

สัปดาห์: วิธีเพิ่มประสิทธิภาพสำหรับเครื่องมือทดสอบซอฟต์แวร์แบบอัตโนมัติ



นาย นพกิตติ นวลชิต

# ศูนย์วิทยทรัพยากร จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

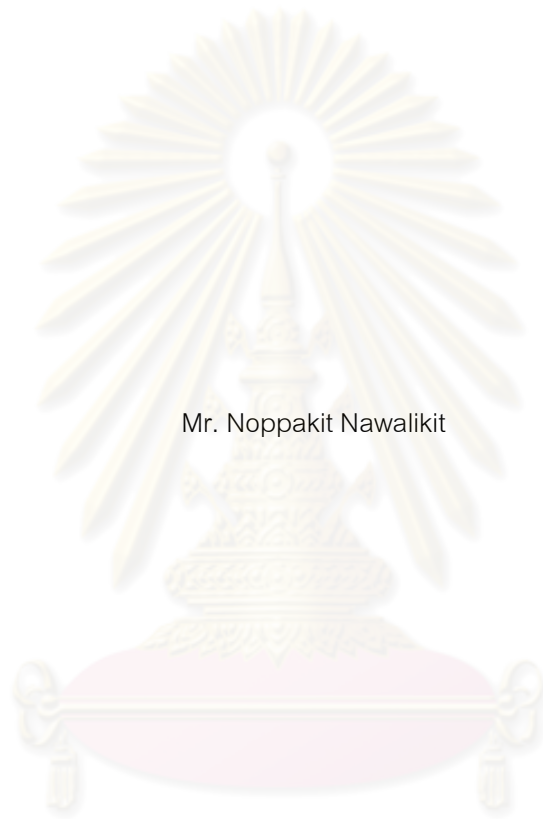
สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ ภาควิชาคณิตศาสตร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2552

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

SBTAR: AN ENHANCING METHOD FOR AUTOMATE SOFTWARE TEST TOOLS



Mr. Noppakit Nawalikit

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements  
for the Degree of Master of Science Program in Computer Science and Information

Department of Mathematics

Faculty of Science

Chulalongkorn University

Academic Year 2009

Copyright of Chulalongkorn University



นพทิศต์ นวลลิขิต : วิธีเพิ่มประสิทธิภาพสำหรับเครื่องมือทดสอบซอฟต์แวร์แบบอัตโนมัติ. (SBTAR: AN ENHANCING METHOD FOR AUTOMATE SOFTWARE TEST TOOLS) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ ดร. ภัทรสินี ภัทรโกศล, 77 หน้า.

การทดสอบซอฟต์แวร์เป็นกระบวนการสำคัญเพื่อปรับปรุงคุณภาพของซอฟต์แวร์ในการพัฒนาซอฟต์แวร์ ฉะนั้น เครื่องมืออัตโนมัติจำนวนมากถูกสร้างขึ้นเพื่อช่วยในการทดสอบการทำงานของซอฟต์แวร์ อย่างไรก็ตามเครื่องมือเหล่านี้ยังมีปัญหาเกี่ยวกับการใช้งานอยู่ อย่างหนึ่งก็คือรายงานการทดสอบซึ่งถูกสร้างขึ้นจากเครื่องมือโดยทั่วไปจะประกอบด้วยข้อมูลที่ไม่มีความหมาย ดังนั้นผู้ทำการทดสอบจึงไม่สามารถนำรายงานดังกล่าวนี้ไปใช้ได้อย่างมีประสิทธิภาพ ปัญหา นอกจากนี้ปัญหาที่เกิดจากความยืดหยุ่นของรายงานที่ถูกสร้างจากเครื่องมืออัตโนมัตินั้นจำเป็นต้องเปิดกับโปรแกรมที่เฉพาะเจาะจงเท่านั้น ดังนั้นเพื่อที่จะกำจัดข้อจำกัดเหล่านี้จากเครื่องมือทดสอบ วิทยานิพนธ์นี้เสนอวิธีใหม่ SBTAR เพื่อปรับปรุงการใช้งานของเครื่องมือทดสอบอัตโนมัติในส่วนของการสร้างรายงานผลลัพธ์จากการทดสอบที่ซึ่งจะนำเครื่องมือชื่อ IBM Rational Robot มาเพิ่มเติมคุณลักษณะบางอย่าง คุณสมบัติแรกจะช่วยให้ผู้ทำการทดสอบสามารถจัดการทดสอบวัตถุทั้งหมดที่วัตถุเหล่านี้ถูกเก็บในฐานข้อมูลแบบรวมศูนย์ โดยสามารถเพิ่ม, ย้าย หรือลบคุณสมบัติของวัตถุ เงื่อนไขของการทดสอบและข้อความแสดงข้อผิดพลาดที่จะถูกบันทึกลงในรายงานผลลัพธ์ คุณลักษณะที่สองจะทำการสร้างรายงานที่ประกอบด้วยข้อมูลที่เป็นประโยชน์ นอกจากนี้รายงานผลลัพธ์นี้ยังเพิ่มความยืดหยุ่น โดยสามารถเปิดใช้งานได้กับ Microsoft Word หรือ WordPad เพื่อให้สามารถอ่าน ฉะนั้นข้อบกพร่องต่าง ๆ ของโปรแกรมที่ถูกทดสอบสามารถระบุได้ง่ายขึ้น

ภาควิชา คณิตศาสตร์.....  
สาขาวิชา วิทยาการคอมพิวเตอร์และสารสนเทศ  
ปีการศึกษา .....2552

ลายมือชื่อนิสิต Noppakit Nawakitt.....  
ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก.....  
P. Bhatarakosol

## 5073630823 : MAJOR COMPUTER SCIENCE AND INFORMATION

KEYWORDS : SOFTWARE AUTOMATION TESTING / AUTOMATED TEST TOOL /  
EFFECTIVE OF TESTING TOOL

NOPPAKIT NAWALIKIT : SBTAR: AN ENHANCING METHOD FOR  
AUTOMATE SOFTWARE TEST TOOLS. THESIS ADVISOR : ASSISTANT  
PROFESSOR PATTARASINEE BHATTARAKOSOL, Ph.D., 77 pp.

Software testing is an important process to improve the quality of software in software development. Thus, many automation tools are created to help testing functionality of software. However, some usability issues of these tools have to be considered. One is that result logs which are generated from tools generally contain useless information. Therefore, testers cannot effectively use these logs; another issue is that all logs are not flexible to use since they need to open with a specific application. In order to eliminate these limitations from the test tools, this thesis proposes a new method, SBTAR, to improve the usability of automated test tools in a part of the result logs. Based on a tool named as IBM Rational Robot, some additional features are proposed. The first feature helps testers to manage all test objects where these objects are stored in a centralized database by adding, moving or deleting the objects' properties, test conditions, and error messages that will be recorded into the result log. The second feature generates report that contains useful information. Moreover, this result log also increases flexibility by Microsoft Word or WordPad to make them readable. Thus, the defects of the tested programs can be identified easily.

Department : Mathematics

Student's Signature Noppakit Nawalikit

Field of Study : Computer Science and  
Information

Advisor's Signature P. Bhattarakosol

Academic Year : 2009



## Acknowledgements

I am deeply indebted to my thesis adviser, Assist. Prof. Dr. Pattarasinee Bhattarakosol, whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

We would like to give thanks for fully help of QA Team of IBM Solution Delivery Co.,Ltd., Thailand who always give help when I faced the problems and thanks for every advices that make me finally got a solution. Also a big gratitude to “RationalUsers” group that sharing the idea, information and example for implementation function with Rational Robot.

Especially, I wish to express my deep sense of gratitude to my parent for their teaching, encouragement and love that helps me to pass the difficult times and complete this thesis.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## Contents

	Page
Abstract (Thai).....	IV
Abstract (English).....	V
Acknowledgements.....	VI
Contents.....	VII
List of Tables.....	X
List of Figures.....	XI
Chapter	
1 Introduction.....	1
1.1 Problem Statement.....	1
1.2 Objectives.....	2
1.3 Scope of thesis and Constraint.....	3
1.4 Definition.....	3
1.5 Benefits.....	4
1.6 Outline of the thesis.....	6
2 Fundamental Knowledge and Literature Review.....	7
2.1 Software Testing.....	7
2.1.1 Testing Approaches.....	8
2.1.2 Testing Techniques.....	8
2.1.3 Method of Software Testing.....	9
2.2 IBM Rational Robot.....	10
2.2.1 The Capabilities of Rational Robot.....	11
2.2.2 Performing functional testing and performance testing.....	11
2.2.3 Adding feature to GUI scripts.....	13
2.2.4 Using Rational Robot with Rational TestManager.....	16
2.2.5 Performing a test by use Rational Robot.....	18

2.2.6	Advantage and disadvantage of using Rational Robot for doing automatedtesting.....	18
2.3	Literature review.....	19
3	RATIONAL ROBOT AND THE ENHANCEMENT METHOD.....	27
3.1	Proposed Method.....	27
3.1.1	SBTAR: The statement transformation capability.....	27
3.1.2	SBTAR and the improvement on result log.....	37
3.2	Applying the SBTAR method to a test.....	44
3.2.1	An example of applying the SBTAR method to a test.....	48
3.2.1.1	Testing Scenario on Website of Chulalongkorn University..	48
3.2.1.2	Generating a script from Rational Robot.....	51
3.2.1.3	ObjectMap file preparation.....	53
3.2.1.4	Apply the SBTAR method to a script.....	54
3.2.1.5	Result log generated from the SBTAR method.....	57
4	Experimental Results.....	61
4.1	Experimental sample.....	61
4.2	Criteria for Measurement.....	62
4.2.1	Usability.....	62
4.2.2	Efficiency.....	62
4.2.3	Reliability.....	63
4.2.4	Reusability.....	63
4.3	Evaluation process.....	63
4.4	Summary of the evaluation.....	64
4.4.1	Usability.....	64
4.4.2	Efficiency.....	65
4.4.3	Reliability.....	66
4.4.4	Reusability.....	67
4.4.5	Overall satisfaction.....	68
5	Discussion and Conclusions.....	69
5.1	Discussion.....	69



5.2 Conclusions.....	70
5.3 Future work.....	71
References.....	72
Appendix A.....	76
Vita.....	77



ศูนย์วิจัยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## List of Tables

Table	Page
3.1 Object information organization in ObjectMap files.....	32



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## List of Figures

Figure		Page
2.1	Startup page of IBM Rational Robot version 7.0.1.....	10
2.2	The architecture of Rational Robot and Rational TestManager.....	16
2.3	Test Log in Rational TestManager.....	17
2.4	Information of the object in Log Event windows.....	17
3.1	The Architecture of Rational Robot and StatementTransformation Function.....	28
3.2	Use case diagram of StatementTransformation function of SBTAR.....	29
3.3	Class diagram of StatementTransformation function of SBTAR.....	30
3.4	Collaboration diagram of StatementTransformation function of SBTAR...	31
3.5	Activity diagram represents StatementTransformation function of the SBTAR.....	33
3.6	Implemented StatementTransformation in the library file, .sbl.....	34
3.7	Declare function StatementTransformation in the header file.....	35
3.8	The architecture of Rational Robot and ScreenBasedTrackingARray function.....	37
3.9	Example of a result log from the SBTAR function.....	38
3.10	Use case diagram of ScreenBasedTrackingARray function of SBTAR....	39
3.11	Class diagram of ScreenBasedTrackingARray function of SBTAR.....	40
3.12	Activity diagram represents ScreenBasedTrackingARray function of the SBTAR.....	40
3.13	Implemented ScreenBasedTrackingARray in the library file, .sbl.....	42
3.14	Declare function ScreenBasedTrackingARray in the header file.....	42
3.15	Use case diagram of the SBTAR method.....	45
3.16	Class diagram of the SBTAR method.....	46
3.17	A collaboration diagram of SBTAR when integrated two functions.....	47
3.18	A concurrent usage of the SBTAR method.....	47
3.19	An index page of Chulalongkorn University website.....	48

### List of Figures (Continued)

Figure		Page
3.20	Captions of the test web pages.....	49
3.21	Links to other pages on the CUW.....	51
3.22	The script that generated from Rational Robot.....	52
3.23	Input screen of the datapool of Rational TestManager.....	53
3.24	Examples of specifying values in ObjectMap.xls.....	53
3.25	Including the header file name.....	54
3.26	Declaring the important variables.....	55
3.27	Applying the SBTAR method to a script.....	56
3.28	Folders for generated reports from SBTAR.....	57
3.29	Report files which generated from the SBTAR.....	58
3.30	Example of report for “Pass” result.....	59
3.31	Example of a report for “Fail” result.....	60
4.1	The contingency table of the usability scores of the SBTAR method.....	64
4.2	The average frequency of SBTAR in the part of Usability.....	64
4.3	The average frequency of SBTAR in the part of Efficiency.....	65
4.4	The average frequency of SBTAR in the part of Reliability.....	66
4.5	The average frequency of SBTAR in the part of Reusability.....	67
4.6	The average frequency of SBTAR in the part of Overall satisfaction.....	68

# CHAPTER 1

## INTRODUCTION

This chapter is firstly states the interested problem in Section 1.1, and then the objective is described in Section 1.2. In Section 1.3, the scope and constraint of this thesis will be discussed follow by definitions of technical terms in Section 1.4. Additionally, benefits of thesis are mentioned in Section 1.5, and this entire thesis is also outlined in Section 1.6.

### 1.1 Problem Statement

In order to obtain qualified software, the software testing process becomes an important part in the software development life cycle. The efficiency of software testing can measure from the number of defects which is found during the test. Additionally, the test also should perform in a short time with cost effective as possible. Thus, many methods were introduced in order to help the testing process. One alternative is the implementing of Software Testing Automation (STA) which is one kind of the software to perform the test instead of manual checking. Mark Fewster and Dorothy Graham claimed that automate testing can save effort higher to 80% comparing with the manual testing [1]. However, some automated test is time consuming and costly but it helps producing better quality software than the manual testing.

Currently, several tools are implemented to help testers to improve the quality of the software. In addition, it also reduces the development time and cost by enhancing the productivity of testers and entire development project teams. As a result of using automated tools, all essential procedures, such as testing and reporting the testing results, will be performed automatically without human interfering. Therefore, many software companies have implemented automate testing tools, such as Rational Robot from IBM, Winrunner from Hewlett-Packard (HP), to ensure functionalities of the verified software. These tools can run scripts which are locally developed or written by supported languages.

Unfortunately, the results that generated by automated tools cannot present details in an understandable format. Moreover, testers cannot use an original log immediately because the log does not contain only usable information, but also useless information out which testers must screen them, and manually generates new reports. Additionally, sometimes, tools will generate information using technical information or application codes which tools can capture from applications under the test when error occurs. Thus, it is very hard to understand and testers may need some more time to find the meaning of them.

Although the automated tools have limitations mentioned above, these tools allow testers to implement scripts for various test conditions, including enhancing the test functions of the software. The outputs from the scripts are the additional information which will be stored in the result log. However, the standard of the output information has never been existed according to different testers' styles. Thus, the scripts which created by difference testers may generate difference meaning results for the same error. So, testers cannot use the original default log to show sequences of the execution process and the data that creates faults. Another limitation is that the result data is stored in the application-based format, only its generator can open the file. Thus, it is not convenience if testers would like to use this log on other machines with different platforms. As a consequence, testers need to install many applications and licenses to access this log.

## 1.2 Objectives

According to the limitations of the existing software test tools described previously, this research has objectives to dissolve those limitations using IBM Rational Robot as a studied tool. The aims of this research are described below.

1. Improve the capability of software test tools by implementing statement transformation function. This function enforces every tester uses the same standard report format and test conditions.



2. Implement a new method, called Screen Based Tracking Array (SBTAR), to generate testing report in a readable form that can be opened by commercial software, such as Microsoft Word, and WordPad.

### 1.3 Scope of thesis and Constraint

Such the limitation of automated tools mentioned on above section, this thesis emphasized on improvement of usability of automated in part of result log. For resolving the limitation, 2 new custom functions have implemented based on automated testing tools, named Rational Robot from IBM. The first function is responsible for drawing the required information from the centralized repository which is used in the verification processes. Moreover, the information will be recorded into the result log when the test is finished. The second function is implemented to support testers in reducing their effort for script's preparation and generating new format of a result log. The result log contains useful information that faster and easier to understand than using the original result log which was generated from the tool. This result log also increases flexibility by Microsoft Word or WordPad to make them readable. In practice, combination of these 2 functions is possible and makes higher benefit than use each of them only at a time.

