trợ cho việc thiết kế các kịch bản kiểm thử cho các ứng dụng Web dựa trên các ngôn ngữ lập trình như Python, C# hay Java [3].

Để thực thi một kịch bản kiểm thử ứng dụng Web, vấn đề đầu tiên chúng ta phải quan tâm chính là trình duyệt. Đối tượng WebDriver sẽ khởi tạo một trình duyệt tương ứng với cách mà chúng ta thiết lập trong mã nguồn. Đối tượng WebDriver có thể là các trình duyệt như FireFox, Internet Explorer, Chrome v.v. Đối tượng WebDriver cũng là đối tượng chính mà từ đó chúng ta có thể tương tác với các đối tượng UI có trên trang Web hiện hành.

Một số Selenium-WebDriver API được sử dụng để kết nối với ứng dụng Web và thực hiện các ca kiểm thử của ứng dụng Web đó, bao gồm:

API kết nối đến một trình duyệt Web: API Selenium này giúp kết nối đến một đối tượng trình duyệt Web. Câu lệnh kết nối đến một trình duyệt Web FireFox.

```
WebDriver firefoxDriver = new FirefoxDriver();
```

Sau khi thực hiện câu lệnh trên, trình duyệt *FireFox* sẽ được khởi động. Đối tượng *firefoxDriver* được sử dụng để điều khiển trình duyệt *FireFox*. Dưới đây là một số phương thức được dùng để điều khiển trình duyệt.

Phương thức điều khiển trình duyệt truy cập tới địa chỉ Web. Câu lệnh sau thực hiện việc truy cập đến trang Web Google.

```
firefoxDriver.get("http://www.google.com");
```

Phương thức điều khiển chuyển hướng trình duyệt. Câu lệnh sau sử dụng phương thức *navigate()* để thực hiện chuyển hướng đến trang Web hoặc URL được yêu cầu.

```
firefoxDriver.navigate.to("http://www.google.com");
```

Và một số phương thức điều khiển khác như: phương thức đóng cửa sổ hiện tại của trình duyệt *Driver.close()*, phương thức thoát khỏi trình duyệt *Driver.quit()* được sử dụng để thoát khỏi trình duyệt và tất cả các cửa sổ đã mở.

API xác định vị trí các phần tử Web: Theo [9,10] Selenium hỗ trợ các kiểu bộ định vị sau: id, name, xpath, dom, identifier, link, css

id và name: Khi bạn mở một trang web bất kỳ, kích chuột phải vào đối tượng xem mã nguồn. Bạn sẽ thấy được id hoặc name của đối tượng web đó. Các đối tượng web: textbox, listbox, radio button...đều có id và name riêng của nó.

xpath: XML Path (XPath) là ngôn ngữ đóng một vai trò quan trọng trong công tác trao đổi dữ liệu giữa các computer hay giữa các chương trình ứng dụng vì nó cho phép ta lựa chọn hay sàng lọc ra những dữ liệu nào mình muốn để trao đổi hay hiển thị.

Link: Lựa chọn yếu tố liên kết trong đó chứa văn bản phù hợp với khuôn mẫu quy định.

CSS: Lựa chọn phần tử sử dụng các bộ lọc css. Ví dụ : **css=a[href="#id3"]**

DOM: Document Object Model (**DOM**) là giao diện độc lập platform và ngôn ngữ cho phép chương trình và script truy xuất động và cập nhật nội dung, cấu trúc và style của tài liệu. Ví dụ: **Dom=javascriptExpression.**

Có nhiều các phương thức khác nhau để xác định vị trí phần tử Web. Người dùng có thể lựa chọn một trong các phương thức bên dưới phù hợp với từng điều kiện và ứng dụng.

Xác định vị trí trả về một giá trị	Xác định vị trí trả về một danh sách
(1) find_element_by_id	(1) find_elements_by_name
(2) find_element_by_name	(2) find_elements_by_xpath
(3) find_element_by_xpath	(3) find_elements_by_link_text
(4) find_element_by_link_text	(4) find_elements_by_partial_link_text
(5) find_element_by_partial_link_text	(5) find_elements_by_tag_name
	(6) find_elements_by_class_name

(6) find_element_by_tag_name
 (7) find_element_by_class_name
 (8) find_element_by_css_selector

(7) find_elements_by_css_selector

Ví dụ về câu lệnh xác định vị trí phần tử Web theo ID = “txt_UserName”. Kết quả câu lệnh trên trả về vị trí phần tử Web có ID = txt_UserName.

Ví dụ về findElement_by_id

```
// <input id="txt_UserName"></input>
WebElement element=driver.findElement(By.id("txt_UserName"));
```

Thao tác với các phần tử Web: Sau khi Selenium xác định được vị trí các phần tử Web. Chúng ta sẽ thực hiện các thao tác lên phần tử Web đó thông qua một số hàm được Selenium-WebDriver cung cấp nhằm thực hiện các thao tác chuột và bàn phím đối với phần tử Web. Theo [11] có tổng hợp tất cả các thao tác được sử dụng trong Selenium Webdriver như Bảng 5.1.

Bảng 5.1. Các thao tác lên phần tử Web

Thao tác lên phần tử Web	Diễn giải
build()	Tạo ra một hành động tổng hợp có chứa tất cả các hành động và luôn sẵn sàng để thực thi.
click()	Thực hiện nhấp chuột tại vị trí hiện tại của con trỏ chuột.
click(WebElement onElement)	Nhấp chuột vào giữa các phần tử được đưa ra.
clickAndHold()	Nhấp chuột (không phát hành) tại vị trí chuột hiện tại.
clickAndHold(WebElement onElement)	Nhấp chuột (không phát hành) giữa các phần tử được đưa ra.
contextClick()	Thực hiện contextClick() tại vị trí chuột hiện tại
contextClick(WebElement onElement)	Thực hiện contextClick() tại vị trí giữa các phần tử được đưa ra.

<code>doubleClick()</code>	Thực hiện một cú nhấp đúp vào vị trí chuột hiện tại.
<code>doubleClick(WebElement onElement)</code>	Thực hiện một cú nhấp đúp tại vị trí giữa các phần tử được đưa ra.
<code>dragAndDrop(WebElement source, WebElement target)</code>	Một phương pháp tiện lợi mà thực hiện bấm và giữ ở vị trí của các phần tử nguồn, di chuyển đến vị trí của phần tử đích, sau đó nhả chuột.
<code>dragAndDropBy(WebElement source, int xOffset, int yOffset)</code>	Một phương pháp tiện lợi mà thực hiện bấm và giữ ở vị trí của các phần tử nguồn, di chuyển dựa vào tọa độ x y được truyền, sau đó nhả chuột.
<code>keyDown(Keys theKey)</code>	Thực hiện nhấn một phím bấm sửa đổi.
<code>keyDown(WebElement element, Keys theKey)</code>	Thực hiện một sửa đổi lần nhấn phím sau khi chọn vào một phần tử.
<code>keyUp(Keys theKey)</code>	Thực hiện sửa đổi một key được phát hành
<code>keyUp(WebElement element, Keys theKey)</code>	Thực hiện sửa đổi một key được phát hành sau khi đã chọn được một phần tử
<code>moveByOffset(int xOffset, int yOffset)</code>	Di chuyển chuột từ vị trí hiện thời (hoặc từ tọa độ 0,0) sang một tọa độ mới được truyền vào.
<code>pause(long pause)</code>	Tạm Dừng
<code>perform()</code>	Phương pháp tiện lợi cho việc thực thi một hành vi nào đó mà không nhất thiết phải gọi đến <code>build()</code> trước.
<code>release()</code>	Thả chuột trái tại vị trí chuột hiện tại.
<code>release(WebElement onElement)</code>	Thả chuột trái tại giữa các vị trí của phần tử được truyền vào.
<code>sendKeys(java.lang.CharSequence... keysToSend)</code>	Gửi giá trị đến các phần tử đang hoạt động.
<code>sendKeys(WebElement element,</code>	Tương đương với câu lệnh

java.lang.CharSequence... keysToSend)	Actions.click(element).sendKeys(keysToSend). Phương pháp này thì khác với câu lệnh bên dưới Phương pháp này thì khác với câu lệnh bên dưới WebElement.sendKeys(CharSequence...)
---------------------------------------	--

Ví dụ với thao tác Click()

```
// Tìm một phần tử Web Checkbox bằng name, có giá trị là GioiTinh
WebElement element = driver.findElement(By.name("GioiTinh"));

// Chọn ô Checkbox đầu tiên
element.get(0).click();
```

Với những API thông dụng như đã kể ở trên, chúng ta dễ dàng kiểm thử trang Web bằng việc kết nối đến trang Web và tiến hành chạy các ca kiểm thử liên quan đến các phần tử Web trên trang Web đó.

Bằng cách sử dụng những API thông dụng như đã nêu ở trên và áp dụng cho các phần tử Web đã được định danh, Selenium dễ dàng kết nối đến trang Web cần kiểm thử và lấy các thông tin về các phần tử của trang Web, thao tác chuột và bàn phím lên các phần tử của trang Web đó. Như vậy, chỉ cần cung cấp các ca kiểm thử cho Selenium, Selenium sẽ thực thi hệ thống theo kịch bản kiểm thử tương ứng.

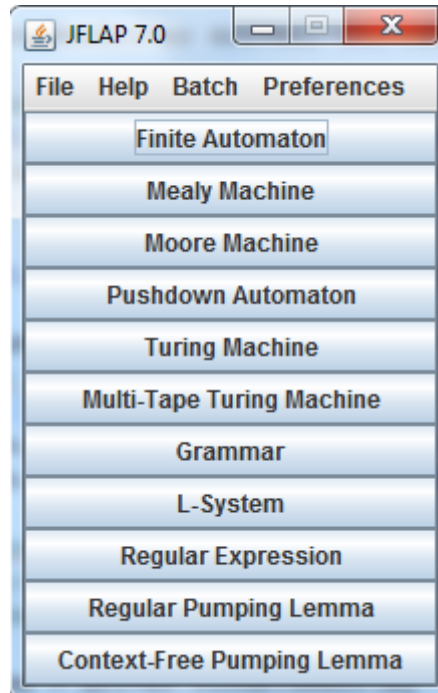
5.1.2. Công cụ JFLAP

Công cụ JFLAP được sử dụng như một tiện ích mở rộng phục vụ cho công việc đặc tả các hành vi của hệ thống dựa trên mô hình. JFLAP là một phần mềm mã nguồn mở, trong đó người dùng có thể biểu diễn các ô-tô-mát hoặc các máy hữu hạn trạng thái. Trong phần này, chúng tôi sẽ mô tả về lợi ích khi sử dụng công cụ JFLAP [8].

Như đã mô tả tệp excel đầu vào tại mục 3.3, việc đưa ra tệp excel cho đầu vào của công cụ nếu làm một cách thủ công, bước đầu tiên sẽ phải vẽ một mô hình từ một công cụ vẽ mô hình nào đó. Sau khi có mô hình, người sử dụng sẽ phải chuyển toàn bộ mô hình đó sang tệp tin excel bao gồm bốn bảng html_id, event, state và transition. Việc mô tả thủ công này tương đối phức tạp, đặc biệt đối với những người sử dụng khi mô tả tệp excel những lần đầu tiên, việc xảy ra sai sót là rất cao, xác suất xảy ra lỗi nhiều. Do đó, việc áp dụng công cụ JFLAP phục vụ cho việc vẽ mô hình và sinh tệp excel đầu vào là vô cùng hữu ích. Dưới đây, nghiên cứu sẽ trình bày về cách tạo ra tệp excel đầu vào.






5.1.2.1. Màn hình biểu diễn

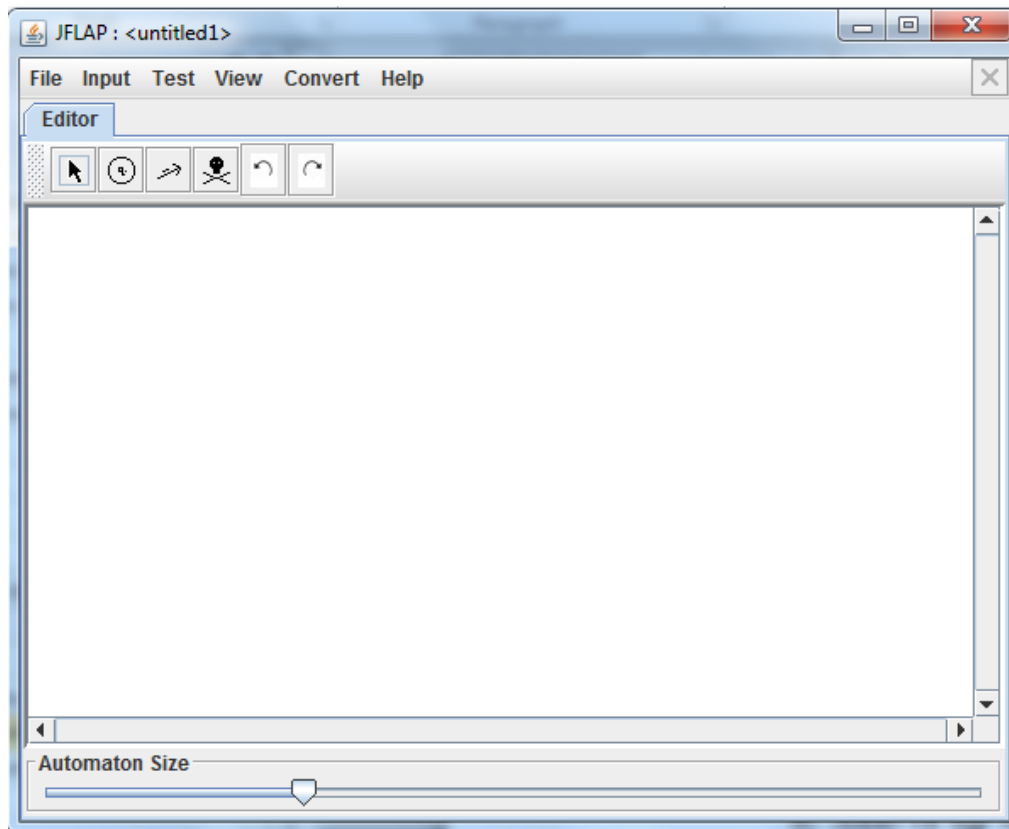
Công cụ JFLAP thì hỗ trợ rất nhiều các chức năng. Tuy nhiên, nghiên cứu chỉ sử dụng chức năng Finite Automaton để thực hiện.



Sau khi chọn chức năng Finite Automata, màn hình chính xuất hiện với các thanh công cụ hỗ trợ thiết kế mô hình.