Since there are various companies that have implemented automated testing tools based on different technologies and languages. From this constraint, SBTAR method which has implemented on Rational Robot from IBM may not be able to run on other tools which use different technology and languages.

### 1.4 Definitions

**IBM:** IBM is the best-known American computer manufacturer, founded by Thomas J. Watson (born 1874-02-17), known as "Big Blue" after the color of its logo.

**IBM Rational Software:** IBM Rational software provides a software development platform that improves the speed, quality, and predictability of software projects.

**Rational Robot:** IBM Rational Robot is an application that used as software automates testing tools which have capability to record activities that user performs. Rational Robot can playback the recorded script and also provides an environment for editing scripts.

**Rational TestManager:** Other Rational tools from IBM which can be integrated with Rational Robot to increase the capability of automate testing in order to manage all testing activities – planning, designing, implementing, executing, and analysing.

**Script File:** This is a file type which will be automatically generated by Rational Robot when users finish recording scripts.

**Library File:** This is a file type which will not be automatically generated by Rational Robot. The library file can store sub-procedures and functions that any script files can be accessed by every script.

**Header files:** This is a file type that declares global properties which will be available for multiple scripts and library source files. These global properties are are custom procedures, constants, and variables.

**Data pool:** A component of Rational Robot and Rational TestManager that is used in a data driven Test.

## 1.5 Benefits

This thesis proposed a new method name “SBTAR” to support the testing process. This method is different from other testing methods of other testing tools since it focuses in storing testing results.

1. Using this method can make a test much convenient, flexible and more standardize rather than fast execution mechanisms with difficult in understanding of the result log. This is because the method provides the capability to work with Microsoft Excel. Microsoft Excel has many capabilities, such as calculate the data, drag and drop feature, and etc. These features of Microsoft Excel help testers in preparing the test data.

2. With a new format of the result log, the testers can derive and fix errors of the developed software correctly and easily. In addition, the new result log has organized the information into an easy to understand format that helps the testers save the interpretation time of the data when presenting to stakeholders.

3. Similar to usage of the tools, the users of this new proposed module can implement the test faster and perform the test which can governable. Thus, the objective of saving time and cost of the project using the automate test tools is completely achieved.

4. Improved automation test to have more standardized. Since the method uses the information that are stored in the centralized database then the testers can save the maintenance time only by changing the error statement at the database in one place and it will available for other testers.

5. SBTAR can be adapted to many kind of application not limit to only kind application under test such as Web application, Client-Server application and etc.

6. Tester who using this method no need to know the structure of this methods, in work they just would like to know only what values that they would like to set as parameters for this method then SBTAR will work for them.

## 1.6 Outline of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, some related works are drawn, including the characteristics of Rational Robot, an automated test tools from IBM. Chapter 3 states the focusing problems, following by the proposed method. Chapter 4 is shown the experimental result of the new method. Finally, the discussions and conclusions are presented in Chapter 5.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## CHAPTER 2

### FUNDAMENTAL KNOWLEDGE AND LITERATURE REVIEW

In this chapter, the fundamental knowledge and literature review for this thesis are described. Thus, principle of software testing will be explained in Section 2.1, in Section 2.2 is reviewing of software test automation. In addition, some related work is reviewed in Section 2.3.

#### 2.1 Software Testing

Software testing is a process to ensure that software can operate as expected and their reliability which is an important factor of software will be achieved. Thus, defects or failures will be disclosed and resolved as many as possible. Therefore, there are various methods to perform the test of the software; each method will be described in the following section.

Throughout topic of testing concepts, it will use several terms that are fundamentals:

- **Validation:** An action for ensuring that the application is working against the original requirements. An example is the comparison of the actual system response of an on-line transaction to what was originally expected, requested, and finally approved in the external design
- **Verification:** An action for checking the application under test to insure it performs as specified by a predecessor. The comparison of a module's code against the technical design specifications document is one example.
- **Process:** a series of actions performed to achieve a desired result that transforms a set of inputs (usually information) into useful outputs
- **Expectations:** a set of requirements or specifications to which an output result of a process must conform in order to be acceptable. One such example

is the performance specification that an online application must return a response in less than two seconds.

### 2.1.1 Testing Approaches

There are 2 approaches which are static testing and dynamic testing; both of them are effective if applied properly and can be used throughout the application lifecycle. Nowadays, many tools have implemented to support both approaches and intention of these tools is to make an improvement on software testing. The tool can use to help tester during automated testing process. Testing tools and techniques will share characteristics from both of these classes. The development and test teams are responsible for selection and use of the tools and techniques best suited for their project.

- Static Testing – Static testing is an examination of a work product's characteristics to an expected set of attributes, experiences, and standards. It is generally not detailed testing, but checks mainly for the sanity of codes, algorithms, or documents. It is primarily the syntax checking of the codes and/or manually reviewing the codes or documents to find errors.

- Dynamic Testing - Dynamic testing is a process of verification or validation by exercising (or operating) an application by observing its behavior to changing inputs or environments. Where a module is statically tested by looking at its code documentation, it is executed dynamically to test the behavior of its logic and its response to inputs.

### 2.1.2 Testing Techniques

There are 3 commons testing techniques which are Black Box testing, White Box testing, and Error Guessing.

- Black Box testing - In the Black Box or functional testing approach, the testers have an "outside" view of the system. Black Box testing is requirements and/or specifications-oriented and is used at all test levels. This test



investigates throughout the feature or functionality of the application under the test. To test the system, all possible input combinations are entered and the outputs are examined. Both valid and invalid inputs are used to test the system.

- White Box testing - In the White Box or structural testing approach, testers have an inside view of the system. Structural testing checking the application by investigating the layout of the code and components are exercised according to some predetermined criteria. White Box testing is the logic oriented concept. Testers are concerned with the execution of all possible paths of control flow through the program. The "White Box" approach is essentially a Unit test method which is sometimes used in the Integration test or in the Operability test and is almost always performed by technical staffs.

- Error guessing – Mainly of this technique depends on the experience of testers. It helps the tester to identify the errors in prior, to identify the test cases on the basis of guessed errors in a particular piece of software coding. Based on the past experience, a set of test data can be created to anticipate those errors that will most often occur. Using experience and knowledge of testers, using invalid data that can representing or produce the common mistakes to verify that the system will handle these types of errors.

### 2.1.3 Method of Software Testing

There are two basic methods in performing software testing

1. Manual Software Testing – is the process of manual testing software which is performed by a human navigates and checks through application screens. Manual testing can take the form of navigate user interfaces, information submission. This test attempt to find faults of the application or even try to hack the application. Manual software testing is a labor-intensive and slow process.

2. Automated Software Testing – is the process of create the test scripts which can be run the test automatically, repetitively and through a number of iterations. Automated Software testing can significantly reduce the effort required for adequate testing or increase the testing which can be done in a limited time. Tests can

be run in minutes that would take hours to run manually. In some situation the test may be needed to execute during the night. As a consequence, testers may create mistakes or miss some tests. So, in order to improve the efficiency of the night tests, applying the test automation that supports testers without human interference is the best solution. However, the ability to apply the tests to the automated test tool is depended on ability of testing staffs or the technical limitations of the tool. Thus, the right combination of tests, technologies, and people can lead to a very high level of test automation and all the inherent benefits.

## 2.2 IBM Rational Robot

IBM Rational Robot is an application that used as software automated testing tools which have capability to record activities that users perform as test cases, playback the recorded script, and also provides an environment for editing script. Rational Robot is used in testing Microsoft Windows client/server and Internet applications which running under Windows NT 4.0, Windows XP, Windows 2000 and Windows 98 environment. Moreover, Rational Robot has extension ability by integrating with Rational TestManager to manage and playback a Rational Robot script, including managing the test logs. However, Rational Robot can be used as a standalone product. The next topic will provided overview of all capabilities of Rational Robot.



Figure 2.1: Startup page of IBM Rational Robot version 7.0.1

### 2.2.1 The Capabilities of Rational Robot

The summarizations for capabilities of automated tools name Rational Robot are described as follow:

- Perform functional testing. Playing back scripts that navigate through the application and test the state of objects through verification points.
- Perform performance testing. Use Rational Robot and Rational TestManager together to record and playback sessions that help testers to determine the performance of a multi-client system within the user-defined standards under varying loads.
- Create and edit scripts using the SQABasic and VU scripting environments. The Robot editor provides color-coded commands with keywords for powerful integrated programming during script development. (VU scripting is used with sessions in performance testing.)
- Perform the test on applications developed with IDEs, such as Visual Studio.NET, Java, HTML, Visual Basic, Oracle Forms, Delphi, and PowerBuilder.
- Perform the test on objects even if they are not visible in the application's interfaces.
- Collect diagnostic information about an application during script playback. Testers can playback scripts under a diagnostic tool and see the results in the log.
- Integrate with other Rational tools, such as Rational Purify, Rational Quantify, and Rational PureCoverage.

### 2.2.2 Performing functional testing and performance testing

Rational Robot has capabilities to creating 2 kinds of scripts which are GUI scripts for functional testing, and VU scripts (or call sessions) for performance testing. The different kinds of scripts in Rational Robot use different languages or syntax. For GUI scripts, it uses "SQABasic" scripts (using MS-Basic language syntax) to capture the commands equivalent to each user's action. On the other hand, the VU scripts (using C language syntax) capture entire streams of conversations HTML, SQL, Tuxedo,

CORBA Inter-ORB IIOP, and raw Sockets Jolt protocol that sent over the network. These are compiled into dynamic-link library (.dll) files and linked into an .obj compiled from a .c source file which calls a .dll file.

- Performance testing with combinations of Rational Robot and Rational TestManager

To perform the performance testing, Rational TestManager is a tool that are executing with Rational Robot in order to conduct performance tests on client/server systems. A client/server system includes client applications accessing databases or application servers, and browsers accessing web servers. The conversations among clients and servers are record in scripts by Robot, while the Rational TestManager schedules and playbacks the scripts.

During playback, the Rational TestManager can emulate hundreds, even thousands, of users placing heavy loads and stress on application databases and web servers.

The benefits of using Rational Robot and Rational TestManager in the performance testing are as follow:

- Find out if the system-under-test performs adequately.
  - Monitor and analyze the response times that users actually experience under different usage scenarios.
  - Test the capacity, performance, and stability of the server under real-world user loads.
  - Discover the server's break point.
  - Use Rational TestManager to view the logs that are created when scripts and schedules were executed.
- Functional testing with Rational Robot

Rational Robot will record actions that testers perform to the application-under-test. These actions include keystrokes, mouse moves and clicks; these actions will be used by that Rational Robot for performing during playback the test. The recorded GUI script establishes the baseline of expected behavior for the application-

under-test. When new releases of the application become available, testers can playback the scripts to test the builds against the established baseline. For the GUI script, the script can be executed on Rational Robot. However, this script still can be organized and executed by Rational TestManager.

There are various features in performing the functional testing by Rational Robot and Rational TestManager as described below.

- Find out if the system-under-test performs adequately by verifying its correctness of functionalities.
- Perform the test as recorded in the script, including keystrokes, mouse moves and clicks.
- Test the functionality on each modules of the application-under-test.
- The object's properties and data would have appropriate test since Rational Robot uses Object Testing technology.
- Use TestManager to view the logs that are created when scripts and schedules were executed.

In the next topic, how to enhance the capabilities of functional testing by use Rational Robot was described.

### 2.2.3 Adding feature to GUI scripts

This section describes the basic information of Rational Robot in details of creating and editing library source files and SQABasic header files.

Rational Robot can also implement sub-procedures and functions, either directly to script files or in library source file, including in scripts. The custom procedures which developed in the library source files can be called from procedures in other files (scripts and other library source files). The library source files are useful for storing custom procedures that could be accessed by multiple scripts. If a custom

procedure needs to be accessed by just a single script, testers should consider adding the procedure to the script rather than to a library source file.

#### A. Library Source Files

There are 2 types of SQABasic library source files which are:

1. .rec (Script File) – This file type will automatically generate by Rational Robot when users finished recording scripts. The compiled script can be executed (called as “playing back a script”). The script file can add verification point during recording, subprocedures and functions also can be added in to the script file.

2. .sbl (Library File) – This file type will not be automatically generated by Rational Robot. Additionally, it does not support the verification point. The library file can store sub procedures and functions that any script files can be accessed by every script, to use sub-procedures and functions must use “CallScript” command.

#### B. Header Files (.sbh)

Header files use for declaring custom procedures, constants, and variables that would like to make them available to multiple scripts and library source files. The header file can be created and edited within Rational Robot itself. They can be accessed by all modules within the project.

#### C. Example of Script file, Library File and Header File

- Example of Script File (Master.rec)

```
'$Include "global.SBH"
```

```
Dim iResult As Integer
```

```
Dim sResult As String
```

```
Sub Main
```

```
Dim iCount As Integer
```

```
'Initially Recorded: 11/4/2009 16:46:27
```

```
End sub
```



- Example of Library File (Mastertlibrary.sbl)

```

Sub TestNumber(input1 as Integer, input2 as Integer)
Dim iText as String
    If input1 > input2 then
        iText = "Input1 greater than Input2"
    ElseIf input 1 < input2 then
        iText = "Input1 less than Input2"
    Else
        iText = "The number you typed equals"
    End if
MsgBox iText

End Sub

```

- Example of Header File (Masterheader.sbh)

```

Global Input1 as Integer
Global Const Input2 as Integer = 10
Declare Sub TestNumber BasicLib "MasterLibrary" (input1 as Integer, input2 as Integer)

```

When playing back a script, Rational Robot repeats the step from actions which performs while recording and automating the software testing process which generated in the .rec file. With this process automation, each new version of application can be tested faster and more thoroughly than the manually testing. Thus, the testing time decreases while both coverage and overall consistency increases.

Furthermore, the automate test can be enhanced by adding the custom procedure or function into the library file (.sbl). This method supports Rational Robot performs all tests dynamically. However, in the case that a customized procedure or functions which in library file (.sbl) is used, the declaration of that function must be stored in the header file (.sbh), then included the header file into the script file (.rec). When playing back finished, Rational TestManager will generate a result log; a result log

is a file that contains records of events that occur while a script is playing back. Logs will be generated and available in Rational TestManager.

#### 2.2.4 Using Rational Robot with Rational TestManager

Rational Robot can be integrated with Rational TestManager to increase the capability of automated testing in charge of managing all testing activities – planning, design, implementation, execution, and analysis. Figure 2.2 is the architecture of Rational Robot and Rational TestManager.

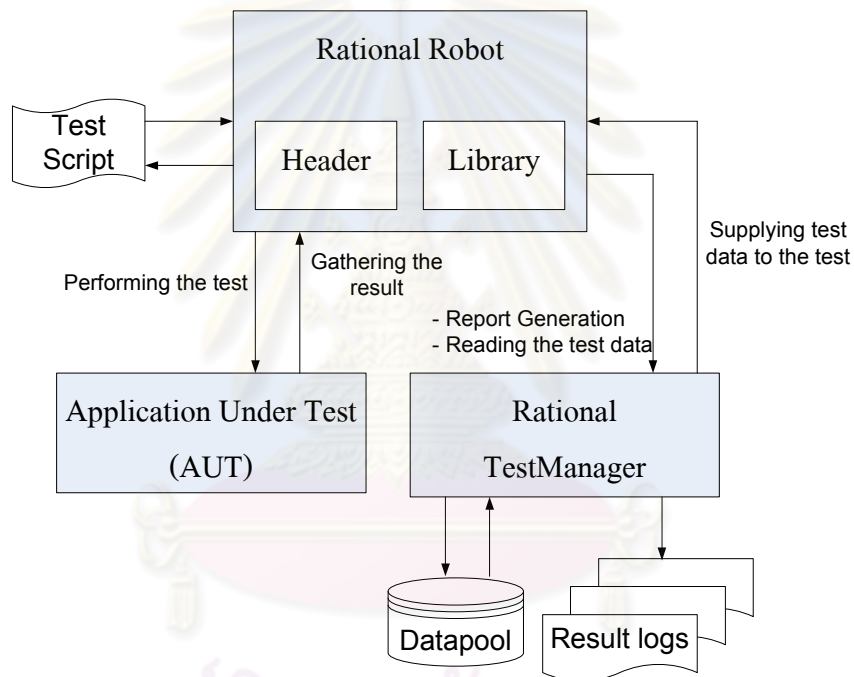


Figure 2.2: The architecture of Rational Robot and Rational TestManager

In the TestManager, testers can evaluate tests to determine the quality of the system-under-test or application-under-test by examining the results of a test in the test log. If the result from the test is “Fail”, the log will show information of “Result” column in red color. On the other hand, it will show in green color for “Pass”. Figure 2.3 is the picture from a test log that was generated by Rational Robot after complete running the test and was summarized and managed by the Rational TestManager.