Bảng 5.2. Các công cụ trên JFLAP


Công cụ	Giải thích
	<i>Attribute Editor</i> - Công cụ xét trạng thái bắt đầu và trạng thái kết thúc.
	<i>State Creator</i> - Công cụ tạo trạng thái (state) mới
	<i>Transition Creator</i> - Công cụ tạo đường chuyển (transition)
	<i>Deletor tool</i> - Công cụ xóa trạng thái (state) và đường chuyển (transition)
	<i>Undo - Redo</i> - Công cụ quay lại bước trước và đi đến bước sau

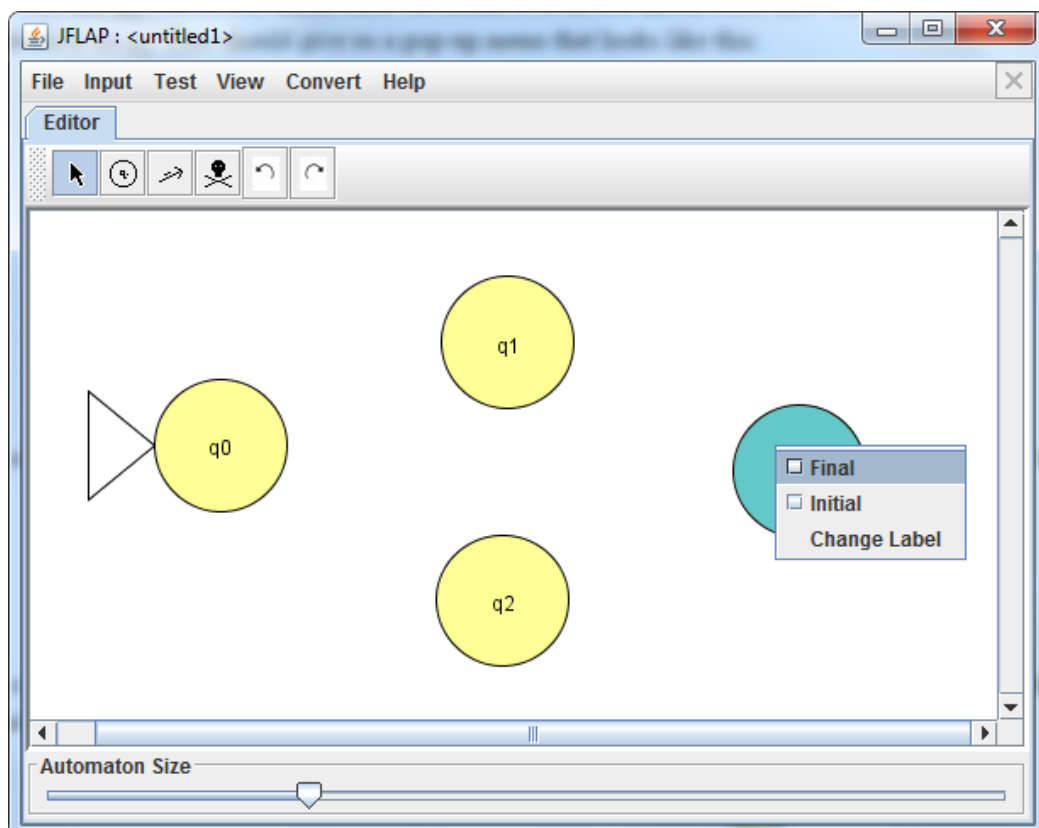
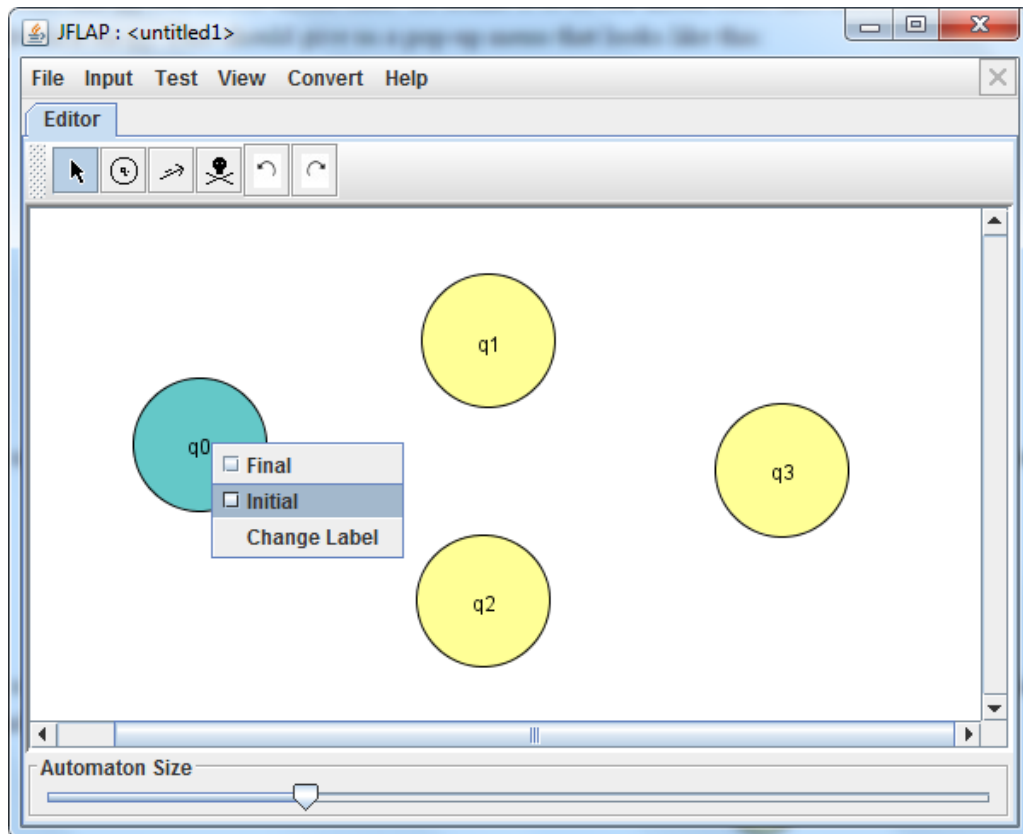


5.1.2.2. Tạo các trạng thái và tạo các đường chuyển trạng thái

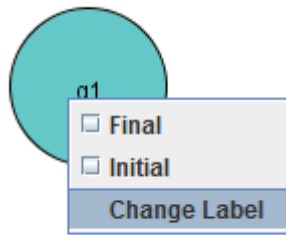
1) Tạo các trạng thái (State)

Để tạo được các trạng thái, người dùng sử dụng công cụ . Người dùng nhấn chuột vào các vị trí khác nhau trong phần khung trống của màn hình, vị trí như mong muốn.

Sau đó, cần xác định trạng thái đầu và trạng thái kết thúc. Để xác định được, người dùng sử dụng công cụ . Người dùng muốn chọn trạng thái nào làm trạng thái đầu thì nhấn chuột phải vào trạng thái đó. Sau đó chọn checkbox “Initial”. Tương tự với trường hợp xác định trạng thái kết thúc, người dùng nhấn chuột phải vào trạng thái mong muốn, chọn checkbox “Final”.



Ngoài ra, đối với các trạng thái, người dùng có thể thay đổi tên bằng cách nhấn chuyển phải và chọn “Change label”



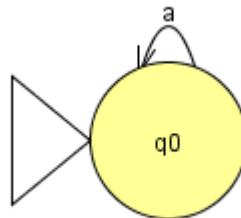
2) Tạo các đường chuyển trạng thái (Transition)

Với bộ công cụ như đã nêu trong mục 1) , người dùng có thể sử dụng để tạo các đường chuyển trạng thái (transition) giữa các trạng thái với nhau.

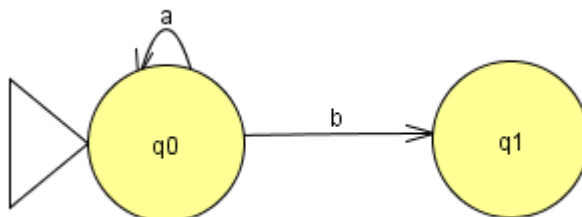
Ta có thể tạo transition cho chính trạng thái đó (self-transition) hoặc tạo transition từ trạng thái nọ sang trạng thái kia bằng cách nhấn vào công cụ *Transition*

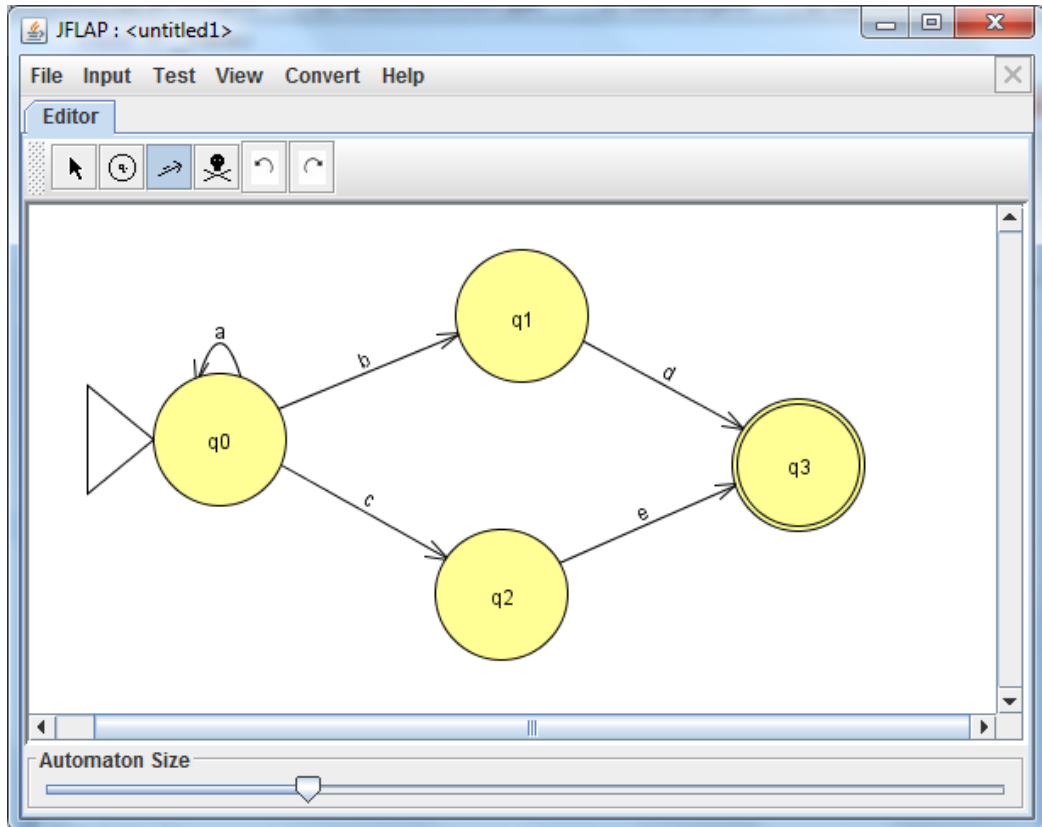


Creator trên thanh công cụ. Để tạo một transition cho chính trạng thái đó, ta nhấn vào trạng thái đó. Người dùng nhập tên trạng thái vào khung textbox, giả sử là transition a. Như vậy ta sẽ có một self-transition a cho trạng thái vừa chọn.



Tương tự trong trường hợp self-transition, người dùng có thể tạo transition giữa hai trạng thái, bằng cách nhấn chuột vào trạng thái thứ này rồi giữ và kéo sang trạng thái kia. Người dùng sẽ nhập tên của transition vào khung textbox. Như vậy, ta đã có một transition giữa hai trạng thái.





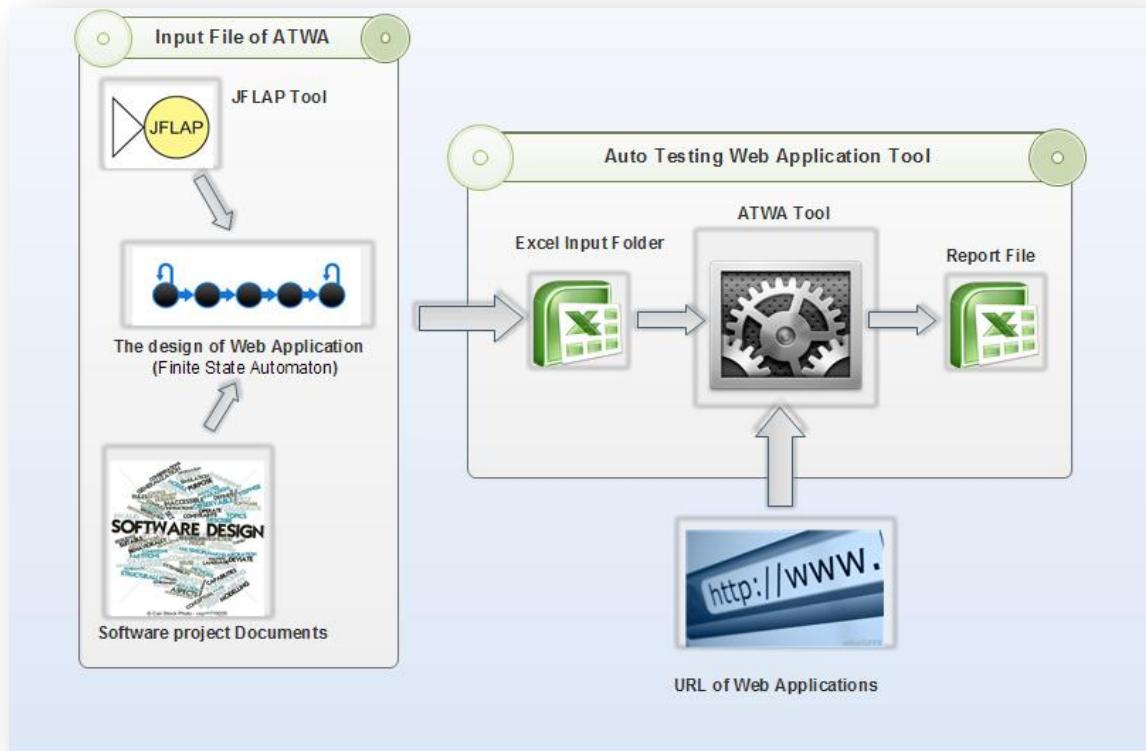
5.2 Giới thiệu công cụ kiểm thử tự động tương tác giao diện cho các ứng dụng Web

Công cụ kiểm thử tự động tương tác giao diện người dùng cho các ứng dụng Web được phát triển và đề xuất bởi [3]. Công cụ được phát triển bằng ngôn ngữ Java và sử dụng phương pháp đặc tả hình thức giao diện trang Web bằng ô-tô-mát hữu hạn trạng thái và phương pháp duyệt đồ thị để sinh các ca kiểm thử. Đầu vào của công cụ là một bộ các tệp excel được sinh ra từ mô hình. Mô hình của tệp tin excel đầu vào sẽ được người thiết kế dựa trên các hành vi hoặc các tài liệu thiết kế để thực hiện. Đầu ra của công cụ là một bộ các đường dẫn kiểm thử, bộ đường dẫn kiểm thử này được lưu trong tệp excel và chính là tệp báo cáo kết quả kiểm thử. Trong tệp báo cáo này, ngoài việc chi tiết mô tả các đường dẫn kiểm thử, công cụ còn đưa ra đánh giá kiểm thử sinh ra là PASS hay FAIL. FAIL có nghĩa là đường dẫn đó không được thực hiện, nguyên nhân không thực hiện được sẽ được mô tả chi tiết, và chỉ ra rõ ràng. PASS là đường dẫn thực hiện thành công và đúng như kết quả mong đợi.

Chi tiết về kiến trúc của công cụ, đầu vào, và đầu ra của công cụ sẽ được mô tả rõ trong các mục bên dưới.

5.2.1 Kiến trúc của công cụ

Kiến trúc của công cụ kiểm thử tự động tương tác giao diện các ứng dụng Web (Auto Testing Web Application) được mô tả như trong hình 5.2.



Hình 5.2. Kiến trúc của công cụ Auto Testing Web Application (ATWA)

Công cụ được chia làm hai phần chính, phần thứ nhất là đầu vào *Input File of ATWA* và phần xử lý của công cụ *Auto Testing Web Application Tool*

Input File of ATWA

Để công cụ có thể hiểu được và sinh các bộ kiểm thử một cách tự động, cần phải thiết kế bộ đầu vào cho công cụ. Như chương 1 đã giới thiệu, kiểm thử tự động các ứng dụng Web trong luận văn này dựa vào mô hình. Do đó, việc đầu tiên người kiểm thử cần thiết kế một mô hình ô-tô-mát hữu hạn mô tả các trạng thái của trang Web. Để thực hiện mô tả ứng dụng của Web cần phải sử dụng các tài liệu của dự án *Software project Documents* như: tài liệu đặc tả (*SRS*), tài liệu thiết kế các màn hình (*Screen Design*), tài liệu thiết kế chi tiết (*Details Design*), tài liệu thiết kế luồng của các màn hình (*Screen Flows*).... Công cụ hỗ trợ vẽ mô hình và sinh bộ đầu vào cho công cụ ATWA sử dụng trong luận văn này chính là công cụ JFLAP (*JFLAP Tool*) (giới thiệu về Selenium tại mục 5.1.2) . Sau khi mô hình được tạo, công cụ JFLAP hỗ trợ việc sinh ra bộ Excel làm đầu vào cho ATWA.

Auto Testing Web Application Tool

Công cụ ATWA nhận đầu vào là địa chỉ của một ứng dụng Web hoặc mô-đun của ứng dụng Web (*URL of Web Applications*) đó và một thư mục chứa các tệp tin excel (*Excel Input Folder*) tương ứng với các bản đặc tả tương tác giao diện của mỗi trang Web được hỗ trợ bởi công cụ JFLAP. Công cụ JFLAP hỗ trợ người dùng sinh các tệp tin excel tương đương như một hoặc nhiều máy hữu hạn trạng thái. Một trang Web thường có nhiều máy hữu hạn trạng thái. Công cụ ATWA sẽ sử dụng thuật toán ghép nối tuần tự để ghép các máy trạng thái tương ứng của các màn hình chức năng thành một máy trạng thái hoàn chỉnh cho cả mô-đun hoặc cả hệ thống đó.

Thuật toán sinh tự động các ca kiểm thử được tiến hành ngay sau đó để sinh ra các đường dẫn kiểm thử từ máy trạng thái hoàn chỉnh có được ở bước trước. Các đường dẫn kiểm thử thỏa mãn điều kiện: Trạng thái bắt đầu và trạng thái kết thúc của đường dẫn kiểm thử tương ứng với trạng thái khởi đầu và trạng thái kết thúc của máy trạng thái [3].

Selenium WebDriver

Công cụ sử dụng Selenium framework được tích hợp trong ATWA để kết nối với mô-đun hoặc hệ thống cần kiểm thử thông qua địa chỉ đã được nhận ở đầu vào của công cụ. Sau đó, các đường dẫn kiểm thử được chạy lần lượt trên giao diện Web, kết quả được đưa ra ở tệp tin Excel báo cáo *Report File* xem đường dẫn nào thành công và đường dẫn nào thất bại.

5.2.2 Đầu vào của công cụ

Từ kiến trúc của công cụ như đã nêu tại mục 5.2.1 ta sẽ đi từng phần theo như đã mô tả. Đầu tiên, dựa vào các tài liệu của hệ thống sử dụng công cụ JFLAP xây dựng mô hình ô-tô-mát hữu hạn trạng thái và xuất tệp tin Excel. Sau đó, để công cụ ATWA có thể đọc được các tệp tin này chúng ta cần sắp xếp và bố cục các tệp tin.

5.2.2.1 Xuất tệp tin Excel từ mô hình đã xây dựng từ công cụ JFLAP

Theo như công cụ phát triển tại [3], công cụ JFLAP đã thực hiện phát triển thành công việc xuất tệp tin excel đầu vào của công cụ. Tuy nhiên, tệp tin excel đầu vào mới chỉ xuất ra duy nhất một bảng transition, chưa có tổng các đường chuyển trạng thái (transition). Sau khi thực hiện xong, tệp tin excel chưa được sử dụng ngay mà cần phải bổ sung thêm 3 bảng khác bằng cách thêm thủ công. Công việc thực hiện thủ công tiềm ẩn nhiều lỗi sai và khiến kiểm thử viên rất mất thời gian để sửa chữa và chạy lại công cụ kiểm thử. Tại luận văn này, tôi đã nghiên cứu và đề xuất cải tiến công cụ bằng cách từ mô hình có thể sinh ra được tệp tin excel hoàn chỉnh có đầy đủ tất cả bốn bảng: Bảng *Element_html* (bảng các phần tử Web), Bảng *State* (bảng trạng thái), Bảng *Event* (Sự kiện), Bảng *Transition* (Sự chuyển trạng thái). Tệp tin excel sẽ được sử dụng để

làm đầu vào cho công cụ kiểm thử tự động mà chỉ cần thêm đúng giá trị cần kiểm thử. Cụ thể về cách định nghĩa mô hình và cách mô tả, đặt tên cho các trạng thái và đường chuyển trạng thái sẽ được mô tả rõ trong hai mục sau đây:

1) Quy tắc đặt tên cho trạng thái và đường chuyển trạng thái

Theo như định nghĩa của tệp tin excel trong mục 3.5. Tệp tin excel gồm có bốn bảng: bảng *Element_html*, bảng *State*, bảng *Event*, bảng *Transition*. Mỗi một bảng đều có chứa các phần tử cần thiết để công cụ kiểm thử tự động có thể lấy để sinh đường dẫn kiểm thử và kiểm thử tự động.

Bảng *Element_html* (bảng các phần tử Web): Bảng này được thiết kế để đặc tả các phần tử Web của trang Web cần kiểm thử. Bảng bao gồm *html_id* và các kiểu dữ liệu của *html_id* đó. Mỗi một phần tử Web đều có kiểu dữ liệu để công cụ kiểm thử có thể phân biệt được. Để lấy được các element này người thiết kế mô hình (hoặc kiểm thử viên) thực hiện định nghĩa trên trạng thái của mô hình.

Cách định nghĩa trên trạng thái như sau:

Định nghĩa 5.1: Cách định nghĩa trạng thái

Tên_trạng_thái|element_html_id|kiểu_phần_tử

Ví dụ: UserName|txt_UserName|textbox

Theo như bảng 5.3. , cột *Value* (giá trị) sẽ không được định nghĩa trong mô hình thiết kế. Sau khi công cụ JFLAP xuất ra tệp tin excel, *element_html_id*, *kiểu_phần_tử* sẽ được bóc tách từ các trạng thái và điền vào bảng 5.3 tương ứng với các cột *html*, *type*. Tại cột *Value*, giá trị không được điền tự động, người thiết kế sẽ tự điền các giá trị cần kiểm thử vào cột này. Tại dòng *Tổng*, công cụ sẽ tự động tính số lượng của các *element_html_id*, và cũng tự động đánh số id (định danh) cho từng *html_id* này.

Bảng 5.3. Bảng *Element_html*

Tổng			
id	html	type	Value
Định danh	element_html_id	kiểu của phần tử	Giá trị của phần tử

Bảng *State* (bảng trạng thái): là bảng mô tả các trạng thái của trang Web. Tương tự như Bảng *Element_html_id*, bảng này mô tả các trạng thái, do đó dựa trên định nghĩa 5.1 đã nêu, công cụ JFLAP sẽ bóc tách lấy *Tên_trạng_thái* sau đó điền vào cột *name* ở Bảng 5.4. Các cột còn lại tương ứng với định danh (id) của bảng 5.3

Bảng Element_html . Phần giá trị giữa các trạng thái với *element_html_id* sẽ được điền bằng tay sau khi công cụ JFLAP xuất Bảng 5.4. Tại dòng *Tổng*, công cụ sẽ tự động tính số lượng của các *element_html_id*, và cũng tự động đánh số *State* cho bảng. Trạng thái đầu tiên được quy định mang giá trị 0, trạng thái kết thúc được quy định mang giá trị 1, và giá trị không phải mở đầu và kết thúc được quy định là giá trị 5.

Bảng 5.4. Bảng Sate (bảng trạng thái)

Tổng								
State	name	IDelement 0	IDelement n
Kiểu trạng thái	Tên trạng thái							

Bảng Event (Sự kiện): Bảng liệt kê tất cả các sự kiện diễn ra trên trang Web. Mỗi sự kiện là một tương tác người dùng lên một phần tử Web. Bảng bao gồm *Event* (*định danh*), *name*, *html*, *action*. Để lấy được các element này người thiết kế mô hình (hoặc kiểm thử viên) thực hiện định nghĩa trên đường chuyển trạng thái (transition) của mô hình.

Cách định nghĩa trên đường chuyển trạng thái như sau:

Định nghĩa 5.2: Cách định nghĩa sự kiện

Tên_sự_kiện|element_html_id|hành_vi

Ví dụ: add_UserName|txt_UserName|addtext

Theo như bảng 5.4. , với cách định nghĩa trên đường chuyển trạng thái (transition) như vừa nêu, công cụ JFLAP thực hiện bóc tách phần *Tên_sự_kiện* và điền vào cột *name*, *element_html_id* sẽ được bóc tách và điền vào cột *html* tương ứng, *hành_vi* sẽ được bóc tách và điền vào cột *action*. Tại dòng *Tổng*, công cụ sẽ tự động tính số lượng của các *Event* và cũng tự động đánh số định danh cho từng *Event* sau đó điền vào cột *Event*.

Bảng 5.5. Bảng Event (bảng sự kiện)

Tổng			
Event	name	html	action

Định danh	Tên sự kiện	Thuộc tính của phần tử Web cần tương tác	Hành vi của sự kiện
-----------	-------------	--	---------------------

Bảng Transition (Sự chuyển trạng thái): Bảng này chứa toàn bộ các *transition* của trang Web. Với các trạng thái (State), và các đường chuyển trạng thái (transition) đã được công cụ bóc tách từ mô hình trạng thái. Công cụ sẽ tự động sinh ra bảng chuyển trạng thái như Bảng 5.6.

Bảng 5.6. Bảng Transition (Sự chuyển trạng thái)

Tổng									
name	Trạng thái 0	Trạng thái 1	Trạng thái n
Trạng thái 0	event 00	event 01
Trạng thái 1
...
...
...
...
...
Trạng thái n	event n(n-1)	event nn

2) Xuất tệp tin excel

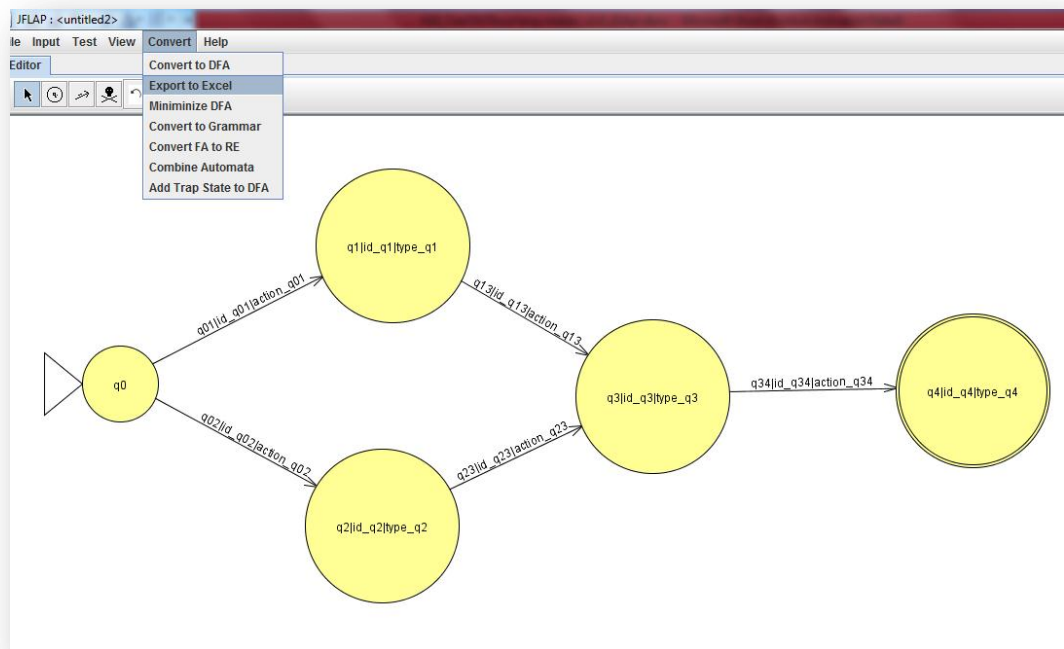
Người thiết kế sau khi đã tạo máy trạng thái thành công, sử dụng công cụ JFLAP ta có thể xuất nội dung ra tệp tin *excel* cho thiết kế đầu vào của công cụ kiểm thử tự động Web. Chúng ta lấy một ví dụ để minh họa cho việc xuất tệp tin excel như Hình 5.5 với các trạng thái được định nghĩa như sau:

- Trạng thái khởi tạo q0 không tương tác với html_id nào
- Trạng thái q1, tương tác với id_q1 và có kiểu dữ liệu là type_q1
- Tương tự như vậy với các trạng thái q2, q3 và trạng thái kết thúc q4
- Đường chuyển trạng thái giữa q0 và q1 có tên là q01, đường chuyển trạng thái này tương tác với id_q01 và có hành vi tương tác là action_q01
- Tương tự như vậy với các đường chuyển trạng thái khác.



Hình 5.3. Ví dụ về một FA với cách định nghĩa như mục a.

Để xuất ra tệp tin *excel*, trên thanh menu ta chọn “Convert/Export to excel”.



Hình 5.4. Xuất ra tệp tin excel.

Sau khi thực hiện xuất ra tệp tin excel từ mô hình Hình 5.4., tệp tin excel được hiển thị như Hình 5.5.

The screenshot shows an Excel spreadsheet with the following data:

Element_html	id	html_id	type	value
	0	id_q1	type_q1	
	1	id_q2	type_q2	
	2	id_q3	type_q3	
	3	id_q4	type_q4	

State	name	0	1	2	3
0	q0				
5	q1				
5	q2				
5	q3				
1	q4				

Event	name	html_id	action
1	q01	id_q01	action_q01
2	q02	id_q02	action_q02
3	q13	id_q13	action_q13
4	q23	id_q23	action_q23
5	q34	id_q34	action_q34

Transition	name	q0	q1	q2	q3	q4
	q0		q01	q02		
	q1				q13	
	q2				q23	
	q3					q34
	q4					

Hình 5.5. Tập tin excel sau khi được xuất từ công cụ JFLAP

Như vậy, tập tin đầu vào của công cụ đã gần như hoàn chỉnh. Để thực hiện chạy công cụ kiểm thử với tập tin đầu vào này, kiểm thử viên sẽ thực hiện thêm các tổng của các giá trị kiểm thử, và giá trị cần kiểm thử vào cột *value* của bảng *element_html* và các cột *id* tương ứng với *html_id* của bảng State như Hình 5.6. (phần bôi vàng)

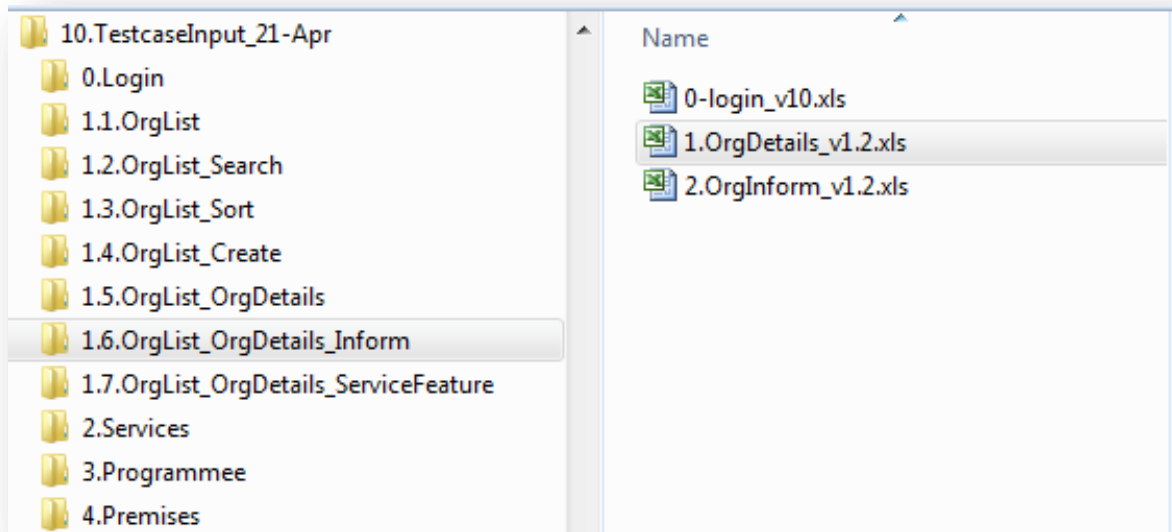
4				3		
Element	id	html_id	type	value_1	value_2	value_3
	0	id_q1	type_q1	Fill Value1 here	Fill Value2 here	Fill Value3 here
	1	id_q2	type_q2	Fill Value1 here		
	2	id_q3	type_q3	Fill Value1 here		
	3	id_q4	type_q4	Fill Value1 here	Fill Value1 here	Fill Value1 here
5						
State	name	0	1	2	3	
0	q0					
5	q1	o				
5	q2		o			
5	q3	o	o	o		
1	q4				o	
5						
Event	name	html_id	action			
1	q01	id_q01	action_q01			
2	q02	id_q02	action_q02			
3	q13	id_q13	action_q13			
4	q23	id_q23	action_q23			
5	q34	id_q34	action_q34			
5						
Transition	name	q0	q1	q2	q3	q4
	q0		q01	q02		
	q1				q13	
	q2				q23	
	q3					q34
	q4					

Hình 5.6. Tập tin excel đầu vào sau khi được điền giá trị kiểm thử

5.2.2.2 Cách đặt tên cho các tập tin Excel

Tập tin excel đầu vào là kết quả của mô hình ô-tô-mát hữu hạn trạng thái đặc tả trang Web mà mục 3.5 và mục 5.2.2.1 đã định nghĩa và mô tả. Một ứng dụng Web thông thường được đặc tả thành nhiều ô-tô-mát hữu hạn trạng thái, và nhiều máy hữu hạn trạng thái. Mỗi một ô-tô-mát hữu hạn trạng thái là một tập tin Excel. Do đó, để công cụ có thể hiểu được thứ tự thực hiện và ghép nối các ô-tô-mát hữu hạn trạng thái này, các tập tin excel được đặt trong các thư mục để và đánh số các thư mục theo chức năng như một bộ kiểm thử (test suite).