The figure displays two screenshots of the Rational TestManager Test Log interface. The top screenshot shows a test run that failed, with a 'Script Command Failure' event highlighted. The bottom screenshot shows a test run that passed, with 'Script Start' and 'Script End' events highlighted.

Event Type	Result	Date & Time	Failure Reason	Computer Name	Defects
Computer Start	Fail	2/8/2552 18:27:48		NOPPAKIT-TP	
Script Start (Master)	Fail	2/8/2552 18:27:48		NOPPAKIT-TP	
Script Command Failure	Fail	2/8/2552 18:28:25		NOPPAKIT-TP	
Script End (Master)	Fail	2/8/2552 18:28:25		NOPPAKIT-TP	
Computer End	Fail	2/8/2552 18:28:25		NOPPAKIT-TP	

Event Type	Result	Date & Time	Failure Reason	Computer Name	Defects
Computer Start	Pass	26/4/2552 11:59:32		NOPPAKIT-TP	
Script Start (Master)	Pass	26/4/2552 11:59:32		NOPPAKIT-TP	
Script End (Master)	Pass	26/4/2552 11:59:43		NOPPAKIT-TP	
Computer End	Pass	26/4/2552 11:59:43		NOPPAKIT-TP	

Figure 2.3: Test Log in Rational TestManager

Furthermore, Rational TestManager also stores result and details of the property of the object in application-under-test that Rational Robot can capture and verified as it is the verification points. Tester can use information showing in Rational TestManager for investigate the problem or what is the problems on during test.

The figure shows a screenshot of the 'Log Event - Script Command Failure' dialog box. The dialog has two tabs: 'General' and 'Configuration'. The 'General' tab is selected, and it displays a table of properties and values for the failed event.

Property	Value
Event Type	Script Command Failure
Start Date/Time	2/8/2552 18:28:25
Stop Date/Time	2/8/2552 18:28:25
Result	Fail
Failure Reason	
Failure Description	Unable to find the object
Script Line Number	43
Script Name	Master
Additional Information	HTMLink Click, "HTMLText=English", ""
Additional Output	
Defects	

Figure 2.4: Information of the object in Log Event windows

### 2.2.5 Performing a test by use Rational Robot

Once tester orders Rational Robot to playing back a script, it repeats whatever tester performing during record and whatever they did the modification by adding any logic or enhancement module or function to script with application or system under test. With automation, testers can test each new release or newer version of application faster and more thoroughly than by manual testing.

There are 2 phases in the automated testing tool for playing back scripts which are listed below.

1. Test development phase – during this phase, testers playback scripts to verify their modules.
2. Regression testing phase – during this phase, testers playback scripts to compare the last version of applications under test to the baseline which are recorded into the scripts during the test development phase.

Before performing a regression testing phase, testers need to restore the environment including of states of Windows environment as well as applications under the test because of they can affect to script playback. The testers need to be sure that application-under-test is in the same state or condition when it was recorded the script. Additionally, testers also have to check for relevance components, such as network, database and other systems whether they are in the same state as when the script was recorded or not. If there are differences between the recorded environment and the playback environment, playback problems can occur.

### 2.2.6 Advantage and disadvantage of using Rational Robot for doing automated testing

In this section will notice through the advantages and disadvantages of Rational Robot briefly where the results that have been written are obtained from users who experienced with Rational Robot.

- Advantages of Rational Robot when performing the automated testing
  - Robot can complete the automated testing without human interference and is faster than the manual test.
  - Users can create custom procedures for specific tasks that can extend usability of Rational Robot.
  - Robot can integrate with other Rational tools, such as Rational TestManager, Rational ClearQuest that will increase efficiency of software testing process.
  - There is the community of developers who are interested in developing a new framework that can integrate to Rational Robot.
  - The basic information of Rational Robot, such as descriptions, and samples of Robot's commands have been provided in the help mode.
- Disadvantages of Rational Robot when performing the automated testing
  - The script which generated from Rational Robot is hard to understand because some specific keywords are used only by the Rational Robot.
  - In order to implement the script in Rational Robot, implementers must have knowledge in programming.
  - The changes of environments, such as removing the object from the screen, have a big impact to the developed scripts.
  - In some purpose, the original format of the result log needs to be translated before use.
  - Rational Robot can show its efficiency when integrated with other Rational tools, such as Rational TestManager; however, users have to pay in a high price for licenses of these tools.

### 2.3 Literature review

As the fact that testing is the key factor of success of software product, everyone must realize this fact. Before software is delivered, they must be tested even though the software is a simple one; the testing must be performed for every functions

and features. Additionally, when the software is altered, the latest version of software must be re-tested because some changes might affect to other un-altered functions or modules. Even though testing has been performed whenever software was modified or changed, there is no guarantee that errors and defects of software can be totally eliminated. Additionally, the current testing process is time consuming when application under test has complicated architecture, such as it consists of various modules. Furthermore, testers might not be able to test every module as need because of time limit.

In the testing process, testers have various techniques to apply for the test, such as manual testing, and automate testing and etc. The manual testing process requires testers to prove the correctness of applications by working step-by-step in front of the computer, many input combinations under different test cases will be entered and outputs are manually observed. As a consequence, the analyzing results may not be completed; moreover, the resource allocation while test may be inappropriate. Therefore, the manual testing can cause double cost of the project without qualified software delivered.

The most popular question in software testing is “Either automated test or manual test is the best solution for my testing phase?”. To answer this question, testers must understand the differences between the automated test and the manual test because both practices also have advantages and disadvantages. For example, the automated test operates faster than the manual test; however, the automated test takes time for implementing the test scripts. In addition, for the consideration should get the time compared to the complexity of application under test. These factors may help in decision whether the manual testing or the automated testing is appropriate to test the system.

As mentioned that the manual testing has defects causes by testers, the automated testing has been introduced to save time and cost. The principle of automated testing is to use a program, called as a tool, as a driver in the testing phase.



The test will be performed without human intervention. The tools will perform the test on application by feeding input, and validating against outputs that return from the application with the expected values. Moreover, the testing tools can execute a large volume of complicated test cases and testers can repeatedly re-run the test with cost free according to the implemented testing scripts from the original test phase. Additionally, automation test tools can carry out in-depth tests more than human because the automated testing tools have capabilities to capture and verify the properties of objects. Furthermore, these tools can execute on multiple computers with different configurations.

According to the benefit of using the automated test tools, many software companies have implemented the tools, such as Rational Robot of IBM, WinRunner of HP, to ensure functionalities of the verified software. Nevertheless, there is another category of the testing tools called as the “open-source”, or “non-copyright” software. This open-source software was developed by various developers and those developed source codes are aggregated over the Internet for any new developers. The new developers can download and alter the source codes by deleting, inserting, and editing the source code to obtain required software.

Another advantage of the open-source application is the knowledge exchangeable among communities of software developers; this leads to new ideas and innovation in software design. One of the open-source is Selenium which was initiated by Jason Huggins [10]. Selenium was used as a reusable testing framework for web applications. Testers execute Selenium directly from a web browser since the engine of Selenium was implemented by JavaScript and iFrames embedded in the browser. As same as other open-source, Selenium allows users to freely create the test in their own style as custom functions. In addition, Selenium also provides a set of standard commands and verification commands which help testers to develop scripts easily.

Even if automation test has many advantages as mentioned above, there are some disadvantages that have to be aware. Considering the testing processes of

the test tools, these processes have to be declared as scripts before the test starts. Thus, lacking proper knowledge in implementing testing scripts and resource management in the test process is able to fail the project after software delivery. Besides, testers must have the analytical skill to interpret the test results.

As mentioned above that the automated testing has disadvantages. Thus, many researchers indicate these defects in different aspects. One issue that has been considered by Rudolf and Klaus [2] is the factor for a large investment cost and a long testing time. This is because the automated testing consists of many components, for example test cases, automate scripts which are used by tools, include components must be implemented, developed and maintained along with the test. Moreover, whenever using the automated testing, every test case has to be transformed to the test scripts using by automated test tools and the testing components should be prepared before script recording, such as test database, application state, etc [3], [6]. Therefore, this paper indicates that before performing such test, the return of investment index must be evaluated. This includes the situations that the scripts have a very small amount re-used rate, or there is no inflexibility in modifying the scripts for other applications. [5], [7], [9]

The second defect of the automated testing is that testers cannot avoid implementing testing scripts. Thus, testers must have special knowledge and techniques to implement scripts since it affects to the quality of the testing scripts and the test [2], [3], [5]. [7]. However, keyword driven mechanism has been implemented to help testers on this issue. Tang and Cao [3] proposed a testing technique that the keyword driven is implemented into the scripts. Moreover, a new framework using an XML file to store all significant data, such as test cases, test scenarios, and test results, had been implemented with an XML parser to read the XML file. The outputs from the parser will be sent to the next testing layer. So, a new experience tester can perform the test easily.

Although the solution of Tang and Cao supports new testers, it is suitable only for applications that have stable requirements according to the development efforts and budgets. Based on the research of Lingzi Jin [4], investment cost and time of the automated testing is 5-10 times more than manual testing. One factor that increases the testing time of the automated testing is that the testing environment must be the same every testing time. Thus, the testing environment for a re-test must be reset to the same situation or same state of the previous test before the testing failure occurred. So, this research suggested that the automated scripts should be able to recover the testing state without human's interfering.

In order to obtain a qualified design and good quality of scripts, testers must have some knowledge and skill based on the test tools. Unfortunately, each test tool has different designs, and different standard scripts' formats. Moreover, the test tools may support only some applications and some technologies. Even though there are solutions to untie this limitation, they might cause more effort and time to customize the original package. Consequently, budget is expanded and delivery time is extended.

Since automated testing still have the general problems in the test implementation such as time, cost and implementer skills. Various researches in software testing related to the automated testing focusing in the testing tools have shown that the automated testing can reduce cost and times of the project, and it also increase the quality of the delivered applications. Some researchers have studied on the effectiveness of automated testing tools to develop a test. Most of these works refer to the replaying tools to perform regression testing on applications, or the way to efficient the tools by proposing the methods to increase reusability and flexibility so as to reduce the cost and time.

A new solution to use for the automated testing tools for executing the functional test is proposed by [9]. This solution has a test driver which uses for interpreting test cases. The testers set all test cases on the "Test Case Management Dashboard" into the class object and route them into a test engine which will then

execute automatically. Good point in Test Case Management Dashboard is that they have key-words which tools can recognize and use to perform the test. Thus, testers or relevance users who are not familiar with the programming can do automated testing or understand the testing steps in the Dashboard.

There are several methods or techniques have been developed to improve capability of automated testing tools to be better than the "Capture/Playback" tools. In test automation implementation, required programming skills is a basic problems. Keyword-based is one solution that Tom Wissink and Carlos Amaro [5] argued that is the best solution among software maintenance methods. The keyword-base will be developed by a set of actions including other keywords, input data and expected results will be recorded into the spreadsheet. After recording keywords to the spreadsheet, they will act as test executors to read and apply values which are written in the spreadsheet to perform a test. Using the keyword-base in the automated test can reduce the number of scripts that need to be created or maintained. The most important issue is reducing the requirement of programming skills.

In addition to the method keyword-driven, data driven, is another way to reduce the maintenance process and be able to perform coverage testing. Data driven is one of many testing methods that applied on software and help reducing the maintenance time and improve the testing coverage. The data driven approach uses the test scripts. These test scripts will test and verify operation based on the values of the data that stored in a centralized (shared) repository. Therefore, testers can save time in maintenance process enormously. Thus, whenever a new release of software with the changed GUI, testers only updates the repository and it will affect to every script. [4]

Considering structure of data in the centralized repository, the information of this repository is stored in separate files. However, these files are maintenance without the scripting and can be used as the data series that contain configure or sets of test data. In addition these data series, later, can add more

information or the configuration is changed to apply on another test that has the same logic and multiple scenarios.

The ATF Framework was proposed by Bin Mu, Mingkui Zhan and Lanfang Hu [11] in order to reduce the workload of testers in revising/creating a new script for a new function of a new release of an application. In the ATF Framework consists of 2 techniques which are data driven and keyword driven. As a result, these techniques enhance the performance and usability of automated testing tools. Consider the testing processes; data storages are needed during the processes of these techniques. Although there are various types of storages, XML is the chosen one for ATF framework. Consequently, this new framework has a high rate of reusability, expandability and robustness.

Although frameworks are implemented to save implementation and testing time of the testers, there is some unexpected problem related to the presentation language on the GUI of the software. It is the fact that all countries around the world have developed applications with GUI presentation in their official language (such as English language) and the national language (such as Korean language). These applications must be tested for quality control; testers must have knowledge for all GUI language presentation, otherwise, a foreign resource must be involved during the test which might be a time consuming and costly. Consequently, in the worst case this international version is delayed.

According to the above problem, Xin Guo et al. [12] purposed an approach for multilingual testing to be much easier, faster and higher quality and more cost effective. This approach works with multiple components where the knowledge base is the core component. The knowledge base is a collection of data that will use to compare test cases and change the language of the test cases from the official or national language to the target language. In additional, the approach also provides test verification capabilities which reduce the possibility of human errors.

In the real world of computing, there are a variety of platforms for application execution, such as Microsoft Windows, UNIX, Linux, AIX, and AS400 of IBM. However, testing is still a major point for ensuring the quality of software on every platform. Thus, Jie Hui et al. [8] proposed a new framework name “LKDT- Linux, keyword-driven distributed test automation framework. The result shows that the testing process can be improved and the workload for script development is reduced about 50%. Moreover, the automation testing framework also helps to reduce runtime and complexity of creating and maintaining scripts.

Sometimes the test cannot be performed effectively because of insufficient resource, time and tools. Weider D. Yu and Giri Patil [13] proposed a framework that applied workflow technique into the testing automation process in order to improve efficiency and productivity of the test team. With this method, a workflow system was implemented and it consists of 3 main parts: Workflow Manager, Workflow Agents, and, User Interface and Communication Protocols. The Workflow Manager is responsible to control and maintain results of every execution, intermediary communicate with Workflow Agents running on different machines by ordering the agents to work and send the execution information to the Workflow Manager.

Previous researchers have studied the effectiveness of automated testing tools to develop a test. Their works mostly mentioned about replaying tools to perform regression testing on applications, or the method to efficient the tools by proposing the mechanisms to increase reusability and flexibility so as to reduce the cost. However, the researchers did not mention about how to utilize abilities of the tools to gain the highest efficient result. For software automation test, using tools to capture and playback does not mean the whole story of “Test Automation” [1], but it is only a very small part of the entire testing processes, the ability to create result and its usability is the one which must be included. The new method called the “SBTAR” has been implemented in this thesis in order to eliminate the mention above problems. The method will be described in the next chapter.



## CHAPTER 3

### RATIONAL ROBOT AND THE ENHANCEMENT METHOD

This chapter describes IBM Rational Robot with the enhancement method name "SBTAR: Screen Based Tracking Array" and how this method help testers to perform the test faster and more standardize. Therefore, in Section 3.1 describes the proposed method that consists of 2 functions: statement transformation, and screen-based tracking array. Then, Section 3.2 the examples to demonstrate the success in the implementation of the proposed method.

#### 3.1 Proposed Method

In this research, a new method called "SBTAR" is introduced; the method was implemented use a capability of IBM Rational Robot to create the customized functions. There are 2 implemented functions. The first function of SBTAR is the statement transformation function, named as StatementTransformation function. This function has aims to reduce the limitation of the data pool mentioned above and increase the standardize of the error messages, provide flexibility for testers in preparing test cases and testing data. The second function relates to the screen presentation. Since every original tool manages errors by presenting on the screen and there is no screen capturing, testers must find the errors' positions manually. Therefore, the testing process for each state will be a time consuming for software development life cycle. Details of each function are described as follows.

##### 3.1.1 SBTAR: The statement transformation capability

SBTAR provides the error statement transformation capability. SBTAR will compare between baseline data and actual data. If there are differences, it will map the error information with general statements that are kept in a database (or an excel file) then write them to result log. Figure 3.1 presents the architecture of Rational Robot and StatementTransformation function.

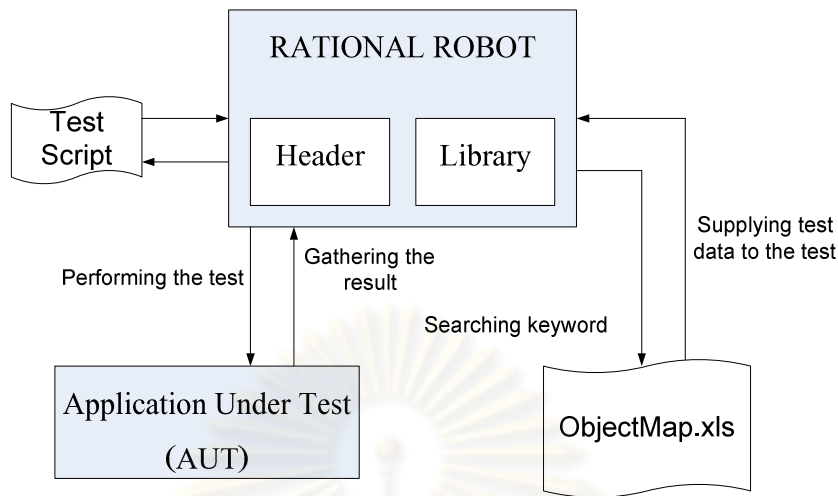


Figure 3.1: The Architecture of Rational Robot and StatementTransformation Function

Normally, Rational Robot has a data pool which is a source of test data variables that scripts can draw during playback. However, Rational Robot must sequentially access the input; it cannot randomly or directly access to the data row. Therefore, testers can choose Microsoft Excel to behave as a data pool for flexible data preparation. The benefit of using Microsoft Excel is that it can apply to other automated testing tools.

According to the benefits of Microsoft Excel mentioned above, SBTAR will integrate this software for its operations by searching and reading the information which store in the cell of Microsoft Excel store to array variables. These variables will be used in Rational Robot as input data for scripts. From the concept mentioned earlier, Figure 3.2 presents a use case of the error statement transformation capability of the SBTAR method.

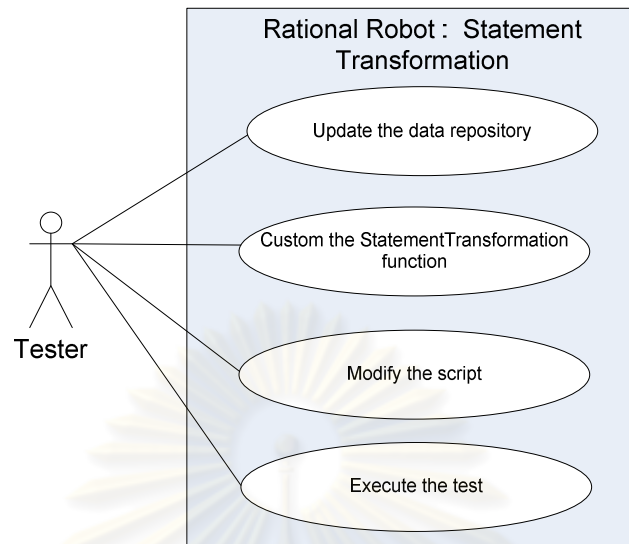


Figure 3.2: Use case diagram of StatementTransformation function of SBTAR

Figure 3.2 shows how the testers use the statement transformation feature of the SBTAR method. The diagram shows activities that testers have interact with the StatementTransformation function of the SBTAR method. In order to apply the StatementTransformation function to the test script, testers have to maintain the test data that stored in the repository. Implementation of the StatementTransformation function is the next step that testers have to implement to the library file of Rational Robot. Then, modifying the test script by putting the important key-words, parameters and procedures that required by the StatementTransformation function will be performed. The last is executing the script to start the automated testing. In the next section each of activity will be described. Moreover, the class diagram of the statement transformation feature of the SBTAR method shows in Figure 3.3.

จุฬาลงกรณ์มหาวิทยาลัย

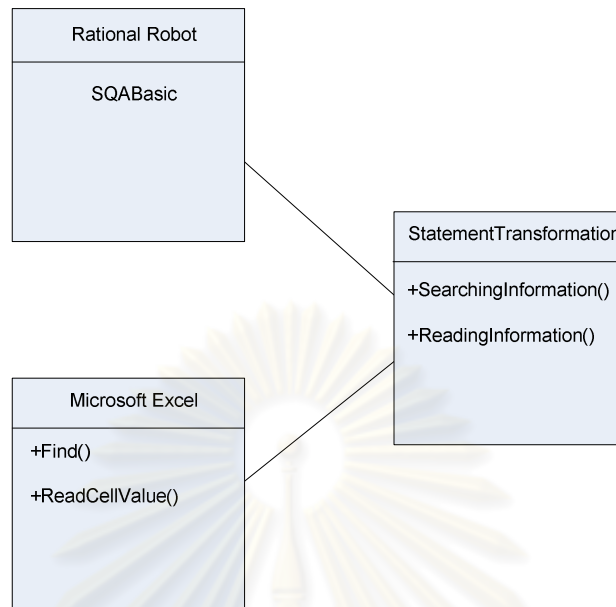


Figure 3.3: Class diagram of StatementTransformation function of SBTAR

From Figure 3.4, SBTAR executes the test on an application under a test object, including the related information, from the file name “ObjectMap.xls” and store in the array variables. SBTAR will use these variables during the test. To search in the file name “ObjectMap.xls”, SBTAR will use the required keyword and search through the spreadsheets. This practice has much efficiency than using the data pool of Rational Robot. Thus, the limitation of Rational Robot according to the data pool is eliminated. In addition, it also reduces the operation time because SBTAR uses capability of Microsoft Excel to search for the specific value.

ศูนย์วิจัยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

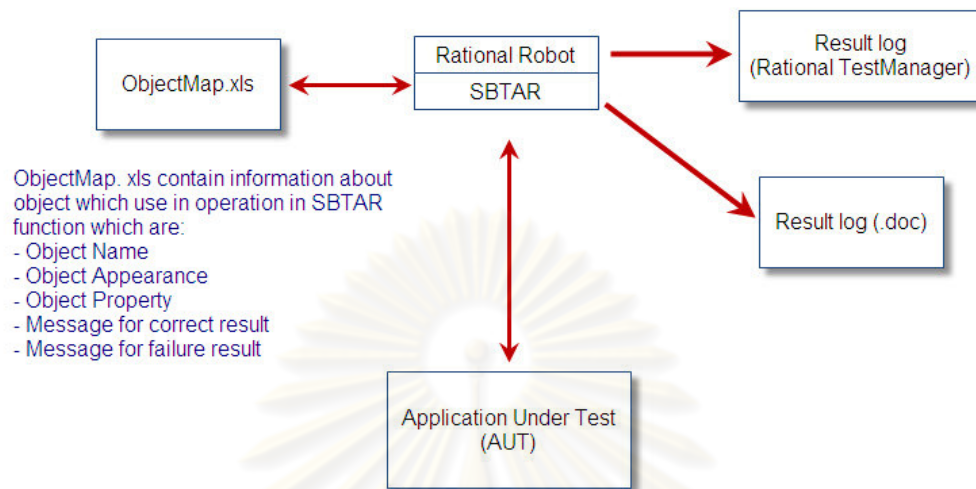


Figure 3.4: Collaboration diagram of StatementTransformation function of SBTAR

In “ObjectMap.xls”, it is containing information about objects which are needed for SBTAR functions. Information contains in the “ObjectMap.xls” are elaborated below:

- Object Name – This column is used for containing the name of objects that will be used for referring in scripts.
- Object Appearance – This column is used for containing property of objects which Rational Robot can use for recognizing the objects on the applications during the test.
- Verification Property – This column is used for containing the property of objects which SBTAR would checks by comparing between the baseline data and the actual data while test execution.
- Baseline Value – This column is used for containing the baseline values that would be used in the verification of SBTAR functions.
- Message for correct result – This column is used for containing a statement that will be written in a result log when the testing is “Pass”.
- Message for failure result – This column is used for containing a statement that will be written in a result log when the testing is “Fail”.

Table 3.1: Object information organization in ObjectMap files

Object Name	Object Appearance	Verification Property	Baseline value	Message for Pass verification	Message for Fail verification
HomeThai	Type=HTMLLink;HTMLText=หน้าหลัก	innerText	หน้าหลัก	Home page was load successfully	Cannot find link "หน้าหลัก" on Home page
StudentThai	Type=HTMLLink;HTMLText=นิสิตปัจจุบัน	innerText	นิสิตปัจจุบัน	Home page was load successfully	Cannot find link "นิสิตปัจจุบัน" on Home page
StaffThai	Type=HTMLLink;HTMLText=บุคลากร	innerText	บุคลากร	Home page was load successfully	Cannot find link "บุคลากร" on Home page
AlumniThai	Type=HTMLLink;HTMLText=ศิษย์เก่า	innerText	ศิษย์เก่า	Home page was load successfully	Cannot find link "ศิษย์เก่า" on Home page
CouncilThai	Type=HTMLLink;HTMLText=สภามหาวิทยาลัย	innerText	สภามหาวิทยาลัย	Home page was load successfully	Cannot find link "สภามหาวิทยาลัย" on Home page
ContactThai	Type=HTMLLink;HTMLText=ติดต่อเรา	innerText	ติดต่อเรา	Home page was load successfully	Cannot find link "ติดต่อเรา" on Home page
MapThai	Type=HTMLLink;HTMLText=แผนที่จวปไซค์	innerText	แผนที่จวปไซค์	Home page was load successfully	Cannot find link "แผนที่จวปไซค์" on Home page

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย



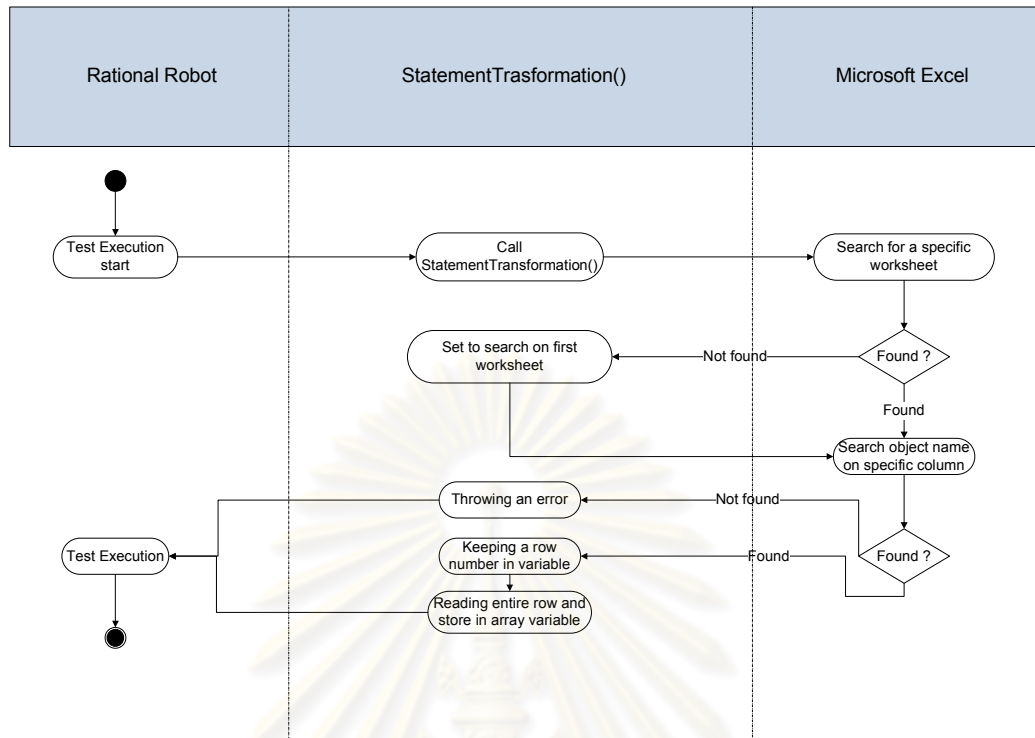


Figure 3.5: Activity diagram represents StatementTransformation function of the SBTAR

According to the activity diagram in Figure 3.5, the processes of StatementTransformation function can be described as follow.

(1) The StatementTransformation function will open an “ObjectMap.xls” and search for the specific worksheet. If SBTAR cannot find the worksheet, it will set the first worksheet to be used.

(2) The StatementTransformation function will search in the worksheet for the required keyword.

(3) If it cannot find the searched keyword, SBTAR will throw an error to the screen presentation.

(4) If SBTAR finds the keyword, it assigns the row's number that contains the keyword to the variable.

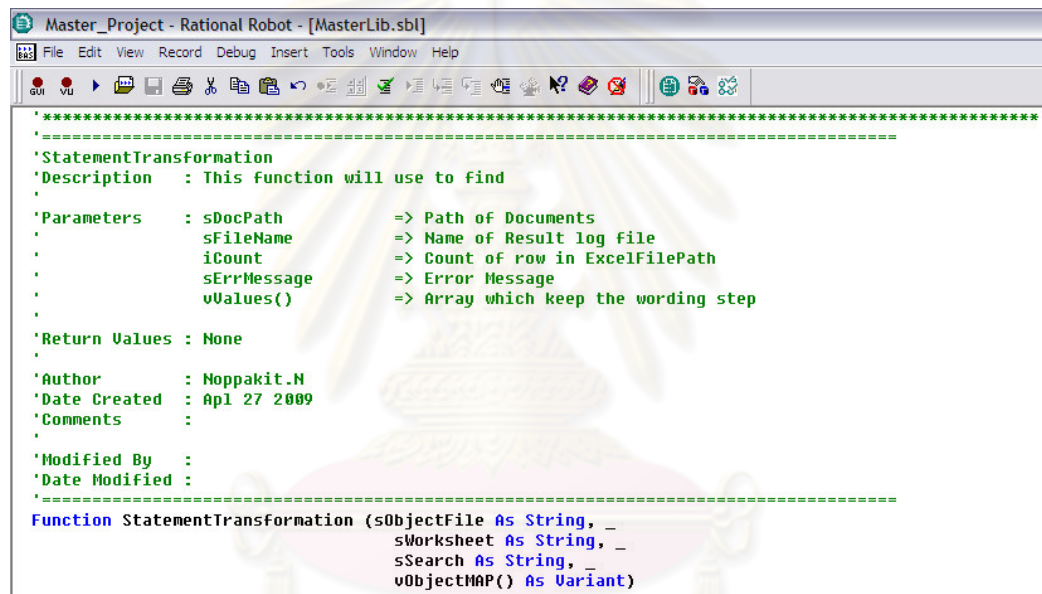
(5) SBTAR reads the information of every cell from the row and then assigns to array variable.

Whenever a function was developed in an individual script file, other scripts or library source files cannot call for supporting their processes. Therefore,

implementing functions such as StatementTransformation function, in a library source file will eliminate the limitation mentioned previously. The next section describes the implementation of the StatementTransformation function in the library source file.

### *A Custom Function*

The StatementTransformation function will be implemented as a function in the library file, .sbl, because this function must be available for every script when needed. The function that was created in the library file is shown in the example presented in Figure 3.6.



```

Master_Project - Rational Robot - [MasterLib.sbl]
File Edit View Record Debug Insert Tools Window Help
gui vu
*****
'StatmentTransformation
'Description : This function will use to find
'
'Parameters : sDocPath => Path of Documents
: sFileName => Name of Result log file
: iCount => Count of row in ExcelFilePath
: sErrMsg => Error Message
: vValues() => Array which keep the wording step
'
'Return Values : None
'
'Author : Noppakit.N
'Date Created : Apl 27 2009
'Comments :
'
'Modified By :
'Date Modified :
*****
Function StatementTransformation (sObjectFile As String, _
sWorksheet As String, _
sSearch As String, _
vObjectMAP() As Variant)

```

Figure 3.6: Implemented StatementTransformation in the library file, .sbl.

After completing the implementation of the custom function in the library file, the next step is to declare the function in the header file in order to provide access permission to other scripts or other library files. Figure 3.7 presents the declaration of the function in the header file.