Ví dụ, Hình 5.7, chúng tôi lưu trữ các tập tin Excel đặc tả ứng dụng Web vào các thư mục con, mỗi thư mục con tương ứng với một chức năng của ứng dụng. Bên trái của Hình 5.7 là thư mục cha chứa toàn bộ các thư mục con. Bên phải Hình 5.7 là một thư mục con chứa các tập tin Excel đặc tả một chức năng và được đánh số theo thứ tự thực hiện của người dùng. Cách lưu trữ này phục vụ cho việc kiểm thử từng chức năng riêng lẻ của ứng dụng Web.

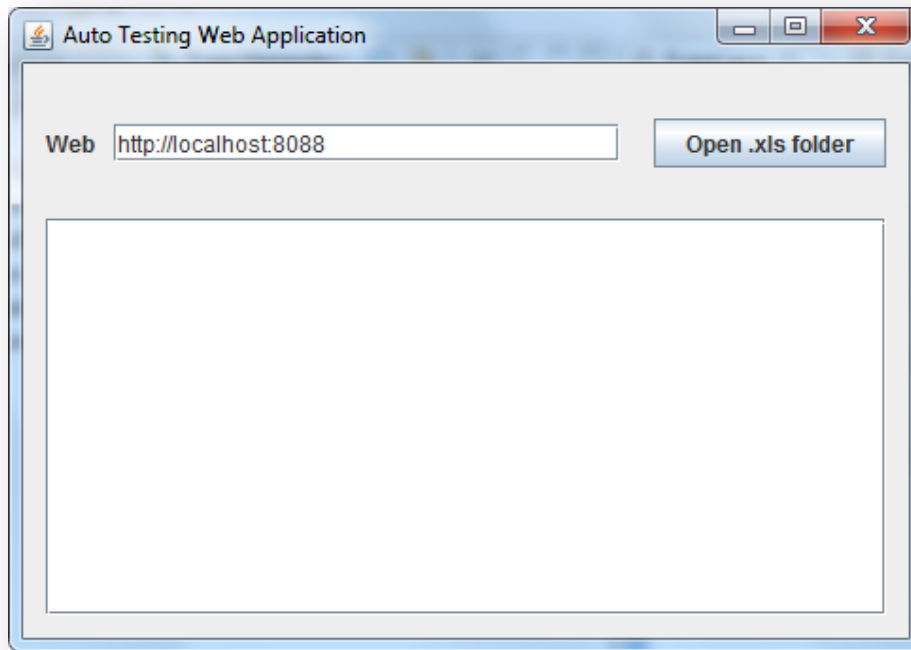


Hình 5.7. Lưu trữ các tệp tin đầu vào

Công cụ sẽ nhận thư mục chứa các tệp tin đặc tả và đọc chúng làm dữ liệu đầu vào. Việc đọc các tệp tin này được thực hiện lần lượt. Với mỗi tệp tin, công cụ sẽ đọc vào từng bảng một theo thứ tự cấu trúc đã được trình bày ở chương 2. Mỗi bảng là một danh sách các mảng. Mỗi mảng chứa các thông tin tương ứng với số cột của bảng. Mỗi ô giá trị của bảng trong tệp tin Excel được đọc vào và lưu vào mảng tương ứng. Sau khi đã đọc xong các tệp tin Excel, công cụ xây dựng ô-tô-mát hữu hạn trạng thái của từng tệp tin dựa vào các mối liên kết giữa các bảng trong nó bằng cách: lấy các trạng thái và các hàm chuyển trạng thái từ bảng *transition*, sau đó tìm trạng thái bắt đầu và trạng thái kết thúc thông qua mối liên kết với bảng *state*, cuối cùng là kết nối các dữ liệu đó thành một ô-tô-mát hữu hạn trạng thái. Sau khi có được các ô-tô-mát hữu hạn trạng thái tương ứng với các tệp tin đặc tả Excel, thuật toán ghép nối được áp dụng để thực hiện việc kết nối các ô-tô-mát hữu hạn trạng thái theo đúng thứ tự các tệp tin Excel thành một mô hình hoàn chỉnh cho toàn ứng dụng Web.

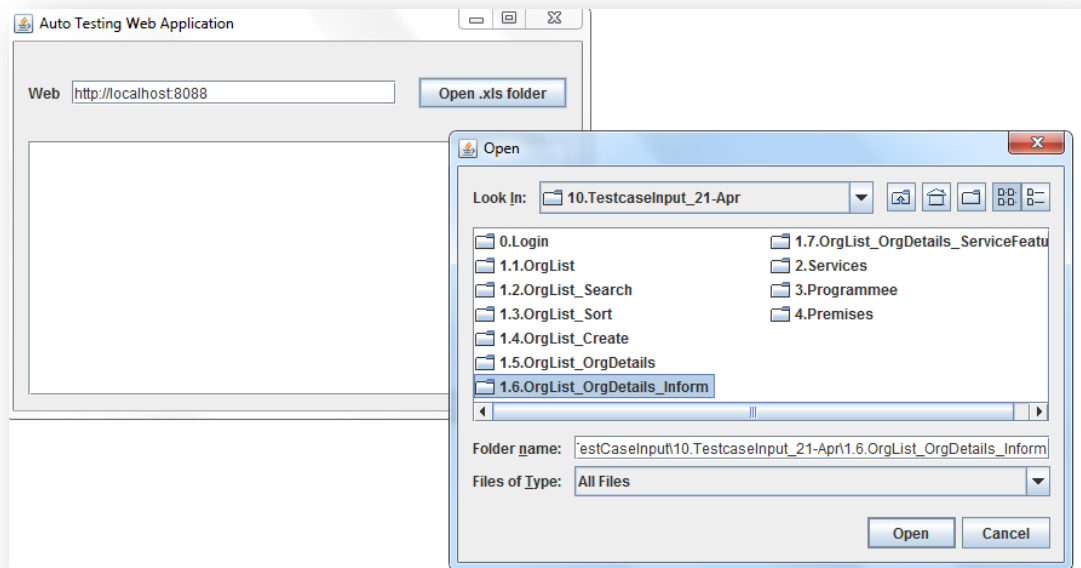
5.2.3 Giao diện và cách sử dụng công cụ ATWA

Theo Hình 5.8. người sử dụng sẽ thực hiện điền URL của trang Web cần kiểm thử vào thanh địa chỉ *Web*. Sau đó thực hiện nhấp chuột *Open .xls folder* để lấy bộ đầu vào được tạo tại mục 5.2.2



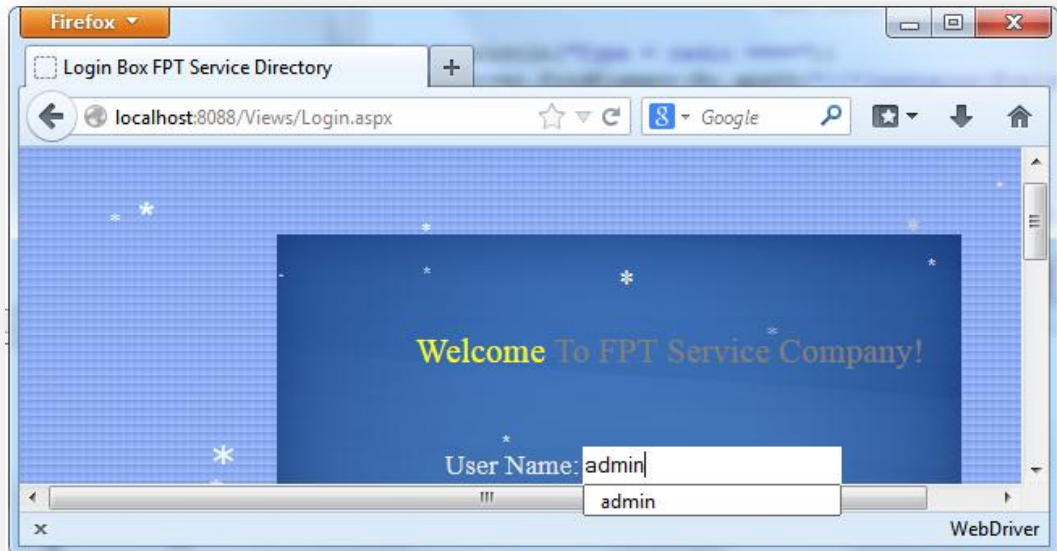
Hình 5.8. Giao diện nhập dữ liệu đầu vào của công cụ

Khoảng trống bên dưới thanh địa chỉ là phân hiển thị kết quả của công cụ. Tại vùng trống này, các đường dẫn kiểm thử sẽ được hiển thị và ghi kèm kết quả và lý do trong trường hợp không kiểm thử đường dẫn không thành công.



Hình 5.9. Chọn bộ kiểm thử để thực hiện

Sau khi chọn đường dẫn đến bộ kiểm thử, công cụ tự động kết nối các tệp tin Excel và mở trình duyệt, mở đường dẫn đã được nhập và thực hiện kiểm thử tự động. Công cụ sử dụng Selenium Webdriver để kết nối nên trong góc bên phải của thanh công cụ sẽ hiển thị chữ Webdriver (hình 5.10). Và cuối cùng, sau khi thực hiện xong, kết quả được hiển thị như Hình 5.11



Hình 5.10. Selenium Webdriver thực hiện kiểm thử tự động trên Web

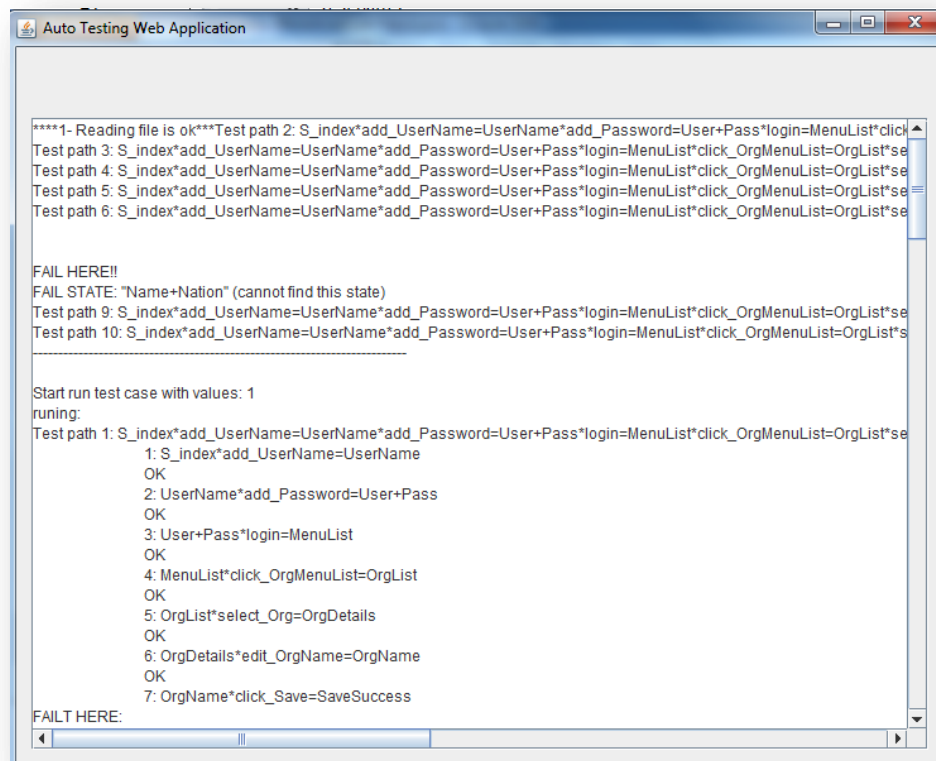
5.2.4 Đầu ra của công cụ

Sau khi thực hiện chọn và kiểm thử tự động (như hướng dẫn tại 5.2.3) công cụ sẽ xuất một tệp tin Excel. Tệp tin này có ba cột. Cột thứ nhất chứa các đường dẫn kiểm thử như đã trình bày tại Chương 4. Cột thứ hai chứa kết quả của đường dẫn kiểm thử *PASS* khi đường dẫn kiểm thử thực hiện thành công, *FAIL* khi đường dẫn kiểm thử thực hiện thất bại. Cột thứ ba nêu lý do trong trường hợp đường dẫn kiểm thử thực hiện bị thất bại.