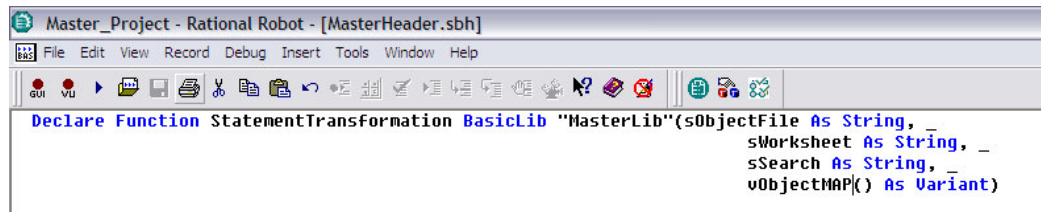


Figure 3.7: Declare function StatementTransformation in the header file

Once the function has been implemented in the library file and declared in the header file, the prepared test scripts have to be customized by putting the specific syntaxes and the important values. Thus, customization of the scripts would be described in the next section.

#### *Script Customization*

The script file must be customized by specifying the value to variables and adding them to the script. There are 4 parameters that are needed to be specified to be used in the StatementTransformation function.

1. "sObjectFile" is a parameter for keeping a value of the location of the file name "ObjectMap.xls".
2. "sWorkSheet" is a parameter for keeping a value of the worksheet's name.
3. "sSearch" is a parameter for storing the search value.
4. "vObjectMAP()" is a parameter for storing information obtained from the "ObjectMap.xls".

An example of a value storing in the "sObjectFile" is shown below.

```
sObjectFile = "C:\RobotNote\Master_Project\TestData\ObjectMap.xls"
```

An example of a value storing in the "sWorkSheet" is presented as follow.

```
sWorksheet = "Sheet1"
```

An example of the value storing in the "sSearch" is shown below.

```
sSearch = "HomeThai"
```

The last step in customizing the script is to put the 'Call' statement to gain access to functions in the library file. The following statement is an example of the call statement.

#### **Call StatementTransformation (sObjectFile, sSearch, sWorksheet, vObjectMAP())**

After calling the call statement above and every declaration of all variables, the StatementTransformation function will open the ObjectMap on the path "C:\RobotNote\Master\_Project\TestData\ObjectMap.xls" searching for the object name "HomeThai" in the worksheet name "Sheet1". The function will assign the information stored in each column of the searched object to the array variable name "vObjectMAP()".

After storing information from the file ObjectMap.xls to the array variable, testers can use this information using the array index. The "StatementTransformation" function will organize the value in the array variable as following:

vObjectMap(1) – The first variable keeps the value reading in column "A" which is containing the value of object's name.

vObjectMap(2) – The second variable keeps the value reading in column "B" which is containing the type and property of the object.

vObjectMap (3) – The third variable keeps the value in column "C" which contains the value of the verification property.

vObjectMap (4) – The forth variable contains the value of expected values or baseline value recorded in column "D".

vObjectMap (5) – The fifth variable reading from column "E" indicates the situation when the software past the verification process, using the message indicator.

vObjectMap (6) – The sixth variable, reading from column "F", contains the message when the verification fails the test.

Consider processes of the StatementTransformation function under SBTAR's architecture. These processes need flexibility in data access; thus, centralize control is the best solution of this requirement. Consequently, all testers can retrieve the required

data easily. The next section will describe the screen management that supports testers to locate errors.

### 3.1.2 SBTAR and the improvement on result log

As an objective of SBTAR is to alter the method in using the result log, a new mechanism will be added into the library of Rational Robot. As the result, a new report will be generated into a Microsoft Word format with the extension “.Doc”. This file type is a common document file that can be read by “Microsoft Word” and “WordPad”. Figure 3.8 presents the architecture of Robot and ScreenBasedTrackingARray function.

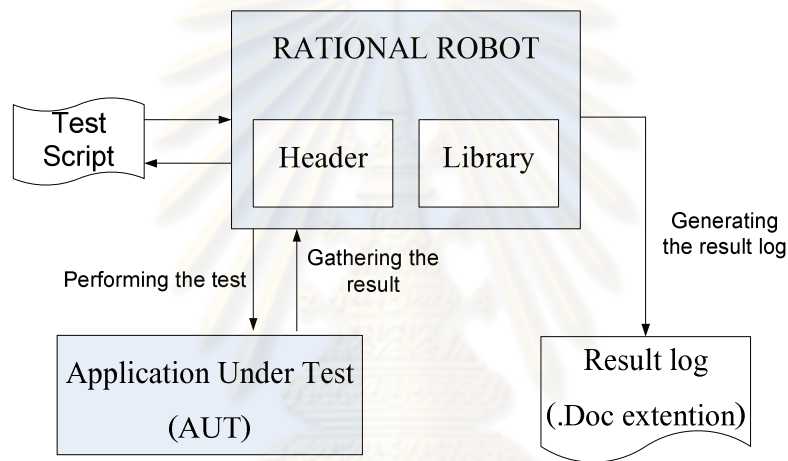


Figure 3.8: The architecture of Rational Robot and ScreenBasedTrackingARray function

Generally, the results which were generated by automated tools cannot be presented in an understandable format. Similarly, testers cannot use the original log immediately because the log contains varieties of information that might not be able to be used. Therefore, testers must perform the information filtering and manually generate new reports. Another problem is that the testing results are storing in the application-based format; so, only the specific application can open the file. Thus, it is not convenience for testers who would like to use this log on other machines with different platforms. As a consequence, testers need to install various specific license applications to gain access to the log.

The new method will use the capability of IBM Rational Robot to create a customized function; the function will generate a new format of a result log. The testers

can customize information and can be recorded into the result log. So, the useless information problems can be expelled and testers can use the result log immediately.

In the new result log, the information that will be included are:

- Name of script
- Name of person who run the script
- Date and time when the script is executed
- Information of each test step
- Test result (Pass or Fail) and description
- Captured screen

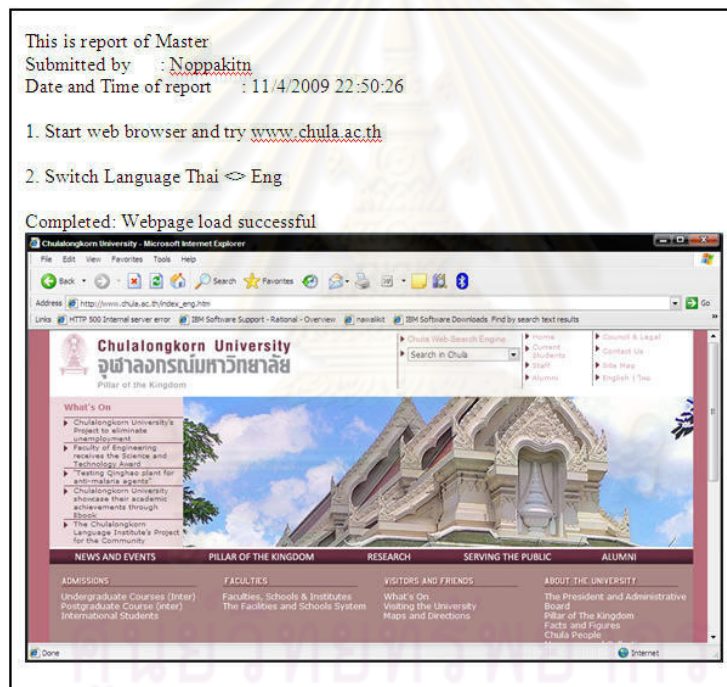


Figure 3.9: Example of a result log from the SBTAR function

Figure 3.9 presents an example of the result log that was created from the new function. All information in the result log can be distributed and easy for testers to understand and to perform a manual re-test process. Additionally, developers can use the information to drive for locating causes and problems of the entire software processes.



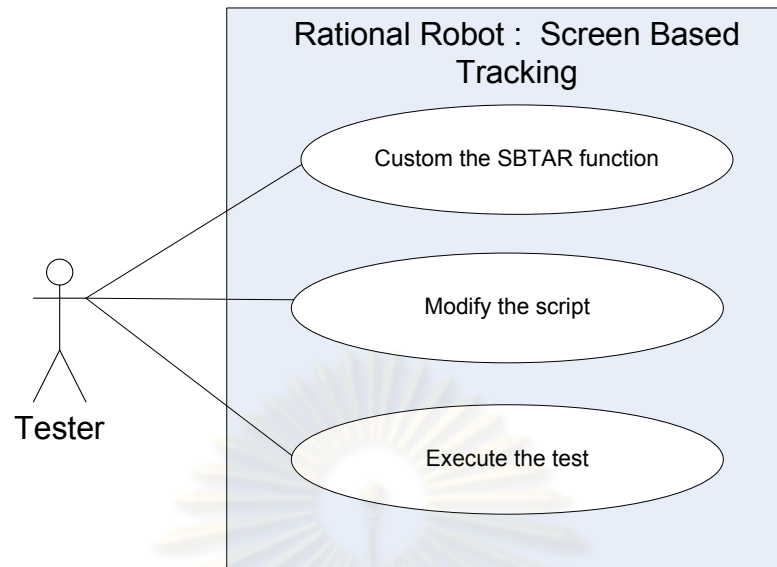


Figure 3.10: Use case diagram of ScreenBasedTrackingARray function of SBTAR

Figure 3.10 shows how the testers use the screen based tracking feature of the SBTAR method. The diagram shows the activities that testers have interact with ScreenBasedTrackingARray function of the SBTAR method. In order to use the ScreenBasedTrackingARray function, the first activity that tester have to do is implementing the ScreenBasedTrackingARray function. Testers have to implement the function in the library file of Rational Robot. Modifying the test script is the next activity. The modification can be completed by putting important key-words, parameters and procedures required by the ScreenBasedTrackingARray function. The last is executing the script to start the automated testing. In the next section each of activity will be described. In the next section each of activity will be described. Moreover, the class diagram of the screen based tracking feature of the SBTAR method shows in Figure 3.11.

จุฬาลงกรณ์มหาวิทยาลัย



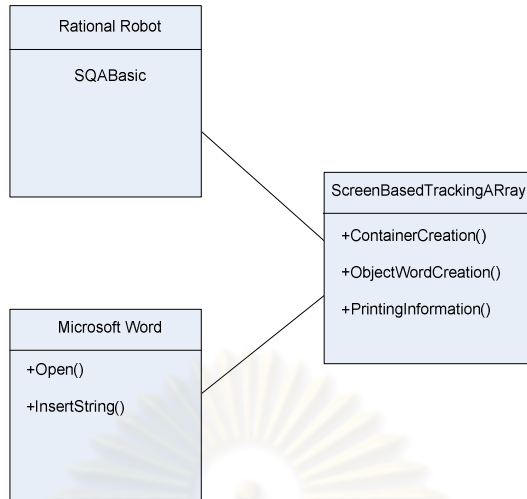


Figure 3.11: Class diagram of ScreenBasedTrackingARray function of SBTAR

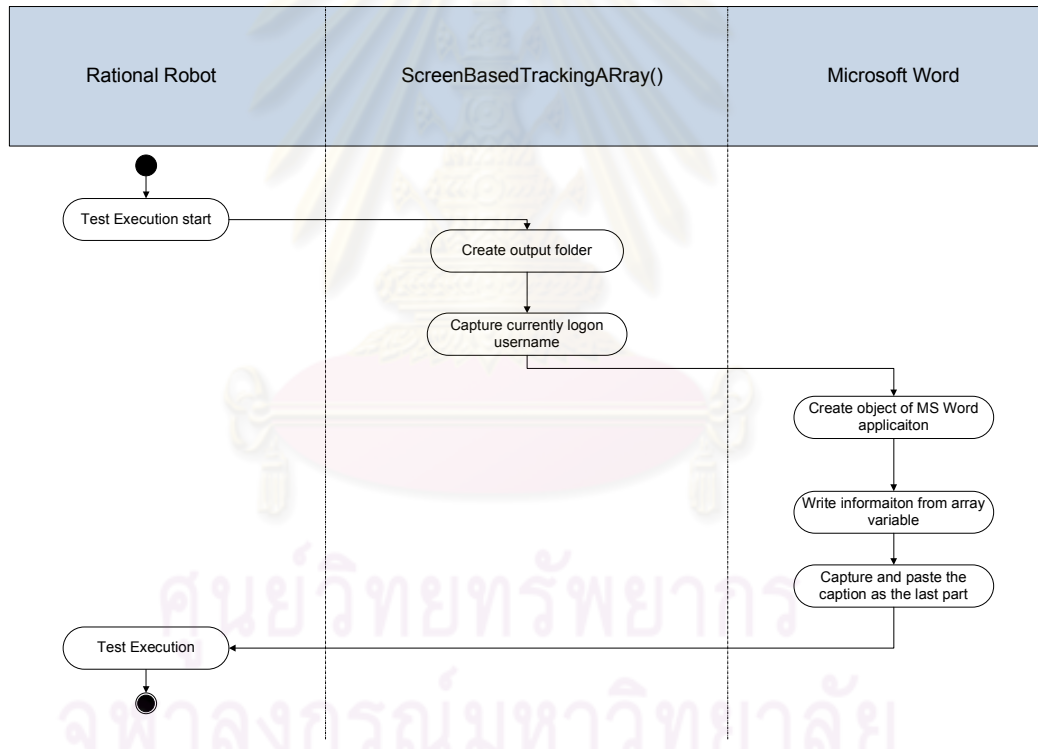


Figure 3.12: Activity diagram represents ScreenBasedTrackingARray function of the SBTAR.

According to the activity diagram in Figure 3.12, the processes of ScreenBasedTrackingARray function can be described as follow.

- (1) Checks an output folder for the folder name which is the execution date; if it is not available then a new folder will be created.
- (2) Captures the current users' logon names and assigns to a variable.
- (3) Creates an object of Microsoft Word application.
- (4) Writes necessary information, such as name of the test script, logon name and date-time of execution, into Microsoft Word.
- (5) Records each testing step until reaching to the last step.
- (6) Prints the last screen by pressing "Prtsc" (PrintScreen) to capture the screen into the clipboard and paste it at the end of file.

As mentioned about the limitation of availability of the script, the ScreenBasedTrackingARray function has been implemented in the library file in order to allow scripts and other library source files can access to this function. The next section will describe the implementation of the ScreenBasedTrackingARray function in the library source file.

#### *A Custom Function*

The custom function will be implemented as a function in the library file, .sbl, because this function must be available for every script when needed. The function that was created in the library file is shown in the example presented in Figure 3.13.

```

'-----
'ScreenBasedTrackingARray (SBTAR)
'Description : This function supports to write contents from Array to MS Word document
.
'Parameters : sDocPath => Path of Documents
.             sFileName => Name of Result log file
.             iCount   => Count of row in ExcelFilePath
.             sErrorMessage => Error Message
.             vValues() => Array which keep the wording step
.
'Return Values : None
.
'Author : Noppakit.N
'Date Created : Apr 27 2009
'Comments :
.
'Modified By :
'Date Modified :
'-----
Function ScreenBasedTrackingARray(sDocPath As String,sFileName As String , _
                                iCount As Integer,sErrorMessage As String, _
                                vTestStep() As Variant)

```

Figure 3.13: Implemented ScreenBasedTrackingARray in the library file, .sbl.

After completing the implementation of the custom function in the library file, the next step is to declare the function in the header file in order to provide access permission to other scripts or other library files. Figure 3.14 presents the declaration of the function in the header file.

```

Declare Function ScreenBasedTrackingARray BasicLib "MasterLib"(sDocPath As String, _
                                                             sFileName As String, _
                                                             iCount As Integer, _
                                                             sErrorMessage As String, _
                                                             vTestStep() As Variant)

```

Figure 3.14: Declare function ScreenBasedTrackingARray in the header file

Once the function has been implemented, the recorded test scripts must customize by putting the specific syntaxes and the important values in order to apply the function to use with the scripts. Thus, customization of the scripts is described in the next section.

### Script Customization

The script file must be customized by specifying the values to variables and adding them to the script. SBSTAR needs 5 parameters which use in its process which are:

1. "sDocPath" is a parameter storing a value of the result log container.
2. "iCount" is a parameter storing the testing steps of the test.
3. "sFileName" is a parameter to store the result log's name.
4. "sErrorMessage" is a parameter storing the result's description.
5. "vTestStep ()" is an array variable storing the descriptions of each testing step.