Bảng 5.7. Các trường hợp thất bại FAIL

Trường hợp thất bại	Diễn giải
FAIL EVENT: Event couldn't do.	Thông báo được hiển thị khi đường dẫn kiểm thử không tìm được sự kiện như mô tả
FAIL STATE: "Name_state" cannot find this state.	Thông báo được hiển thị khi đường dẫn kiểm thử không tìm được trạng thái như mô tả

Real Output (“giá trị mong đợi”) and Expected Output (“giá trị thực tế”) of element “Name_element” are different	Thông báo được hiển thị khi giá trị mong đợi không giống với giá trị thực tế.
--	---



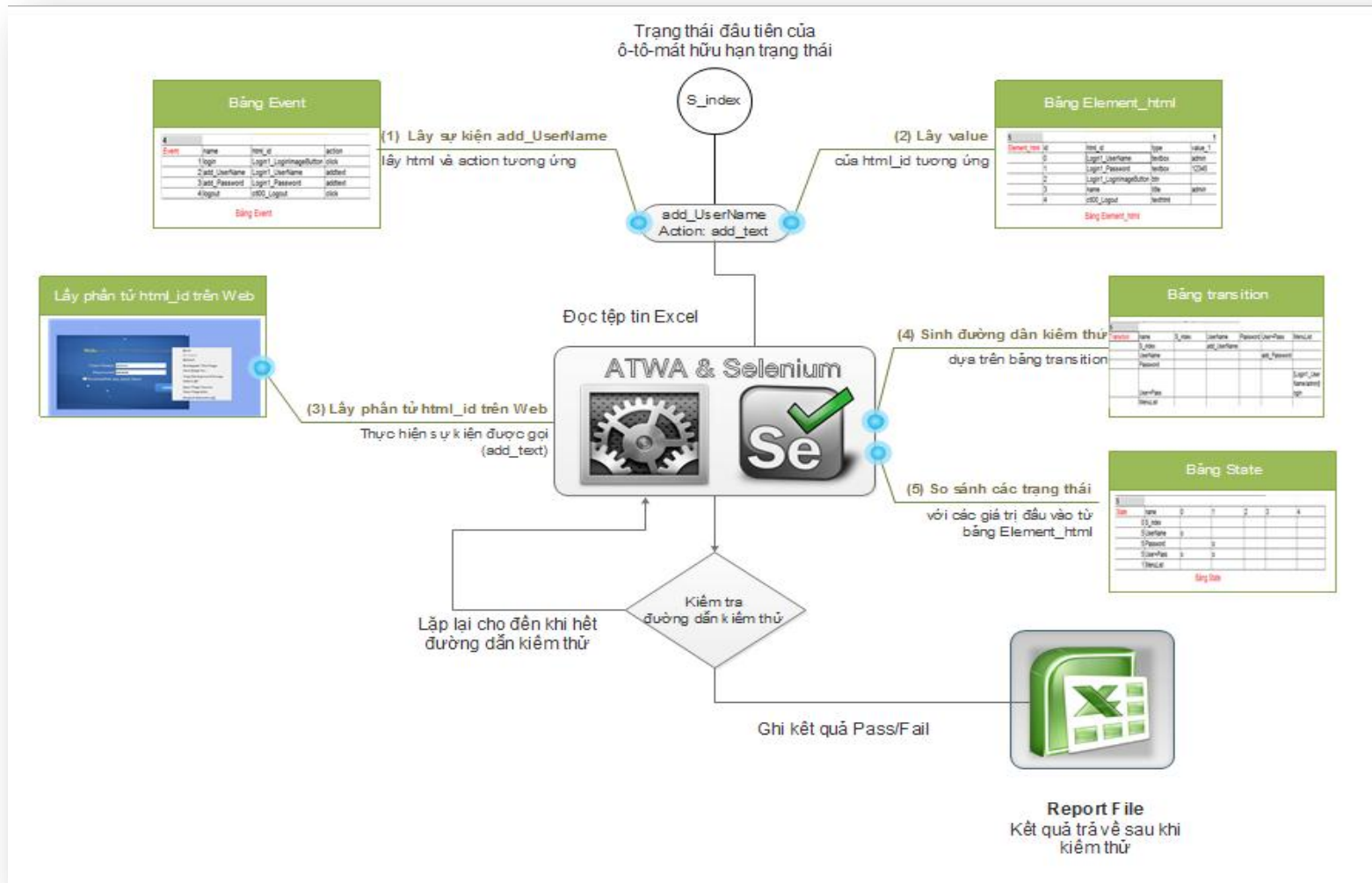
Hình 5.11. Kết quả hiển thị sau khi chạy xong bộ kiểm thử

Ví dụ về đường dẫn kiểm thử sau khi ghép nối ba ô-tô-mát hữu hạn trạng thái.

Đường dẫn kiểm thử:

S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*edit_Add1=Add1*click_Save=SaveSuccess

Công cụ thực hiện từng *transition*: *S_index * add_Username = Username* như hình 5.14. Ví dụ về hoạt động của công cụ ATWA để xuất ra tệp tin báo cáo.



Hình 5.12. Ví dụ về hoạt động của công cụ ATWA

Xuất phát từ trạng thái *S_index* là trạng thái đầu của ô-tô-mát, công cụ thực hiện sự kiện *add_UserName* với action tương ứng *add_text*

- (1) Công cụ sẽ đọc bảng *Event* và dò sự kiện *add_UserName* và lấy *html* cùng action tương ứng
- (2) Với *html_id* đã tìm tại (1), công cụ tìm value của *html_id* này trong bảng *Element_html*
- (3) Tại bước này, dựa vào công cụ Selenium, vị trí phần tử *html_id* đã tìm tại (1) sẽ được thực hiện sự kiện *add_text*
- (4) Dựa trên bảng *Transition*, công cụ ATWA đã sinh được bộ đường dẫn kiểm thử.
- (5) Tại bước (3), nếu thực hiện *add* thành công thì sẽ tiếp tục, nếu không thành công công cụ ATWA dựa vào bảng *State* để lấy các phần tử *html* được định nghĩa trong trạng thái *Username*.

Công cụ lấy giá trị của các phần tử *html* đó trên giao diện trang Web đã được thực hiện sự kiện và so sánh với giá trị được định nghĩa trong bảng *Element_html*.

- (6) Nếu so sánh không thành công thì dừng thực hiện đường dẫn (4) và đưa ra thông báo thất bại (*FAIL*) và lý do thất bại.

Nếu so sánh thành công thì tiếp tục thực hiện các *transition* cho đến hết đường dẫn kiểm thử của (4). Khi thực hiện hết đường dẫn kiểm thử thì kết luận kiểm thử thành công (*PASS*)

Kết quả này sẽ được xuất ra tệp tin Report File như Hình 5.13

5.2.5 Thực nghiệm

5.2.5.1 Giao diện của ứng dụng Web

Trong mục này, tôi sẽ giới thiệu cách áp dụng công cụ kiểm thử tự động vào ứng dụng FPT-SD Web. Đây là một trang Web quản lý các tổ chức do FPT phát triển. FPT-SD viết tắt của từ FPT - Services Directory - kho lưu trữ dịch vụ của công ty FPT. Trang Web có các chức năng chính như mô tả tại bảng 5.8 và hình 5.14 .

Bảng 5.8. Bảng chức năng chính của trang Web FPT Services

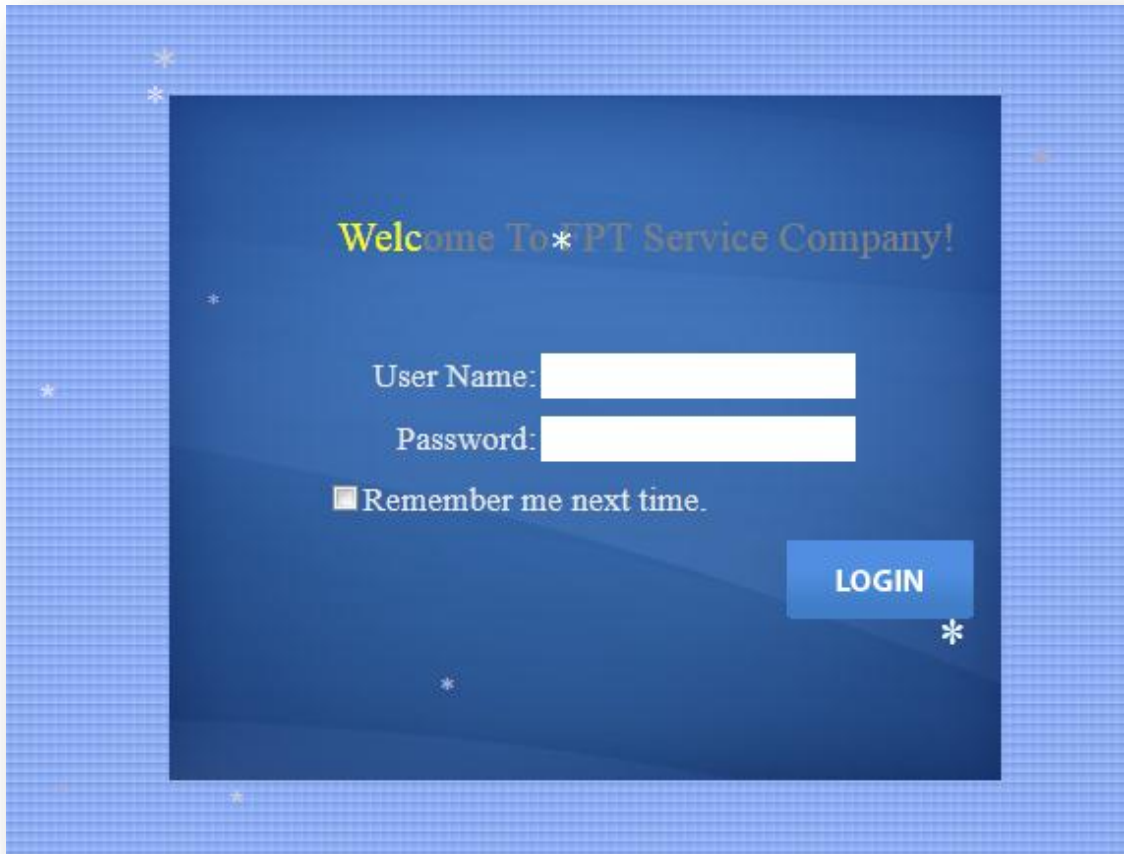
#	Tên chức năng	Mô tả
1	Logon & Logout	Cho phép người dùng đăng nhập vào hệ thống

2	Organisations	Duy trì và quản lý dữ liệu trong vùng tổ chức (Organisation) của hệ thống .
3	Services	Duy trì và quản lý dữ liệu cho các dịch vụ (Services)
4	Programmes	Bảo trì và quản lý dữ liệu của chương trình (Programme)
5	Premises	Duy trì và quản lý dữ liệu cho các cơ sở, phương tiện và người dùng
6	Geographic Data	Duy trì và quản lý dữ liệu trong khu vực địa lý của hệ thống.

	A	B	C
1	Test path 1: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*edit_OrgName=OrgName*click_Save=SaveSuccess	FAIL	FAIL EVENT:Event couldn't do.
2	Test path 2: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*select_Nation=NationCountry*click_Save=SaveSuccess	PASS	
3	Test path 3: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*check_Preffered=PreferredOrg*click_Save=SaveSuccess	PASS	
4	Test path 4: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*select_Nation=NationCountry*check_Preffered=Nation+Preferred*click_Save=SaveSuccess	PASS	
5	Test path 5: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*edit_OrgName=OrgName*select_Nation=Name+Nation*click_Save=SaveSuccess	FAIL	FAIL EVENT:Event couldn't do.
6	Test path 6: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*check_Preffered=PreferredOrg*select_Nation=Nation+Preferred*edit_OrgName=Name+Nation+Preferred	PASS	
7	Test path 7: S_index*t_username=UserName*t_password=User+Pass*t_login=Menu*click_OrgMenu=OrgList*select_Org=OrgDetails*select_Nation=NationCountry*check_Preffered=Nation+Preferred*		

Hình 5.13. Kết quả kiểm thử.

4	Search/Sort/Include In-active Organisation	Các chức năng chính trên danh sách Organisation như tìm kiếm (Search), sắp xếp (Sort), Hiện thị cả các tổ chức bị In-Active (Include In-Active)
---	--	---



Hình 5.15. Giao diện trang đăng nhập



Hình 5.16. Danh sách các chức năng (Menu List)

ORGANISATION SERVICES GEOGRAPHY PREMISES						
Organisation List						
All 0-9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z					<input type="button" value="Create"/>	<input type="checkbox"/> Include In-active
Organisation Name	Head Office Address Line1	Postcode	Contact	Is Active?		
Kiến trúc và nội thất Thời Đại	19 Maurer Court Greenwich1	SE15 OSS	Desiree Hector	Yes		
Công ty TNHH FPT Information Service	17 North Central Expwy1	SE15 OSS	Linh19 nguyen manh	Yes		
Kiến trúc và nội thất Thời Đại	19 Maurer Court Greenwich1	SE15 OSS	Desiree Hector	Yes		
Đại học FPT	17 North Central Expwy1	SE15 OSS	Linh19 nguyen manh	Yes		
Cao đẳng Thực Hành FPT	10 Maurer Court Greenwich	SE10 OSS	Nicole Angie	Yes		
Công ty TNHH Cung ứng Kiến Vàng	Đường Lê Thánh Tông (NR Nguyễn Tiến Việt), Khu Khà Lẽ, Phường Võ Cường, Thành phố Bắc Ninh, Bắc Ninh	SE10 OSS	Nicole Angie	Yes		
Công Ty TNHH Dịch vụ BVTQ Quế An	Số 32A, Khu 1, Thị trấn Phố Mới, Huyện Quế Võ, Bắc Ninh	SE10 OSS	Linh18 nguyen manh	Yes		
Data Solution IT Company	123 Holyhood Drive, Los Angeles	SE11 OSS	NganLT1 Carey	Yes		
Total System Services	13 North Central Expwy1	SE15 OSS	Desiree Hector	Yes		
Bản quản lý chương trình hợp tác với WHO	138A Giảng Võ, Phường Kim Mã, Quận Ba Đình, Hà Nội	SE14 OSS	Nicole Angie	Yes		

Hình 5.17. Giao diện các chức năng của Organisation.

Organisation List (Danh sách tổ chức): Trang Web hiển thị danh sách tất cả các tổ chức liên kết với FPT, cho phép người quản trị hệ thống thực hiện các chức năng như Bảng 5.9 và Hình 5.17

Search/Sort/Include In-active Organisation (Tìm kiếm/Sắp Xếp/ Bao gồm In-active): Các chức năng phụ của trang Danh sách tổ chức như Hình 5.18, 5.19

Organisation List					
All 0-9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z					

Hình 5.18. Chức năng tìm kiếm theo bảng chữ cái

Create và Edit Organisation (Tạo mới và sửa tổ chức): Chức năng chính của tổ chức là chức năng tạo mới và có thể sửa một tổ chức trong danh sách tổ chức. Giao diện như Hình 5.20

Trang Create và Edit Organisation chia ra làm nhiều các chức năng con hay còn gọi là các Tab thông tin như hình 5.21 - 5.25

Organisation Name
Kiến trúc và nội thất Thời Đại
Công ty TNHH FPT Information Service
Kiến trúc và nội thất Thời Đại
Đại học FPT
Cao đẳng Thực Hành FPT
Công ty TNHH Cung ứng Kiến Vàng
Công Ty TNHH Dịch vụ BVTQ Quế An
Data Solution IT Company
Total System Services
Bản quản lý chương trình hợp tác với WHO
Hitachi Solution

Hình 5.19. Chức năng sắp xếp Organisation

Organisation Details

Information Service Feature Premise Detail Materials BU/Directorates In-active Save Back

Organisation Name *	<input type="text" value="Kiến trúc và nội thất Thời Đại"/>	Preferred Organisation	<input type="checkbox"/>
Organisation Short Description *	<input type="text" value="Cung cấp các đồ nội thất gia đình và khách sạn"/>	Expression of Interest	<input type="checkbox"/>
Lead Contact *	<input type="text" value="Desiree Hector"/> ...	Type of Business *	<input type="text" value="abattoir1 (manufacture)"/> ...
Address Line 1 *	<input type="text" value="19 Maurer Court Greenwich1"/>	SIC Code	<input type="text" value="10111"/>
Address Line 2	<input type="text" value="Số 3/2 - Trần Phú - Hà Nội"/>	Organisation Full Description	<input type="text" value="Tư vấn nội thất văn phòng hiện đại
Văn phòng bài trí nội thất lịch lãm
chuyên nghiệp không chi thể hiện"/>
Address Line 3	<input type="text"/>	Phone Number *	<input type="text" value="0438636976"/>
Post Code *	<input type="text" value="SE15 OSS"/> ...	Fax	<input type="text" value="0438636977"/>
City/Town	<input type="text" value="Hà Nội"/>	Email	<input type="text" value="thoidaicompany@gmail.com.vn"/>
County	<input type="text" value="Việt Nam"/>	Web Address	<input type="text" value="http://thoidai.com.vn"/>
Nation/Country	<input type="text" value="Germany"/>	Charity Number	<input type="text" value="5423"/>
		Company Number	<input type="text" value="0384"/>

Hình 5.20. Giao diện các chức năng của Organisation.

Organisation Details

Information | **Service Feature** | Premise Detail | Materials | BU/Directories In-active Save Back

<p>Organisation Specialism</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Metal Health <input type="checkbox"/> Learning Disability <input type="checkbox"/> Dyslexia <input type="checkbox"/> Deaf/Hard of Hearing <input type="checkbox"/> Blind/Partially Sighted 	<p>Services Personal Circumstances Capabilities</p> <ul style="list-style-type: none"> <input type="checkbox"/> Lone Parent <input type="checkbox"/> Carer Responsibili
<p>Services Disabilities Capabilities</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Difficulty in hearing <input type="checkbox"/> Diabetes <input type="checkbox"/> Condition restricting <input type="checkbox"/> Chest, Breathing 	<p>Services Ethnicity Capabilities</p> <ul style="list-style-type: none"> <input type="checkbox"/> White Irish <input type="checkbox"/> White British
<p>Services Barriers Capabilities</p> <ul style="list-style-type: none"> <input type="checkbox"/> Basic skills <input type="checkbox"/> Refugee <input checked="" type="checkbox"/> ESOL <input type="checkbox"/> Lone Parent 	<p>Accreditation</p> <ul style="list-style-type: none"> <input type="checkbox"/> Accreditation 4 <input type="checkbox"/> Accreditation 3 <input type="checkbox"/> Accreditation 2 <input type="checkbox"/> Accreditation 1 <input type="checkbox"/> ISO 9001
<p>Services Benefits Capabilities</p> <ul style="list-style-type: none"> <input type="checkbox"/> Income Support <input type="checkbox"/> Incapacity Benefit <input type="checkbox"/> Employment <input type="checkbox"/> Disability living 	

Hình 5.21. Giao diện phần Service Features

Organisation Details

Information | Service Feature | **List Product** | Premise Detail | Materials | BU/Directories In-active Save Back

<p>EOI Programmes</p> <ul style="list-style-type: none"> <input type="checkbox"/> Program3 <input type="checkbox"/> Program2 <input type="checkbox"/> Program1 <input type="checkbox"/> Programme 3 <input type="checkbox"/> Programme 2 	<p>EOI Services</p> <ul style="list-style-type: none"> <input type="checkbox"/> EOI Service3 <input type="checkbox"/> EOI Services2 <input type="checkbox"/> EOI Services1 <input type="checkbox"/> Service 3 <input type="checkbox"/> Service 2
---	---

Hình 5.22. Giao diện phần List Product

Organisation Details

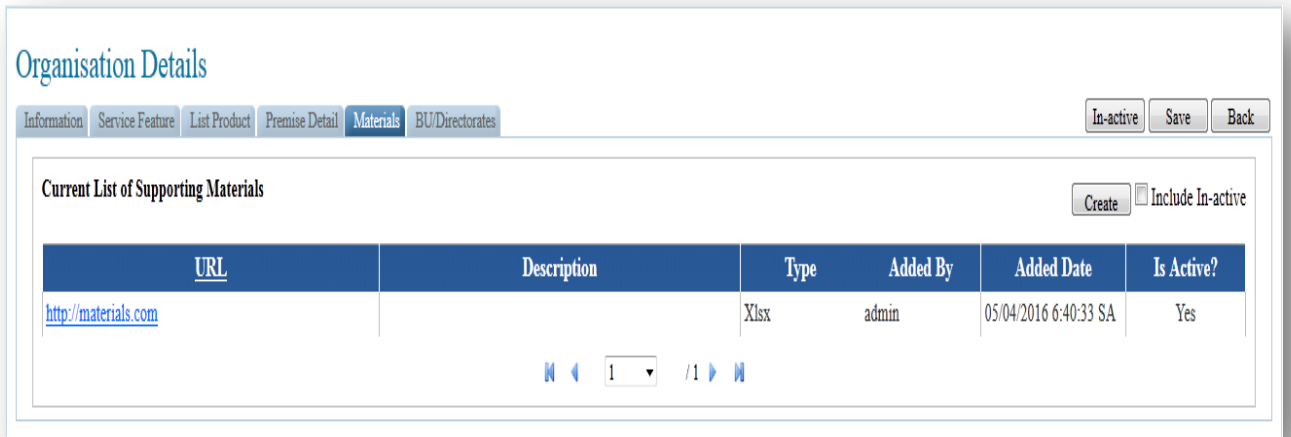
Information | Service Feature | List Product | **Premise Detail** | Materials | BU/Directories In-active Save Back

Premise

Located In

Ward:	<input type="text"/>	NHS Authority:	<input type="text"/>
Borough:	<input type="text"/>	Govt Office Region:	<input type="text"/>
Local Authority:	<input type="text"/>	Trust Region:	<input type="text"/>
Unitary Authority:	<input type="text"/>	Trust District:	<input type="text"/>

Hình 5.23. Giao diện phần Premise Detail



Hình 5.24. Giao diện phần Materials

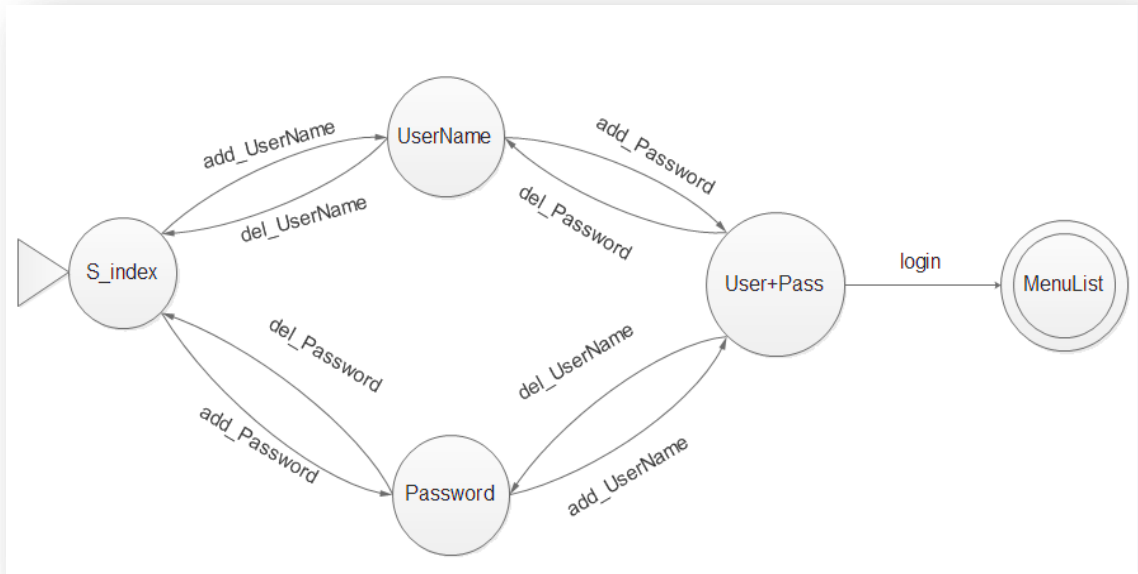


Hình 5.25. Giao diện phần Bu/Derectorates

5.2.5.2 Đặc tả mô hình Ô-tô-mát hữu hạn trạng thái

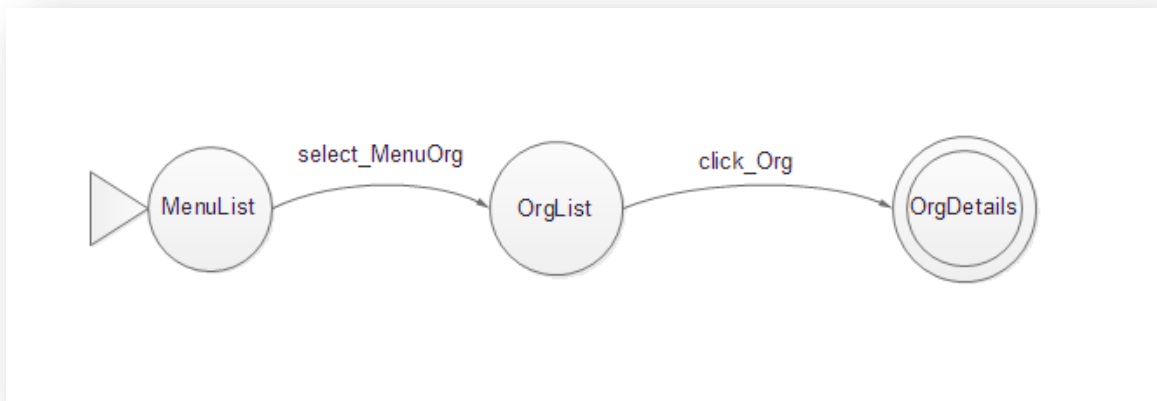
Từ giao diện được giao diện được mô tả tại mục 5.2.4.1 và mô hình ô-tô-mát hữu hạn trạng thái được chia làm ba mô hình lần lượt như các hình bên dưới.

Login



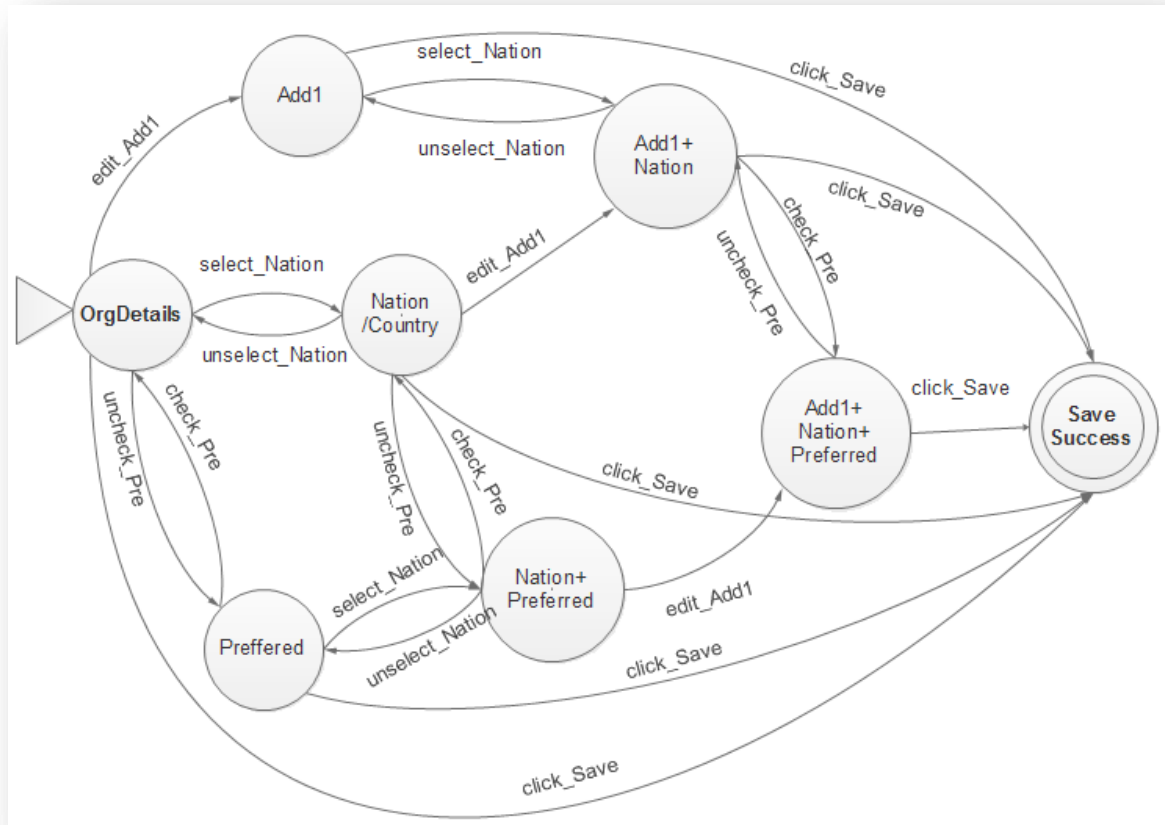
Hình 5.26. Mô hình ô-tô-mát hữu hạn trạng thái trang Login

Organisation Deatails



Hình 5.27. Mô hình ô-tô-mát hữu hạn trạng thái chức năng *Organisation Details*

Edit Organisation

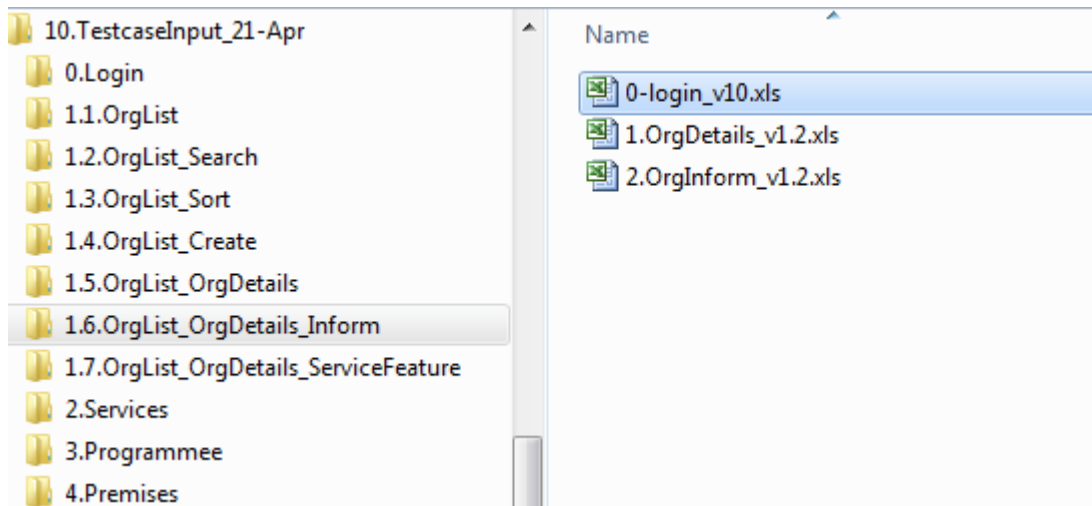


Hình 5.28. Mô hình ô-tô-mát hữu hạn trạng thái chức năng *Organisation Details - Tab Information*

5.2.5.3 Tập tin đầu vào

Hình 5.29 là thư mục chứa các bản đặc tả tương tác giao diện của chức năng *Organisation*. Thư mục này gồm 3 tập tin Excel đặc tả tương tác giao diện của 6 trang Web tương ứng như các hình 5.26, 5.27, 5.28

- (1) Trang đăng nhập - trang Web được chọn làm mốc *0-login.xls*
- (2) Trang chi tiết tổ chức *Organisation Details 2-OrgDetails.xls*
- (3) Trang Sửa *Organisation* với tab *Information 3-OrgInform.xls*

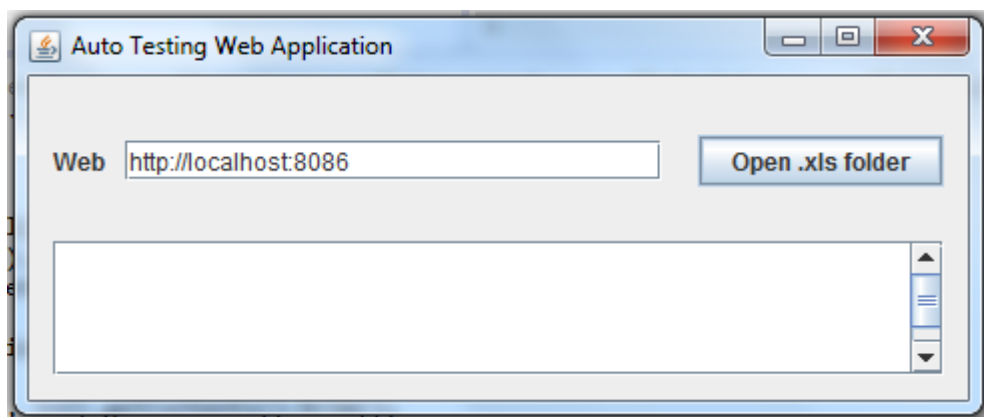


Hình 5.29. Thư mục các tệp tin đặc tả chức năng Organisation

5.2.5.4 Kết quả đầu ra

Để kiểm thử tự động ứng dụng Web FPT Services, chúng tôi chọn đầu vào cho công cụ kiểm thử tự động tương tác giao diện Web gồm: địa chỉ của ứng dụng Web và thư mục chứa các tệp tin đặc tả. Hình 5.30 là giao diện công cụ đã được chọn đầu vào. Trong giao diện này, đường dẫn ứng dụng Web Quản lý thông tin cán bộ là <http://localhost:8088> và thư mục các tệp tin đặc tả được chọn là thư mục trong hình 5.30.