An example of a value storing in the "sDocPath" is drawn below.

```
sDocPath = "C:\Master\"
```

The second parameter is "iCount". It is used to indicate the sequence of testing step. This parameter must be reset to "0" every time before executing a new test. After resetting the value of "iCount", testers have to add description of each test step by assigning a testing description to the array variable name "vTestStep()"; then, "iCount" value is increasing by 1. The example of "iCount" and "vTestStep()" are presented below.

```
iCount = 0
iCount = iCount + 1
vTestStep(iCount) = "Start web browser and try www.chula.ac.th"
StartBrowser http://www.chula.ac.th/, "WindowTag=WEBBrowser"
Window WMaximize, "", ""
```

For assigning a value to "sErrorMessage", testers can insert into a part of the validation scripts. The example of assigning the "sErrorMessage" into the validation scripts is written below.

```

Delayfor(3000)
iResult = SQAGetProperty("Type=Window;WindowTag=WEBBROWSER", _
    "Caption",sResult)
If instr(sResult, "Chulalongkorn University") <> 0 then
    sErrmessage = "Pass : Webpage load successful"
    sFileName = "Valid_Master.doc"
Else
    sErrmessage = "Fail : Webpage load unsuccessful"
    sFileName = "Invalid_Master.doc"
End If

```

The last step in customizing the script is to put the 'Call' statement to gain access to functions in the library file. The following statement is an example of the call statement.

```

Call ScreenBasedTrackingARray(sDocPath, sFileName, iCount, msgtext, vTestStep())

```

After calling ScreenBasedTrackingARray function, the result log is created and presented the defects of software during the test execution. Thus, testers can easily locate the problems on the application. The next section describes the screen management that supports testers to locate errors.

### 3.2 Applying the SBTAR method to a test

After implementing two functions "StatementTransformation" and "ScreenBasedTackingARray", "SBTAR" can create the highest benefit of Rational Robot. Although these two functions are implemented and embedded to Rational Robot, there is no complication in using Rational Robot. Thus, testers can learn how to use its features and functions. However, testers need to know the prerequisite setting and information that should be prepared before executing the function.

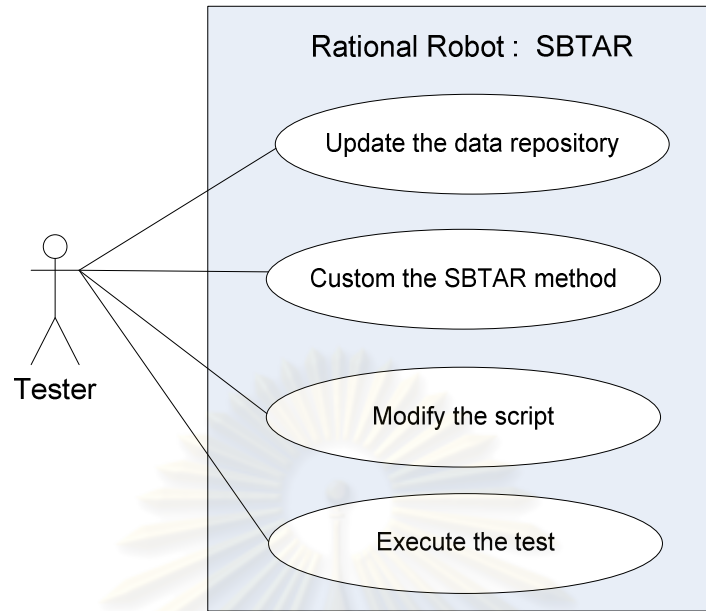


Figure 3.15: Use case diagram of the SBTAR method

Figure 3.15 shows how the testers use the SBTAR method. The diagram shows activities that testers interact with the SBTAR method. In order to apply the SBTAR method to the test script, testers have to maintain the test data that stored in the repository. Implementing the StatementTransformation function and the ScreenBasedTrackingARray function are performed. These implementation will be installed in the library file of Rational Robot. The next activity is modifying the test script by putting the important key-words, parameters and procedures that required by the functions. The last is executing the script to start the automated testing. In the next section each of activity will be described. Moreover, the class diagram of the SBTAR method shows in Figure 3.16.

จุฬาลงกรณ์มหาวิทยาลัย

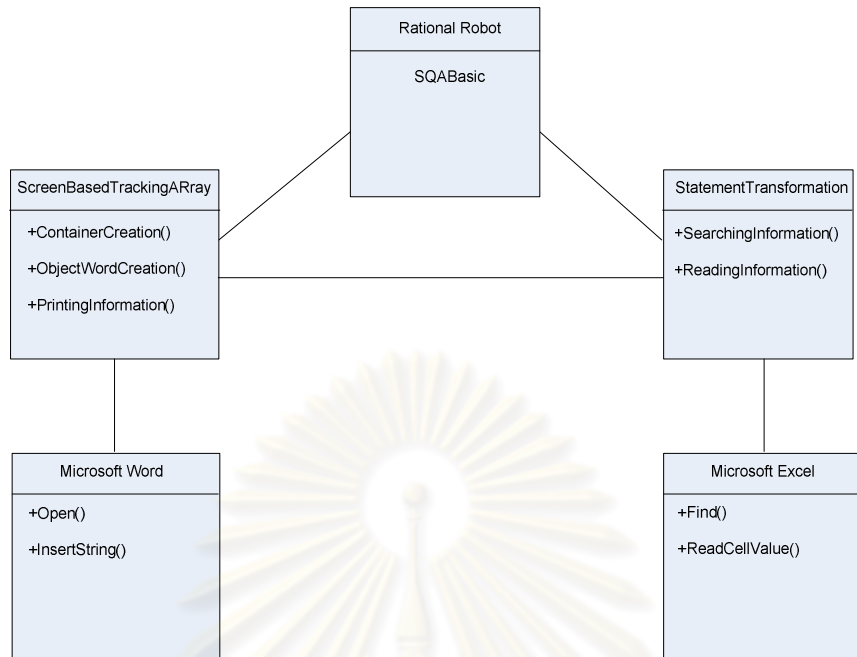


Figure 3.16: Class diagram of the SBTAR method

The main idea of implementing SBTAR is to reduce manual processes by automating the testing functions of the tools as many as possible. Referring to Figure 3.17, the test starts when testers execute Rational Robot by performing the testing steps recorded in the script. Verification points will be verified to check correctness of the application under the test. During the processes, Rational Robot calls the StatementTransformation function to obtain the baseline values of the verification process. In the verification process, the Rational Robot will compare between expected values and actual values. After this function completes the tasks and reaches to the last testing step or a failure of the verification process occurs, the ScreenBasedTrackingARray function will be started. As the result of running this function, an error report is generated in the Microsoft Word format (.Doc extension) and stored in a specific path.



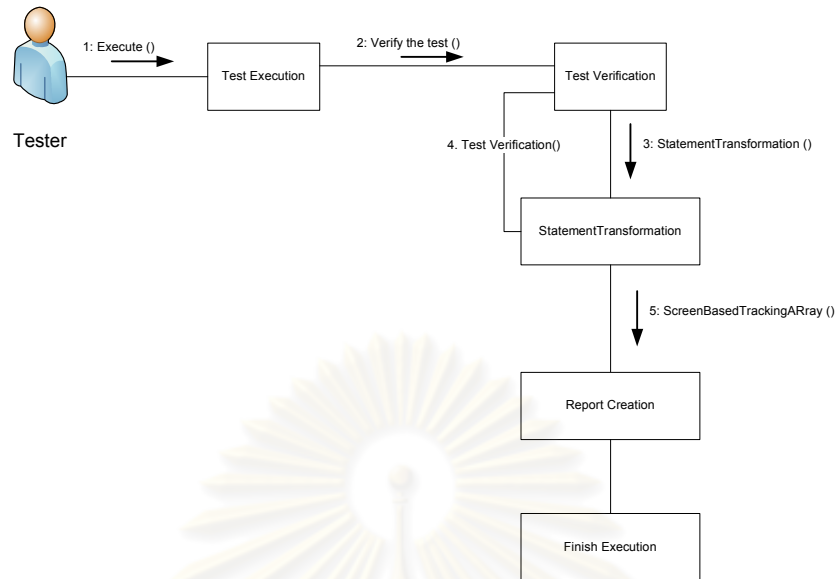


Figure 3.17: A collaboration diagram of SBTAR when integrated two functions

Since the StatementTransformation function implements a centralized database and two additional functions mentioned above are implemented in the library file, many testers can perform their testing concurrently. Figure 3.18 shows the collaboration diagram of the SBTAR method using by 2 testers. The experiment demonstrating the test with SBTAR is described in the next section.

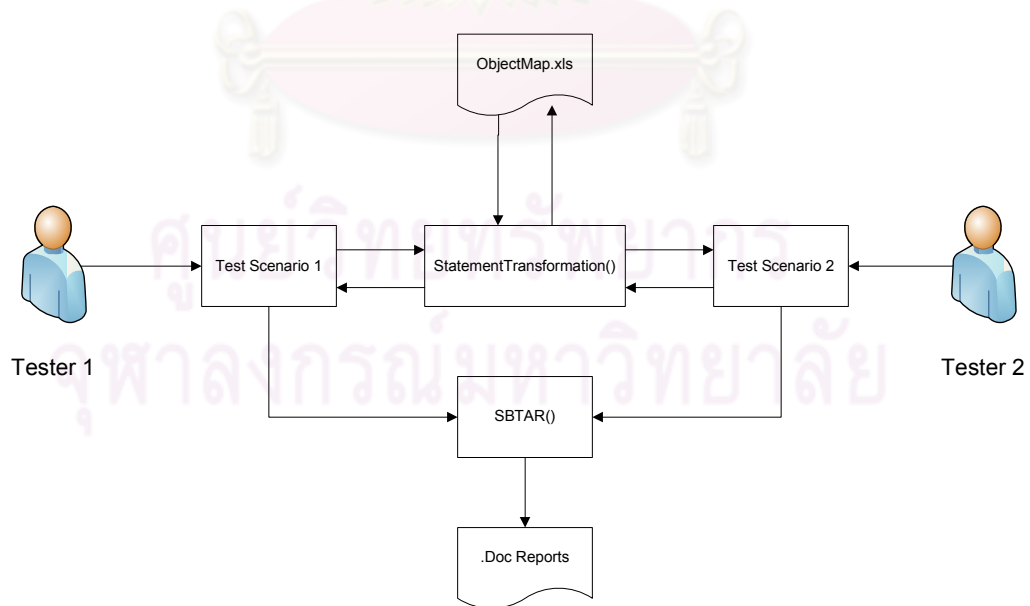


Figure 3.18: A concurrent usage of the SBTAR method

### 3.2.1 An example of applying the SBTAR method to a test

The experiment to determine the efficiency of the SBTAR method is performed on IBM Rational Robot version 7.0.1. In addition, the test in this research was performed on the sites of Chulalongkorn University, checking for defects on the display procedure. The expected result is that the SBTAR method reduces times of implementing and maintaining scripts. Therefore, performance and productivity of testers is improved. Figure 3.19 presents the index page of Chulalongkorn University website.

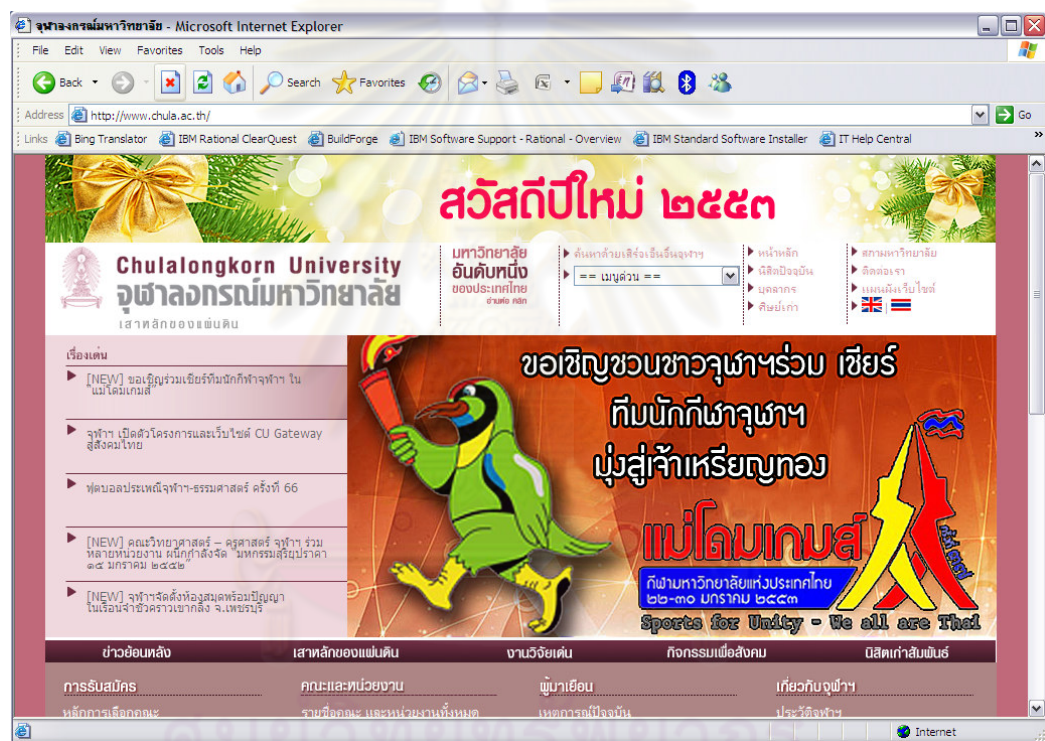


Figure 3.19: An index page of Chulalongkorn University website

#### 3.2.1.1 Testing Scenario on Website of Chulalongkorn University

Before starting the test on Chulalongkorn University's website (CUW), the URL is "www.chula.ac.th", the characteristics of this web must be clearly defined. The CUW constructs by many pages such as "Student page", "Staff page", "Alumni page", "The Council", "Contact us", and "Site map". However, this experiment will perform on the following pages; Figure 3.20 is all captured pages in the test.

- หน้าหลัก (Main) which the caption of this web page is “จุฬาลงกรณ์วิทยาลัย”.
- นิสิตปัจจุบัน (Student) which the caption of this web page is “นิสิตปัจจุบัน”.
- บุคลากร (Staff) which the caption of this web page is “บุคลากร”.
- ศิษย์เก่า (Alumni) which the caption of this web page is “ศิษย์เก่า”
- สภามหาวิทยาลัย (The Council) which the caption of this web page is “ติดต่อศูนย์สื่อสารองค์กร”.
- ติดต่อเรา (Contact us) which the caption of this web page is “ติดต่อเรา”.
- แผนผังเว็บไซต์ (Site map) which the caption of this web page is “Site Map”.

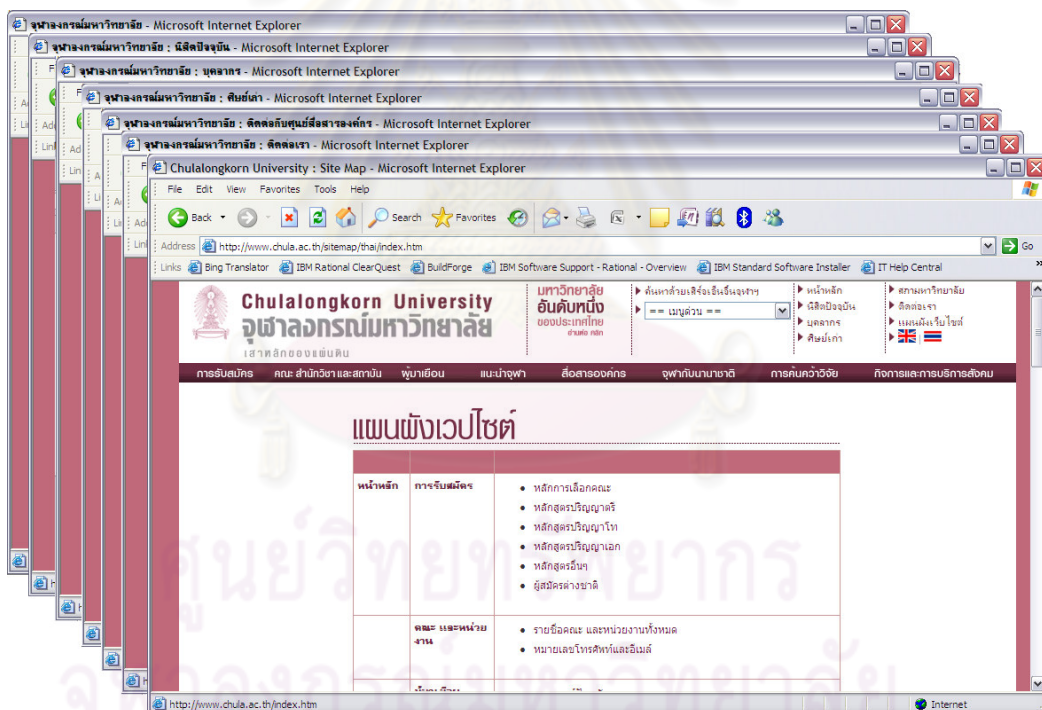


Figure 3.20: Captions of the test web pages

Users can access the required web page by clicking on the link which is located on the top right of the current web page. The testing conditions and cases are drawn below.