Sau khi nhập dữ liệu đầu vào, công cụ sẽ thực hiện đọc các tệp tin đầu vào và tiến hành mô hình hóa hệ thống để tạo ra các ô-tô-mát hữu hạn trạng thái tương ứng. Tiếp theo, công cụ sẽ thực hiện thuật toán ghép nối như đã trình bày ở chương 3 để tạo ra ô-tô-mát hữu hạn trạng thái cho toàn ứng dụng.



Hình 5.30. Giao diện của công cụ

Áp dụng thuật toán 4.1 và thuật toán 4.2 đã nêu tại mục 4.1.2 cho ô-tô-mát hữu hạn trạng thái của trang *Organisation Details - Tab Information*. Bảng 5.10 bao gồm

33 transitions của ô-tô-mát hữu hạn trạng thái trang *Organisation Details - Tab Information* sau khi đã ghép nối.

Bảng 5.10. Các transition của trang *Organisation Details - Tab Information*

#	Transition		
1	S_index----add_User---->UserName	17	Add1----click_Save---->SaveSucess
2	S_index----add_Password---->Password	18	Nation----unselect_Nation---->OrgDetails
3	UserName----del_User---->S_index	19	Nation----edit_Add1---->Add1+Nation
4	UserName----add_Password---->User+Pass	20	Nation----check_Pre---->Nation+Preffered
5	Password----del_Password---->S_index	21	Nation----click_Save---->SaveSucess
6	Password----add_User---->User+Pass	22	Preffered----uncheck_Pre---->OrgDetails
7	User+Pass----del_Password---->UserName	23	Preffered----sel_Nation---->Nation+Preffered
8	User+Pass----del_User---->Password	24	Preffered----click_Save---->SaveSucess
9	User+Pass----login---->MenuList	25	Add1+Nation----unselect_Nation---->Add1
10	MenuList----select_MenuOrg---->OrgList	26	Add1+Nation----check_Pre---->Add1+Nation+Pre
11	OrgList----click_Org---->OrgDetails	27	Add1+Nation----click_Save---->SaveSucess
12	OrgDetails----edit_Add1---->Add1	28	Nation+Preffered----uncheck_Pre---->Nation
13	OrgDetails----sel_Nation---->Nation	29	Nation+Preffered----unselect_Nation---->Preffered
14	OrgDetails----check_Pre---->Preffered	30	Nation+Preffered----edit_Add1---->Add1+Nation+Pre
15	OrgDetails----click_Save---->SaveSucess	31	Nation+Preffered----click_Save---->SaveSucess
16	Add1----sel_Nation---->Add1+Nation	32	Add1+Nation+Pre----uncheck_Pre---->Add1+Nation
		33	Add1+Nation+Pre----click_Save---->SaveSucess

Với dữ liệu đầu vào là 15 trạng thái và 33 *transition* như bảng 5.10, chúng ta áp dụng thuật toán 4.1 và 4.2 sẽ được kết quả là 11 đường dẫn kiểm thử như bảng 5.11. Cột bên trái của bảng là số thứ tự các đường dẫn và cột bên phải là chi tiết từng đường dẫn.

Bảng 5.11. Các test path (đường dẫn kiểm thử) được sinh ra từ mô hình trang *Organisation Details - Tab Information*

#	Test path
1	Test path 1: S_index*add_User=UserName*del_User=S_index*add_Password=Password*del_Password=S_index*add_User=UserName*del_User=S_index*add_Password=Password*del_Password=S_index
2	Test path 2: S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*del_User=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*del_User=Password
3	Test path 3: S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*edit_Add1=Add1*sel_Nation=Add1+Nation*unselect_Nation=Add1*click_Save=SaveSuccess
4	Test path 4: S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*edit_Add1=Add1*sel_Nation=Add1+Nation*check_Pre=Add1+Nation+Pre*click_Save=SaveSuccess
5	Test path 5: S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*edit_Add1=Add1*sel_Nation=Add1+Nation*check_Pre=Add1+Nation+Pre*uncheck_Pre=Add1+Nation*click_Save=SaveSuccess
6	Test path 6: S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*sel_Nation=Nation*unselect_Nation=OrgDetails*check_Pre=Preferred*uncheck_Pre=OrgDetails*click_Save=SaveSuccess

7	<p>Test path 7:</p> <p>S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*sel_Nation=Nation*unselect_Nation=OrgDetails*check_Pre=Preffered*sel_Nation=Nation+Preffered*uncheck_Pre=Nation*click_Save=SaveSucess</p>
8	<p>Test path 8:</p> <p>S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*sel_Nation=Nation*unselect_Nation=OrgDetails*check_Pre=Preffered*sel_Nation=Nation+Preffered*uncheck_Pre=Nation*check_Pre=Nation+Preffered*click_Save=SaveSucess</p>
9	<p>Test path 9:</p> <p>S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*sel_Nation=Nation*unselect_Nation=OrgDetails*check_Pre=Preffered*sel_Nation=Nation+Preffered*uncheck_Pre=Nation*check_Pre=Nation+Preffered*unselect_Nation=Preffered*click_Save=SaveSucess</p>
10	<p>Test path 10:</p> <p>S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*sel_Nation=Nation*unselect_Nation=OrgDetails*check_Pre=Preffered*sel_Nation=Nation+Preffered*uncheck_Pre=Nation*check_Pre=Nation+Preffered*edit_Add1=Add1+Nation+Pre*click_Save=SaveSucess</p>
11	<p>Test path 11:</p> <p>S_index*add_User=UserName*del_User=S_index*add_Password=Password*add_User=User+Pass*del_Password=UserName*add_Password=User+Pass*login=MenuList*select_MenuOrg=OrgList*click_Org=OrgDetails*sel_Nation=Nation*unselect_Nation=OrgDetails*check_Pre=Preffered*sel_Nation=Nation+Preffered*uncheck_Pre=Nation*edit_Add1=Add1+Nation*unselect_Nation=Add1*sel_Nation=Add1+Nation*check_Pre=Add1+Nation+Pre*uncheck_Pre=Add1+Nation*click_Save=SaveSucess</p>

	A	B	C
1	Test path 1: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*edit_OrgName=OrgName*click_Save=SaveSuccess	FAIL	FAIL EVENT:Event couldn't do.
2	Test path 2: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*select_Nation=NationCountry*click_Save=SaveSuccess	PASS	
3	Test path 3: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*check_Preferred=PreferredOrg*click_Save=SaveSuccess	PASS	
4	Test path 4: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*select_Nation=NationCountry*check_Preferred=Nation+Preferred*click_Save=SaveSuccess	PASS	
5	Test path 5: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*edit_OrgName=OrgName*select_Nation=Name+Nation*click_Save=SaveSuccess	PASS	
6	Test path 6: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*check_Preferred=PreferredOrg*select_Nation=Nation+Preferred*edit_OrgName=Name+Nation+Preferred	PASS	
7	Test path 7: S_index*add_UserName=UserName*add_Password=User+Pass*login=MenuList*click_OrgMenuList=OrgList*select_Org=OrgDetails*select_Nation=NationCountry*check_Preferred=Nation+Preferred*edit_OrgName=Name+Nation+Preferred	PASS	

Hình 5.31. Kết quả thực hiện đường dẫn kiểm thử hiển thị trong tệp tin đầu ra

Kết quả sau khi duyệt đồ thị đã thỏa mãn các điều kiện sau:

- Đảm bảo tất cả các cạnh đều đã được duyệt qua
- Mọi test path đều bắt đầu bằng trạng thái khởi đầu của đồ thị.
- Trạng thái cuối cùng của mỗi test path là trạng thái nằm trong tập trạng thái kết thúc của đồ thị.

Như vậy, chúng ta đã sinh ra tất cả các đường dẫn kiểm thử từ mô hình ô-tô-mát hữu hạn trạng thái của *Organisation - Tab Information*

Với các đường dẫn kiểm thử được sinh ra, công cụ sử dụng Selenium WebDriver để kết nối với các trình duyệt Web và tiến hành chạy các đường dẫn kiểm thử đó. Quá trình thực hiện kiểm thử được thực hiện thông qua các đường dẫn kiểm thử. Các đường dẫn kiểm thử này được coi như các kịch bản đầu vào cho công cụ Selenium.

Với kịch bản đã có, công cụ Selenium sử dụng các API như đã trình bày ở phần 5.1.1 để tiến hành kết nối đến trình duyệt.

Sau khi thực hiện kiểm thử ứng dụng Web bằng công cụ kiểm thử tự động tương tác giao diện Web, kết quả thực hiện sẽ được xuất ra một tệp tin Excel. Nội dung tệp tin cho người dùng biết chi tiết về các ca kiểm thử bao gồm: các đường dẫn kiểm thử, kết quả của đường dẫn kiểm thử đó và nếu đường dẫn đó không thực hiện thành công thì chỉ ra nguyên nhân. Hình 5.31 là một số kết quả sau khi thực hiện các đường dẫn kiểm thử trong tệp tin đầu ra.

5.2.6 Kết quả áp dụng và cải tiến công cụ

Dựa trên đề xuất từ [3] và việc phát triển công cụ từ [2], luận văn đã áp dụng thành công vào Website của công ty FPT Software. Ngoài việc áp dụng thành công, luận văn còn thực hiện phát triển tự động phần đầu vào cho công cụ. Dưới đây là chi tiết phần kết quả áp dụng và cải tiến công cụ

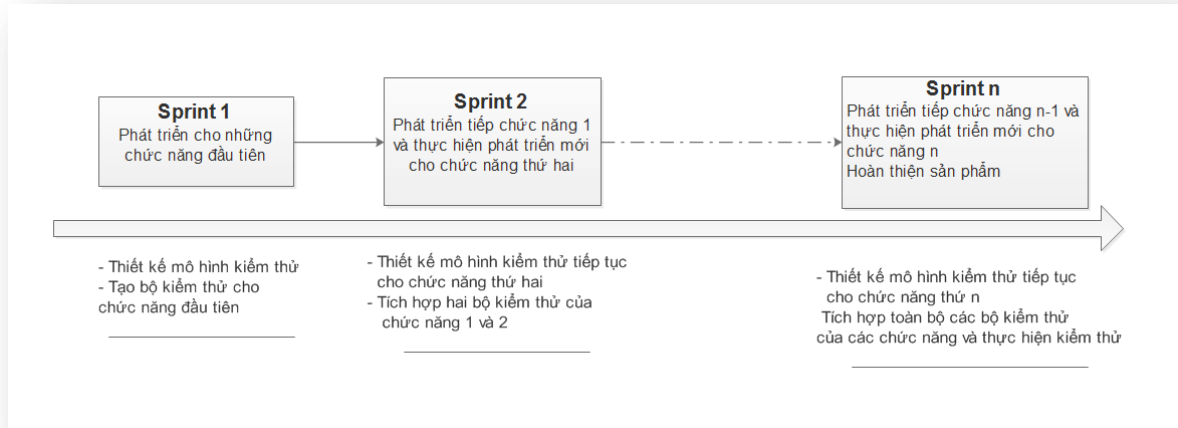
5.2.6.1 Kết quả áp dụng

Phương pháp kiểm thử tự động tương tác giao diện người dùng, cụ thể là phương pháp sinh bộ kiểm thử tự động dựa trên mô hình đã được áp dụng thành công vào trang Web FPT Service của Công ty phần mềm FPT đặc biệt là giai đoạn kiểm thử chấp nhận. Trước đây, Website FPT Service được thực hiện với mô hình truyền thống, sau khi giai đoạn kiểm thử mức hệ thống hoàn tất, toàn bộ trang Web được gửi cho phía đơn vị kiểm thử chấp nhận.

Có hai hình thức kiểm thử chấp nhận là, kiểm thử alpha và kiểm thử beta. Kiểm thử alpha thường được thực hiện bởi đội phát triển là chính và ở ngay đơn vị phát triển. Kiểm thử beta sẽ do người sử dụng thật dùng thử và đánh giá tại máy của người sử dụng và nơi làm việc, môi trường thật [1].

Tuy nhiên, theo xu hướng phát triển hiện nay, hầu hết các đơn vị tại FPT Software đã thực hiện áp dụng phát triển theo mô hình Agile, một mô hình phát triển nhanh. Không giống như các dự án truyền thống, kiểm thử chấp nhận trong các dự án Agile rất khác biệt so với các dự án truyền thống, nơi kiểm thử chấp nhận xảy ra ở phần cuối của vòng đời phần mềm. Trong dự án Agile kiểm thử chấp nhận được thực hiện trước khi phần mềm được chuyển giao. Theo [13], kiểm thử chấp nhận cũng có xu hướng được tự động hóa để họ có thể chạy như là kiểm thử hồi quy (regression test). Kiểm thử tự động rất quan trọng đối với mọi dự án Agile. Các gói sản phẩm phát triển thường xuyên yêu cầu các chu kỳ phản hồi ngắn, do đó kiểm thử hồi quy phải nhanh chóng và chính xác.

Luận văn áp dụng phương pháp sinh bộ kiểm thử và tạo ra bộ kiểm thử áp dụng kiểm thử chấp nhận cho mô hình Agile. Việc xây dựng bộ kiểm thử sẽ thực hiện dần qua từng giai đoạn phát triển (sprint) của mô hình. (hình 5.32)



Hình 5.32. Thực hiện kiểm thử qua từng giai đoạn theo mô hình Agile

Mỗi một sprint, kiểm thử viên sẽ tạo các mô hình và bộ đầu vào. Đến giai đoạn cuối hoặc các sprint cuối, các bộ kiểm thử sẽ được tích hợp và kiểm thử. (Hình 5.35). Việc phát triển dần như vậy sẽ đỡ tốn công sức và thực hiện hiệu quả trong công việc kiểm thử hồi quy. Kết quả thực nghiệm đã được trình bày tại mục 5.2.4

5.2.6.2 Cải tiến công cụ

Luận văn ngoài việc áp dụng thành công cho trang Web FPT Service còn cải tiến công cụ để thực hiện tiến dần tới tự động hóa một cách hoàn toàn cho công cụ kiểm thử tự động do [2] đề xuất.

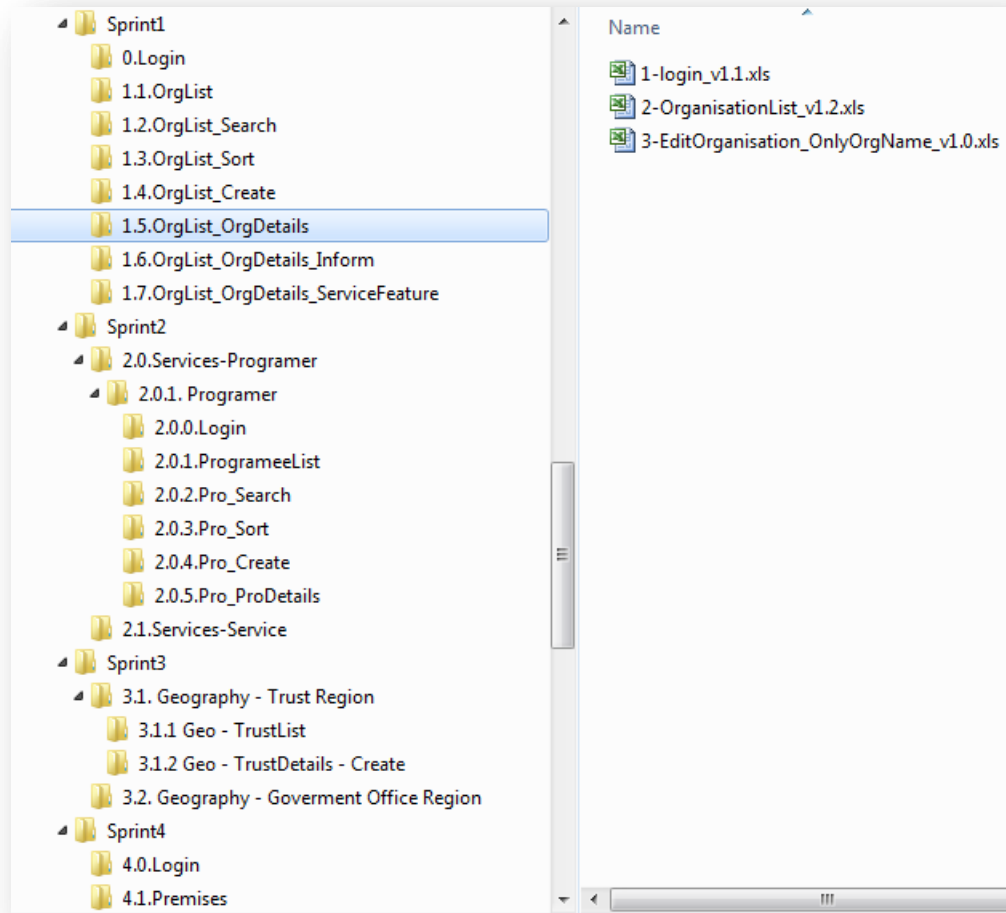
Cải tiến công cụ JFLAP tạo tệp tin đầu vào.