The cases that will be applied in the test are:

1. Verifying the completeness of loading the pages of the CUW,
2. Verifying the load time for each the page of the CUW based on the defined time boundary.

The testing scenario for these cases is defined as follows.

1. Open web browser with URL "www.chula.ac.th".
2. Click on the link located on the top right of the screen, such as "นิติ  
ปัจจุบัน", as shown in Figure 3.21.
3. Wait until the page has been successfully loaded; if the page cannot be loaded within 1 minute, the function will be considered as "fail".
4. Verifying the caption of the page, if the value has matched with the expected. Therefore, the result is "pass".
5. If the test does not "fail", the test will continue on the next page by repeating the step 2, 3 and 4 until the last test case is reached.

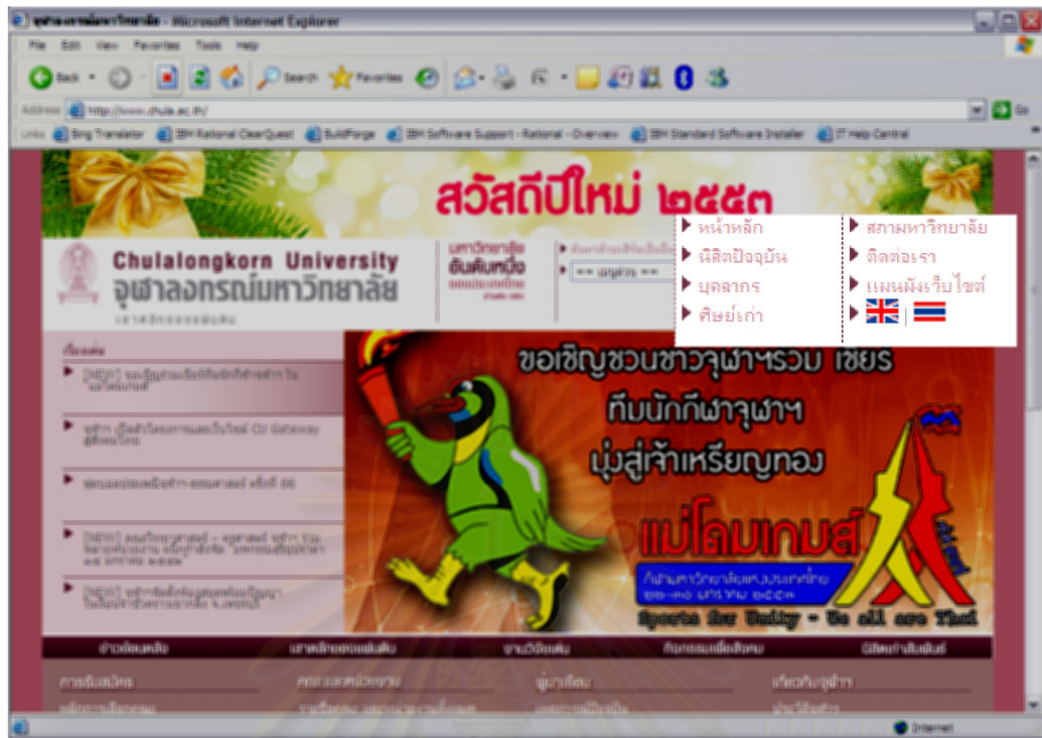


Figure 3.21: Links to other pages on the CUW

As mentioned above, the verification of the correctness of a page is checking the caption of the browser that has been changed according to the changed page of website. The script that generated from Rational Robot for the scenario mentioned above will explained in the next section.

### 3.2.1.2 Generating a script from Rational Robot

The script file will be generated automatically by Rational Robot when recording script is finished. The script file is contains information that are crucial for performing the automated testing of Rational Robot. The information that Rational Robot have capture and store in the script are keyword that represent user actions, object properties, comments, and keyword of functionality of Rational Robot. Figure 3.16 shows the generated script from Rational Robot that recording the scenario.



```

'$include "global.SBH"
Sub Main

    Dim Result as Integer

    'Initially Recorded: 18/11/2552 22:50:12
    'Script Name: Master

    'Start Browser with URL of web application
    StartBrowser "http://www.chula.ac.th", "WindowTag=WEBBrowser"
    Window SetContext, "WindowTag=WEBBrowser", ""
    Window WMaximize, "", ""

    'Verifying "หน้ามหาวิทยาลัย" page
    HTMLLink Click, "HTMLText=หน้ามหาวิทยาลัย", ""
    Result = HTMLLinkUP (CompareProperties, "HTMLText=หน้ามหาวิทยาลัย;Index=2", "UP=Object Properties")

    'Verifying "หน้ามหาวิทยาลัย" page
    HTMLLink Click, "HTMLText=หน้ามหาวิทยาลัย", ""
    Result = HTMLLinkUP (CompareProperties, "HTMLText=หน้ามหาวิทยาลัย;Index=2", "UP=Object Properties1")

    'Verifying "บุคลากร" page
    HTMLLink Click, "HTMLText=บุคลากร", ""
    Result = HTMLLinkUP (CompareProperties, "HTMLText=บุคลากร;Index=2", "UP=Object Properties2")

    'Verifying "หน้าเว็บไซต์" page
    HTMLLink Click, "HTMLText=หน้าเว็บไซต์", ""
    Result = HTMLLinkUP (CompareProperties, "HTMLText=หน้าเว็บไซต์;Index=2", "UP=Object Properties3")

    'Verifying "สารทวารวดี" page
    HTMLLink Click, "HTMLText=สารทวารวดี", ""
    Result = HTMLLinkUP (CompareProperties, "HTMLText=สารทวารวดี;Index=2", "UP=Object Properties4")

    'Verifying "หน้าเว็บไซต์" page
    HTMLLink Click, "HTMLText=หน้าเว็บไซต์", ""
    Result = HTMLTableUP (CompareProperties, "HTMLText=หน้าเว็บไซต์;Index=2", "UP=Object Properties5")

    'Verifying "หน้าเว็บไซต์" page
    HTMLLink Click, "HTMLText=หน้าเว็บไซต์", ""
    Result = HTMLLinkUP (CompareProperties, "HTMLText=หน้าเว็บไซต์;Index=2", "UP=Object Properties6")

    'Close Win
    Window CloseWin, "", ""

End Sub

```

Figure 3.22: The script that generated from Rational Robot

From Figure 3.22, ① shows the "\$include" statement that testers specify the name of the header file in order to refer to the function(s) which are implemented in the library file. ② presents the variable declaration section in the script. Rational Robot provides comment capability that testers can describe the testing steps on each line of code. The comments, user actions, and verification processes are recorded in the script as showing in ③.

According to the mechanism of SBTAR, captions of the current version of the CUW have been recorded as the expected value in the ObjectMap file. The expected value will be used to compare with the actual value in the verification process. The preparation of ObjectMap that will be used in this example will be presented in the next section.

### 3.2.1.3 ObjectMap file preparation

Normally, the test data preparation for Rational Robot will be performed on a datapool which is not convenience. Since Microsoft Excel has the capabilities on calculate the data such as sum, average or other drag and drop features, then preparing the data using Microsoft Excel is easier and faster. Figure 3.23 shows the input screen of a datapool.

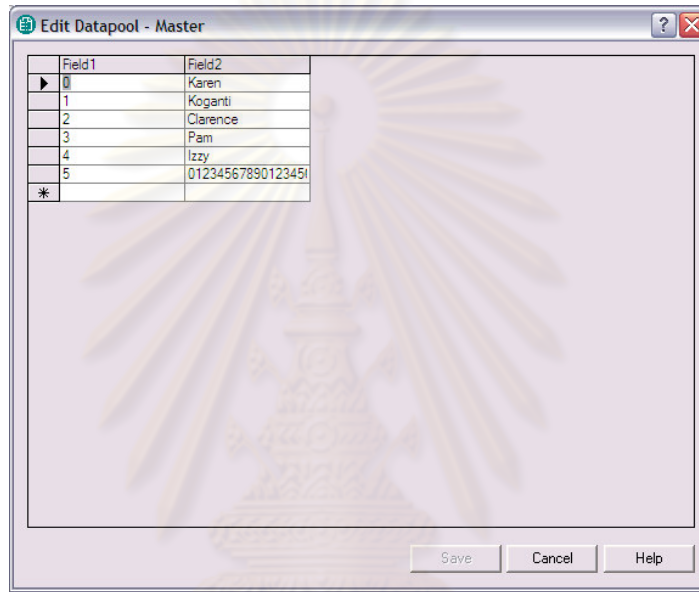


Figure 3.23 Input screen of the datapool of Rational TestManager

The ObjectMap file used by the SBTAR method is prepared by the standard and widely used software which is Microsoft Excel. After the file ObjectMap.xls was created, testers open the file and specify values for defined columns. Figure 3.24 shows examples of all values in the defined columns that are required for the SBTAR method.

A	B	C	D	E	F
Object Name	Object Appearance	Verification Property	Baseline Value	Message for Pass verification	Message for Fail verification
HomeThai	Type=HTMLLink;HTMLText=หน้าหลัก	innerText	จุฬาลงกรณ์	Page was load successfully	"หน้าหลัก" page cannot load successfully
StudentThai	Type=HTMLLink;HTMLText=นักศึกษาปัจจุบัน	innerText	นักศึกษปัจจุบัน	Page was load successfully	"นักศึกษปัจจุบัน" page cannot load successfully
StaffThai	Type=HTMLLink;HTMLText=บุคลากร	innerText	บุคลากร	Page was load successfully	"บุคลากร" page cannot load successfully
AlumniThai	Type=HTMLLink;HTMLText=ศิษย์เก่า	innerText	ศิษย์เก่า	Page was load successfully	"ศิษย์เก่า" page cannot load successfully
CouncilThai	Type=HTMLLink;HTMLText=สภามหาวิทยาลัย	innerText	สภามหาวิทยาลัย	Page was load successfully	"สภามหาวิทยาลัย" page cannot load successfully
ContactThai	Type=HTMLLink;HTMLText=ติดต่อเรา	innerText	ติดต่อเรา	Page was load successfully	"ติดต่อเรา" page cannot load successfully
MapThai	Type=HTMLLink;HTMLText=แผนที่บริเวณ	innerText	Site Map	Page was load successfully	"แผนที่บริเวณ" page cannot load successfully

Figure 3.24: Examples of specifying values in ObjectMap.xls



Additionally, testers can utilize this file to have much ease of use ObjectMap.xls by entering colors or specify format freely which will not affect any functionalities of the SBTAR method.

#### 3.2.1.4 Apply the SBTAR method to a script

When applying the SBTAR method to the test, scripts which Rational Robot has generated when finish recording needs to be modified. First of all, the header file that referring to the library contains the SBTAR methods must to include to the script.

To include the header, **1** needs to be updated by put the header file name as shows in Figure 3.25.

```
'$Include "global.SBH"
'$Include "MasterHeader.SBH"
Sub Main

  Dim Result as Integer

  'Initially Recorded: 18/11/2552 22:50:12
  'Script Name: Master
```

Figure 3.25: Including the header file name

Additionally, before customizing the script, variables used by StatementTransformation and ScreenBasedTRackingArray functions must be declared. After all variables have been declared, the initial values of those variables are defined.

In order to declare the variables **2** has to be updated. Figure 3.26 shows the variables declaration for the SBTAR method.

```

'$Include "global.SBH"
'$Include "MasterHeader.SBH"

Sub Main

    Dim sObjectFile           As String
    Dim sWorksheet            As String
    Dim vObjectMAP()          As Variant
    Dim sDocPath               As String
    Dim sFileName              As String
    Dim iCount                 As Integer
    Dim sErrorMessage         As String
    Dim vTestStep(1 to 100)   As Variant

    'Initially Recorded: 18/11/2552 22:50:12
    'Script Name: Master

    sDocPath = "C:\RobotNote\Master_Project\Result_Logs\"
    sFileName = "Valid_Master_Script_" & sDate & ".Doc"
    sWorksheet = "Sheet1"
    sObjectFile = "C:\RobotNote\Master_Project\TestData\ObjectMap.xls"

```

Figure 3.26: Declaring the important variables

Rational Robot provides a set of commands that testers can use to implement the scripts for the test. These commands support testers in creating various test cases using a small number of codes. The examples of these commands are If-Else, Do-Loop, For-Loop, Delay, Call statement, Goto and SQAGetProperty, etc. The

③ needs to be updated by adding the verifications and important commands of the SBTAR method. Figure 3.27 shows the sample script that shows the use of such commands and "Call" statement for the SBTAR method.

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

```

'$Include "global.SBH"
'$Include "MasterHeader.SBH"

Sub Main

    Dim Result
    Dim sResult As String

    Dim sObjectFile As String
    Dim sWorksheet As String
    Dim vObjectMAP() As Variant

    Dim sDocPath As String
    Dim sFileName As String
    Dim iCount As Integer
    Dim sErrorMessage As String
    Dim vTestStep(1 to 100) As Variant

    Dim sDate As String
    Dim vScenario() As Variant

    'Initially Recorded: 18/11/2552 22:50:12
    'Script Name: Master

    sDate = Format (Date,"ddmmyy")

    On error goto SBTAR

    sDocPath = "C:\RobotNote\Master_Project\Result_Logs\"
    sFileName = "Valid_Master_Script_" & sDate & ".Doc"
    sWorksheet = "Sheet1"
    sObjectFile = "C:\RobotNote\Master_Project\TestData\ObjectMap.xls"

    iCount = iCount + 1
    vTestStep(iCount) = "Start web browser and try www.chula.ac.th"

    StartBrowser "http://www.chula.ac.th", "WindowTag=WEBBrowser"

    Window SetContext, "WindowTag=WEBBrowser", ""
    Window WMaximize, "", ""

    Call Excel_Read_CellValue (sObjectFile, vScenario(), 1, "A", "Sheet2")

    i = 1

    Do while i <= ubound(vScenario,1)

        Call StatementTransformation (sObjectFile, sWorksheet, vScenario(i), vObjectMAP())

        iCount = iCount + 1
        vTestStep(iCount) = "Verify correctness of " & vObjectMAP(1) & " page"

        HTMLLink Click, vObjectMAP(2), ""

        DelayFor (2000)
        Result = SQAGetProperty("Type=Window;WindowTag=WEBBrowser","Caption",sResult)

        'Verify that webpage was loaded successfully
        IF instr(sResult, vObjectMAP(4)) <> 0 THEN
            vTestStep(iCount) = vTestStep(iCount) & " << " & vObjectMAP(5)
        ELSE
            sErrorMessage = vObjectMAP(6)
            sFileName = "Invalid_Master_Script_" & sDate & ".Doc"
            Goto SBTAR
        End IF

        i = i + 1

        DelayFor (3000)
        HTMLLink Click, "HTMLText=หน้า" & i & "หน้า", ""

    Loop

    SBTAR:

    Call ScreenBasedTrackingArray(sDocPath,sFileName,iCount,sErrorMessage,vTestStep())

    'Close Win
    Window CloseWin, "", ""

End Sub

```

Figure 3.27: Applying the SBTAR method to a script

After the scripts have been customized to work with the SBTAR method, Rational Robot will execute the test according to the test steps based on the scripts. Once the test has completed, the ScreenBasedTrackingARray functions will create a report in the Microsoft Word format (.doc extension) using the value defined in the variable "sFileName", at the path declared in the "sDocPath" variable. The result of the test generated from the SBTAR method will be shown in the next section.

### 3.2.1.5 Result log generated from the SBTAR method

When finishing the test, the report is created in the folder that was created with the name according to the execution date. This folder is stored in the path specified in the sDocPath variable. Figure 3.28 presents how the SBTAR method organized the folders for the generated reports.