Công cụ JFLAP được trình bày tại mục 5.2.2, công cụ JFLAP trước đây mới chỉ dừng lại cho việc thực hiện sinh bảng transition trong bốn bảng của đầu vào. Các phân tử Web, các trạng thái được trích dẫn từ mô hình dựa trên cách đặc tả đầu vào cho mô hình. Tại mỗi trạng thái hoặc sự kiện, mỗi phân tử cần lấy được phân tách nhau bởi dấu “[”]. (Định nghĩa trong 5.2.2.1 mục b)

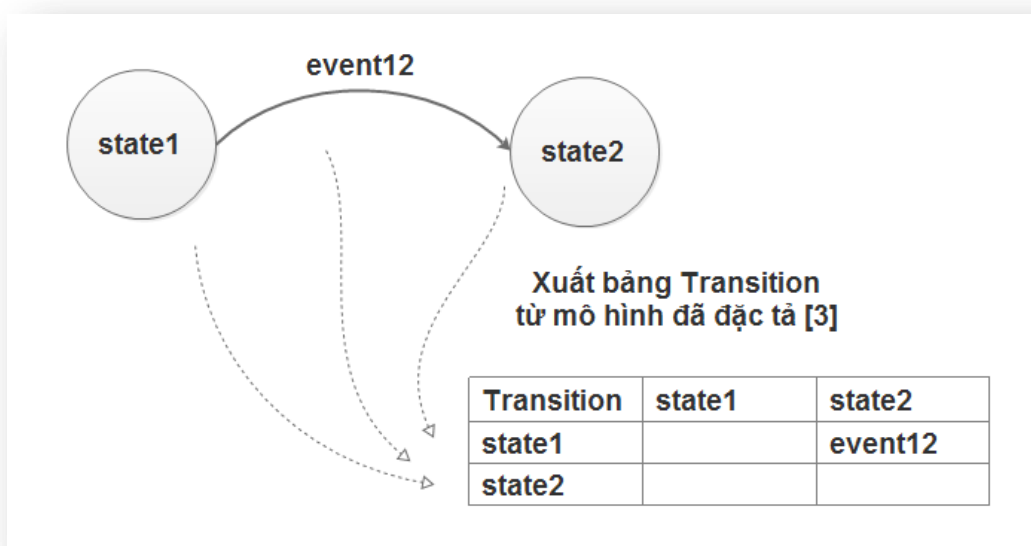
Ví dụ:

Tên_trạng_thái|Html_id_trạngthái|Kiểu_Htmlid

Tên_sựkiện|Html_id_sựkiện|Sựkiệntác động_htmlid

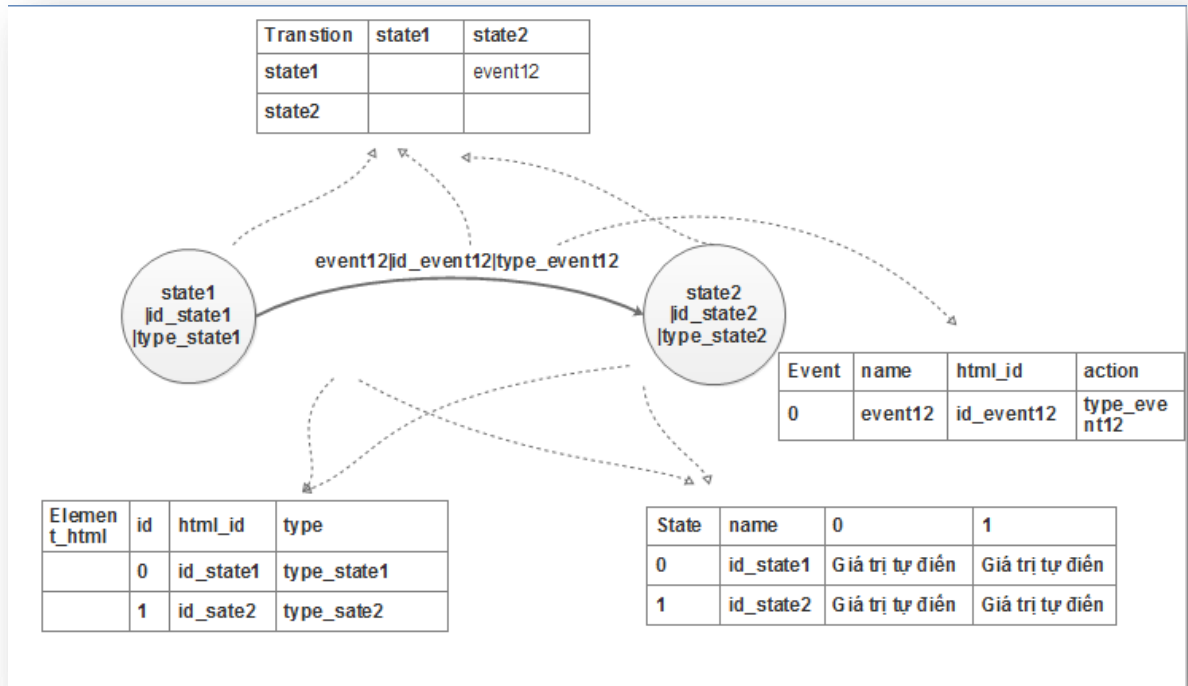


Hình 5.33. Tạo bộ kiểm thử qua từng sprint



Hình 5.34. Sinh đầu vào từ mô hình theo [3]

Như hình 5.34 mô tả việc sinh bộ đầu vào từ công cụ JFLAP, công cụ mới chỉ thực hiện sinh được một bảng duy nhất Transition. Tại hình 5.35, luận văn đã cải tiến công cụ bằng việc đặc tả kiểu mới nên sinh được bộ đầu vào đã bao gồm tất cả bốn bảng.



Hình 5.35. Cải tiến sinh đầu vào từ mô hình từ đề xuất của [3]

5.2.7 Ý nghĩa của công cụ thực nghiệm

Công cụ kiểm thử tự động ứng dụng Web của chúng tôi là giải pháp cho việc kiểm thử tự động tương tác giao diện cho các ứng dụng Web. Thực nghiệm cho thấy hướng phát triển tiềm năng đặc biệt trong xu thế phát triển mô hình Agile ngày càng mạnh. Cách thiết kế mô hình, kiểm thử tự động nhanh chóng cho giai đoạn kiểm thử chấp nhận của mô hình Agile mang ý nghĩa lớn. Phương pháp và công cụ vẫn còn những hạn chế nhưng vẫn có ý nghĩa quan trọng trong việc kiểm thử tự động tương tác giao diện các ứng dụng Web. Công cụ tự động sinh và thực thi các ca kiểm thử trên hầu hết các phần tử Web của các ứng dụng Web. Đây là giải pháp mang tính hiệu quả và có tính chính xác cao, giảm kinh phí và rút ngắn thời gian cho quá trình kiểm thử Web.

Ý nghĩa chính trong công cụ kiểm thử tự động tương tác giao diện người dùng chính là thay vì phải bỏ công sức rất lớn để viết một bộ kiểm thử theo chủ quan của người kiểm thử thì công cụ cho phép tạo từng bộ kiểm thử riêng lẻ và có khả năng thực hiện kiểm thử một cách ghép nối. Trong trường hợp kiểm thử hồi quy, hoặc kiểm thử chấp nhận, việc thay đổi hoặc cập nhật chức năng của trang Web là điều không thể

tránh khỏi, nhưng với công cụ kiểm thử tự động ATWA, việc thực thi kiểm thử đã giảm thiểu được công sức và chi phí thực hiện. Theo thực nghiệm, khi đo đạc với tần suất thực thi cho thấy với 11 đường dẫn kiểm thử, mỗi đường dẫn kiểm thử có khoảng 33 đường chuyển trạng thái, thời gian thực hiện mất ~ 3phút trong khi thực hiện thủ công với một đường dẫn kiểm thử mất hơn 1 phút, gấp 10 lần so với kiểm thử thủ công. Như vậy với kiểm thử công cụ, ngoài thời gian nhanh hơn kiểm thử thủ công, còn đảm bảo được tính chính xác và tính bao phủ của các ca kiểm thử, do đó thực hiện toàn bộ các ca kiểm thử cho toàn hệ thống sẽ rất nhanh chóng. Việc áp dụng công cụ để kiểm thử tự động thực sự mang lại rất nhiều lợi ích.

Ngoài các ý nghĩa của thực hiện áp dụng công cụ, việc phát triển tiện ích JFLAP cho phép người dùng tiến hành đặc tả các thiết kế một cách dễ dàng hơn. Với giao diện trực quan và dễ sử dụng, kiểm thử viên chỉ cần vẽ lại các thiết kế dưới dạng các máy hữu hạn trạng thái. Tiện ích sẽ xuất ra tệp tin excel đầu vào cho công cụ kiểm thử.

Với những lợi thế và ưu điểm đã nêu, trong tương lai công cụ có khả năng áp dụng hiệu quả cho các ứng dụng Web lớn trong thực tế. Hiện tại công cụ đã được triển khai thử nghiệm với một số ứng dụng Web tại công ty FPT Software và đã nhận được kết quả tích cực.

Chương 6: KẾT LUẬN

Kiểm thử tự động dường như trở thành một xu hướng tất yếu trong việc phát triển phần mềm nhanh, mô hình Agile hay Scrum là một ví dụ. Chúng ta không thể phủ nhận việc áp dụng kiểm thử tự động làm cho hiệu quả công việc cao hơn, giảm thiểu rủi ro trong sai sót, và giảm chi phí.

Kiểm thử tự động tương tác giao diện người dùng được xem như một giải pháp rất hữu hiệu khác với những kiểm thử tự động thường sử dụng là kiểm thử hiệu năng, kiểm thử bảo mật, kiểm thử chịu tải.v.v. Kiểm thử tương tác giao diện người dùng sẽ cho hiệu quả khác biệt khi thực hiện một cách tự động việc kiểm tra tính đúng đắn của chương trình so với các tài liệu thiết kế và các tài liệu đặc tả ban đầu. Không những thế, kiểm thử tự động tương tác giao diện người dùng còn giúp kiểm thử nhanh và chuẩn xác. Luận văn đã áp dụng và phát triển dựa trên đề xuất của [2] và [3]. Đề xuất của [2] và [3] đã đưa ra được các tính năng sau: kiểm thử luồng giao diện, xác định được phần tử định danh, xác định các phần tử có kiểu *name*, *class*, giao diện popup, kiểm thử được các loại phần tử Web như Dropdownlist, Checkboxlist, RadioList, v.v.

Tuy nhiên, dựa vào công cụ [3] và cải tiến của [2] thì việc kiểm thử tự động còn gặp nhiều khó khăn trong khâu thiết kế đầu vào, tạo các tệp tin excel, đặc biệt đối với những người lần đầu sử dụng sẽ rất khó và không tránh khỏi sai sót. Luận văn ngoài việc áp dụng thành công cải tiến của [3] và [2] đã cải tiến việc tạo tệp tin đầu vào bằng việc phát triển công cụ JFLAP. Đối với [2] tệp tin đầu vào được tạo bằng cách thủ công. Đối với [3], công cụ JFLAP đã được đưa vào sử dụng, nhưng mới chỉ dừng tại việc xuất ra một tệp tin excel chỉ có duy nhất một bảng, việc tạo thủ công cho phần tiếp theo cũng gặp hạn chế, dễ xảy ra sai sót.

Phương pháp kiểm thử tự động dựa vào hành vi tương tác giao diện ứng dụng Web đã trình bày có các ưu điểm nổi bật như: chỉ phụ thuộc vào phần hiển thị HTML mà không phụ thuộc vào ngôn ngữ lập trình Web, kiểm thử được cho hầu hết các loại phần tử Web và dễ dàng tạo các tệp tin đầu vào giúp giảm chi phí và công sức thực hiện. Với những ưu điểm trên, phương pháp hứa hẹn sớm được áp dụng rộng rãi trong thực tế, trở thành một công cụ hữu hiệu cho việc kiểm thử các ứng dụng Web hiện nay.

Về thực nghiệm, chúng tôi đã áp dụng công cụ kiểm thử tự động tương tác giao diện các ứng dụng Web cho một số hệ thống tại FPT Software. Dựa vào kết quả thực nghiệm, công cụ đã cho chúng ta thấy được tính khả dụng cũng như khả năng tạo những đường dẫn kiểm thử bao phủ được hầu hết các trường hợp có thể xảy ra trong hệ thống.

Trong tương lai, chúng tôi sẽ áp dụng công cụ cho các hệ thống phức tạp hơn nhằm chứng minh tính hiệu quả của phương pháp. Đồng thời, chúng tôi sẽ khắc phục và cải tiến công cụ để có thể kiểm thử tự động trên các trình duyệt có phiên bản cao hơn. Cải thiện công cụ JFLAP để tích hợp với công cụ ATWA với mục đích có thể đọc trực tiếp từ mô hình mà không cần phải thông qua tệp tin excel từ đó tiến tới một công cụ mang ý nghĩa tự động một cách hoàn toàn.

TÀI LIỆU THAM KHẢO

Tiếng Việt

- [1] Phạm Ngọc Hùng, Trương Anh Hoàng, Đặng Văn Hưng (2014), *Giáo trình kiểm thử phần mềm*, NXB Đại học Quốc gia Hà Nội.
- [2] Lê Thị Phụng (2015), *Nghiên cứu về kiểm thử dựa trên mô hình và ứng dụng*, Luận văn thạc sĩ, Trường Đại Học Công Nghệ, Đại học Quốc Gia Hà Nội.

Tiếng Anh

- [3] Khanh Trinh Le, Hieu Dinh Vo and Pham Ngoc Hung (2015), “A Method for Automated User Interaction Testing of Web Applications”, *Journal on Information and Communications Technology (JoICT)*, vol. E-3, no. 8(12), pp. 28-37
- [4] Mark Fewster , Dorothy Graham (2006), “Software Test Automation”, *The First Combined International Conference on Formal Approaches to Software Testing and Runtime Verification, FATES’06/RV’06*, pp. 115–132.
- [5] Ann Millikin (2014), *What Types of Software Testing Can and Should Be Automated*, <http://www.seguetech.com/>.
- [6] Ana Cristina Ramada Paiva Pimenta (2006), *Automated Specification-Based Testing of Graphical User Interface*, Thesis of Porto University, Portugal.
- [7] Selenium Document Team (2010), “Selenium Documetation version 1.0”, *Note to the reader on Website* <http://www.seleniumhq.org/docs>
- [8] JFLAP Tutorial, Guideline for user on Website <http://jflap.org/tutorial/>
- [9] Baiju Muthukadan (2016), “Selenium Python Bindings”, 2, Guideline from <http://selenium-python.readthedocs.org/>, pp.15-20
- [10] Selenium-WebDriver API Commands and Operations, Guideline for user on Website <http://www.seleniumhq.org>
- [11] Vineet Kumar (2015), *Action Class in Selenium*, Journal on Website <http://www.seleniumwebdriver.in>
- [12] Anneliese A. Andrews, Je Outt, Roger T. Alexander (2001), “Testing Web Application by Modeling with FSMs”
- [13] Report of David consulting group, (2013), *How can we make intergration/acceptance testing truly Agile*, Report from <http://www.softwarevalue.com/>