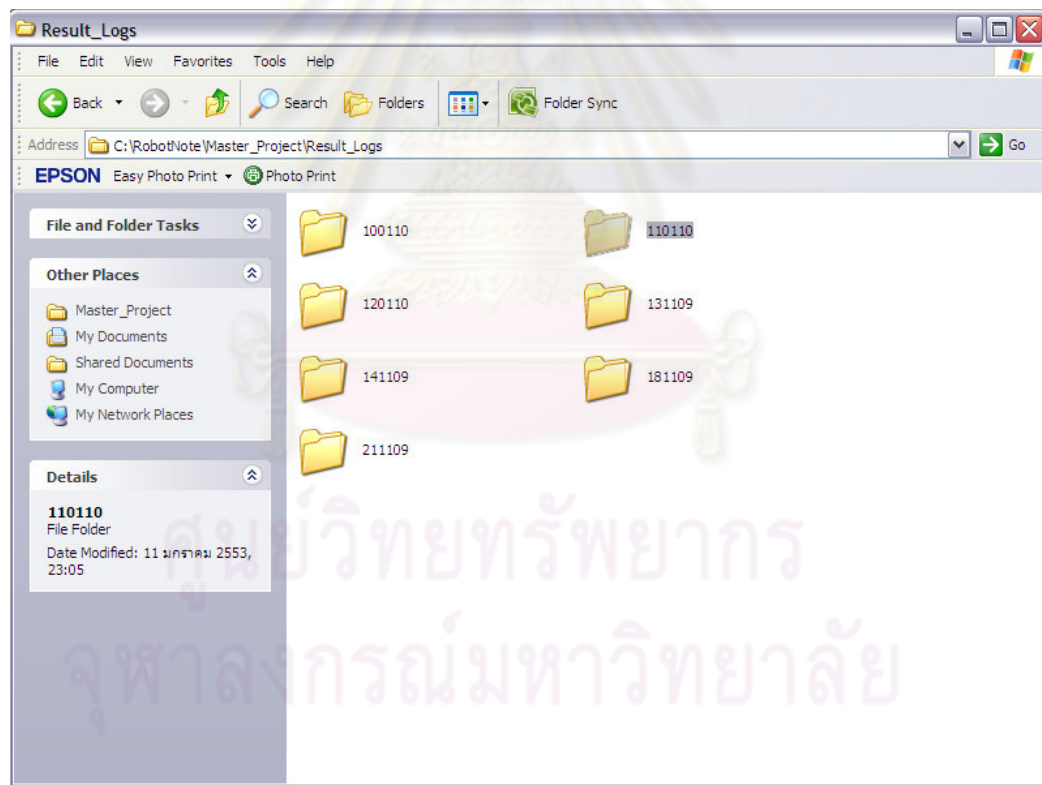


Figure 3.28: Folders for generated reports from SBTAR

Figure 3.29 shows the result logs that created from the scripts mentioned previously. According to the script with the verification process, it will assign a new

name for generating the result log when the result of verification is 'fail'. Such name of the result log helps testers or persons who involved in the test easily recognize the verifying result.

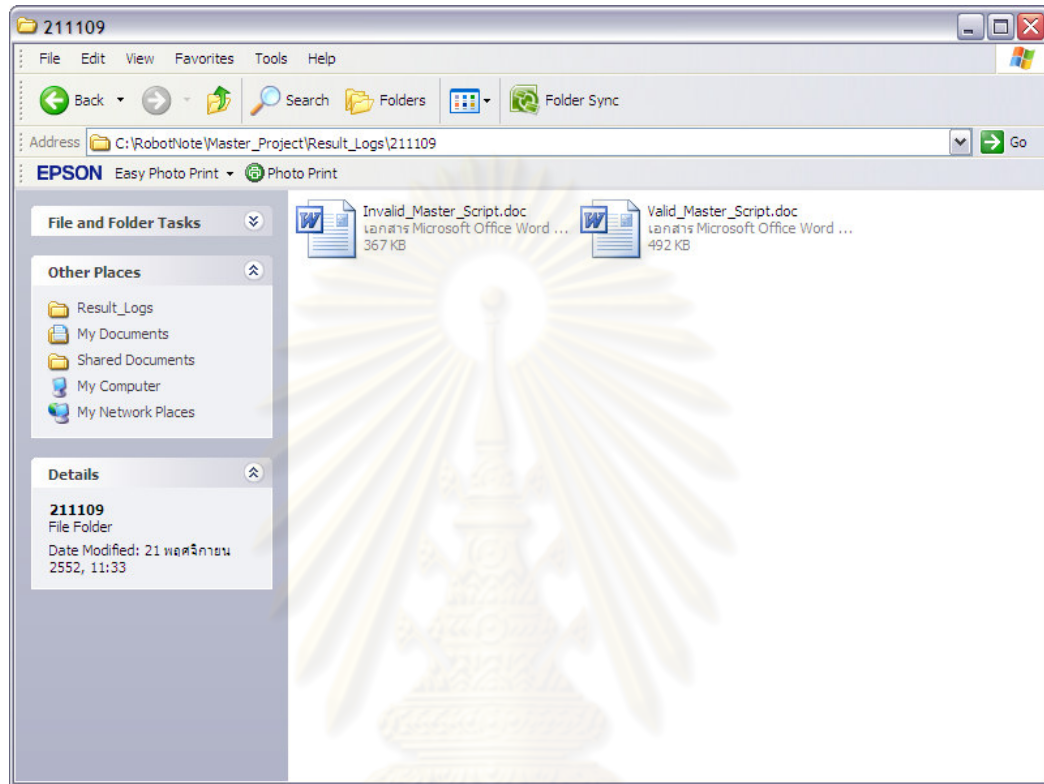


Figure 3.29: Report files which generated from the SBTAR

In the result log that is created from the SBTAR method, it is containing the useful information that reading from the ObjectMap file. The result log is created based on the structure that has been defined in the ScreenBasedTrackingARray function. Figure 3.30 presents an example of a report when all test steps pass.

จุฬาลงกรณ์มหาวิทยาลัย

This report of Valid\_Master\_Script\_120110

Submitted by : [NoppakitN](#)

Date and Time of report : 12/1/2010 9:12:05

1. Start web browser and try [www.chula.ac.th](#)
2. Verify correctness of [HomeThai](#) page << Page was load successfully
3. Verify correctness of [StudentThai](#) page << Page was load successfully
4. Verify correctness of [StaffThai](#) page << Page was load successfully
5. Verify correctness of [AlumniThai](#) page << Page was load successfully
6. Verify correctness of [CouncilThai](#) page << Page was load successfully
7. Verify correctness of [ContactThai](#) page << Page was load successfully
8. Verify correctness of [MapThai](#) page << Page was load successfully

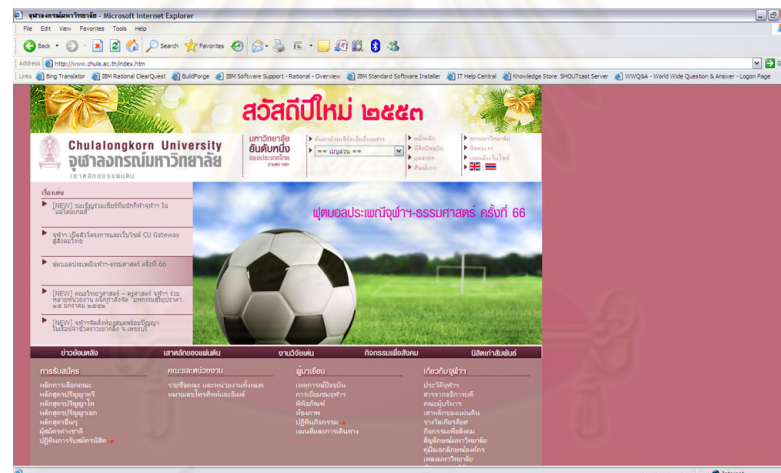


Figure 3.30: Example of report for “Pass” result

The result log that created from the SBTAR method uses the same logic in creating the result log for “Pass” or “Fail”. The difference is that the information that is used for creating the result log has been read from different fields in the ObjectMap file. Figure 3.31 presents an example of a report for “Fail” result.



This is report of Invalid\_Master\_Script\_120110

Submitted by : NoppakitN

Date and Time of report : 12/1/2010 9:18:45

1. Start web browser and try www.chula.ac.th
2. Verify correctness of HomeThai page << Page was load successfully
3. Verify correctness of StudentThai page << Page was load successfully
4. Verify correctness of StaffThai page << Page was load successfully
5. Verify correctness of AlumniThai page << Page was load successfully
6. Verify correctness of CouncilThai page << Page was load successfully
7. Verify correctness of ContactThai page << Page was load successfully
8. Verify correctness of MapThai page << "แผนที่จังหวัด" page cannot load successfully

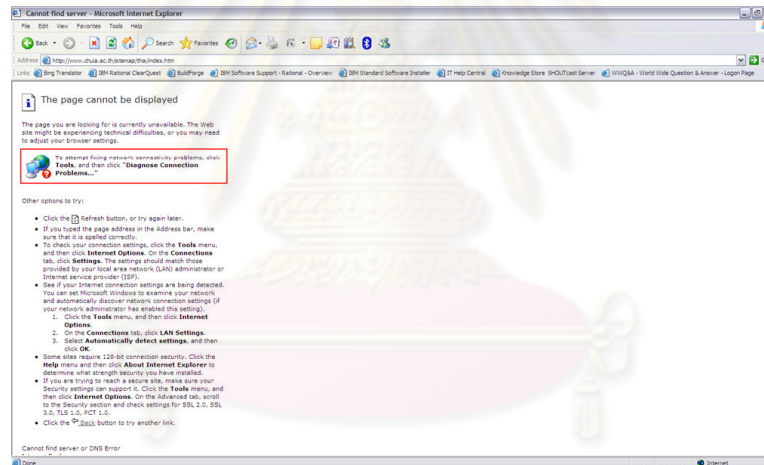


Figure 3.31: Example of a report for "Fail" result

The example mentioned in this section has demonstrated that applying the SBTAR method will obtain a useful report which contains readable information better than the original tools. Therefore, testers can locate the defects in a small amount of time comparing to the legacy methods. The next chapter will describe the empirical study when implementing the SBTAR in Rational Robot.



## CHAPTER 4

### EMPIRICAL STUDY

This chapter aims to demonstrate the satisfaction of users after implementing the SBTAR method for 2 months in a banking system. In this system, there are two main teams with individual responsibilities: tester teams and development teams. Details of this organization will be described in Section 4.1. Section 4.2 describes the criteria for measuring the users' satisfaction while Section 4.3 is the evaluation process. Then, Section 4.4 presents the summary of the evaluation.

#### 4.1 Experimental sample

In this experiment, there are 24 persons involved in the evaluation of the SBTAR method. These persons can be classified into 3 independent groups: the project owner, the development team, and the test team. The first group is the project owner or end users who initiated the requirements to implement, create or enhance the ability of applications; this group has 5 members during the evaluation period. After the applications have been implemented based on their requirements, they are responsible to do the UAT test and review the project documents, including the result from the SIT test.

The second team is the development team consisting of 6 members; 2 are the project managers and 4 are developers. Their responsibilities are implementing the web application that help bank's customers manage their money via the website, saving time, reducing expenses and enhancing their business processes. Moreover, they are responsible on unit testing and performance testing of the application.

The last team is the test team consisting of 13 people; 2 people work as the test managers, 2 people work as the test leads and other 9 are testers. The mission of the test team is ensuring that software has the best quality before delivering to customers. Currently, the test team use many automated testing tools to support their missions, such as Rational Robot. After the SBTAR method was created, they have

agreed to apply the SBTAR method for extending the capabilities of Rational Robot to improve their tests more effectively.

## 4.2 Criteria for Measurement

According to the standard software quality control criteria, 4 criteria are considered to indicate the satisfaction of users: usability, efficiency, reliability, and reusability. The following sections will describe on each measurement criteria.

### 4.2.1 Usability

Usability emphasizes on the ease of use of the system that helps users reach to their objectives/goals. Currently the computer software has much complexity than the legacy software. Then, the usability is counted as a significant factor for software failure. As a consequence, this factor is recognized as an important issue in the application development process. The concept of this criterion is highlighting on ease of use to achieve the purpose, instead of adding features and functions. A well-known example about usability is the "Google web" which there is only text box and only button on the page that users need not learn before using it. Thus, in order to design a high usability application, designers should concern about how to implement a qualified application which is easy to understand, easy to use, easy to learn, easy to install. In addition, the designers should not focus only a nice designed interface.

### 4.2.2 Efficiency

The efficiency of the method can be measured by considering on the resource usage during the execution. Moreover, for the efficiency measurement, reducing time and improving the productivity of the users also should be under consideration. An example of the good efficiency application, the method can be finished the tasks within small amount of time, consume low percentage of CPU and memory, and etc. Moreover, the method can help on reducing effort of the users.

#### 4.2.3 Reliability

Reliability of software is the possibility of software operations that do not contain errors in a specific time period under the specified environment. The software reliability is an important factor which affects to the quality of the system or applications. Furthermore, the software reliability indicates the completion (perfection) of design with the standard quality control mechanism in the development processes.

#### 4.2.4 Reusability

In computer science and software engineer, a reusability of software means the ability to embed an implemented module, or an object, or a project of previous developed software into new developing software without modification. The benefit of this feature is that the development time will be reduced and the developed software is reliable.

In order to evaluate these 4 factors in the SBTAR method, a questionnaire was created. This questionnaire contains 8 questions that relate to the 4 criteria mentioned above. The next section will presents the evaluate process for the SBTAR method.

### 4.3 Evaluation process

The survey was performed using a questionnaire that was created and sent to the project owners, the development team and the test team. There are 2 sessions that have been conducted in order to clarify the meaning of the survey for each team. For the first session, it has been conducted for the test team; and the second session is for the project owners and the development team. After the clarification, users complete the survey by rating each question. The rating in the survey has 4 levels: "Very Satisfied", "Satisfied", "Neutral", and "Dissatisfied". The returned surveys will be summarized in order to prepare for calculation. Figure 4.1 presents the contingency table of the measurement scores for usability of the SBTAR method.

Questions	Very Satisfied	Satisfied	Neutral	Dissatisfied
1. The method easy to adapt to your work	20	4	0	0
2. The method can help you finish your work faster	24	0	0	0
3. The method can help your work easier	19	5	0	0
Accumulate	63/3 = 21	9/3 = 3	0	0

Figure 4.1: The contingency table of the usability scores of the SBTAR method.

#### 4.4 Summary of the evaluation

##### 4.4.1 Reusability

Figure 4.2 shows the average frequency of reusability issue that 24 persons rate for the SBTAR method. It is clear that there are 21 peoples are very satisfied with the comment that the SBTAR method is easy to be installed and easy to be used. Consequently, higher productivity of the normal process can be obtained.

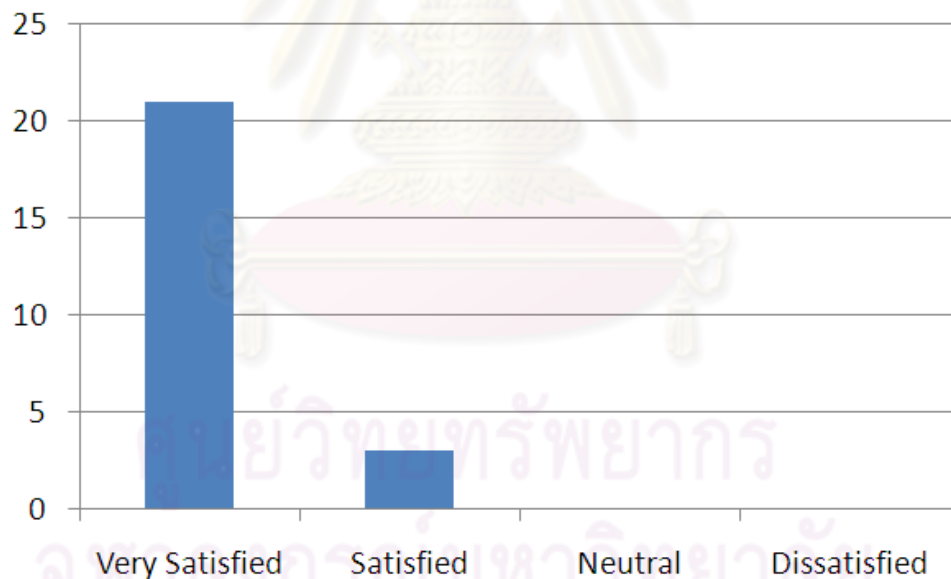


Figure 4.2: The average frequency of SBTAR in the part of Usability

#### 4.4.2 Efficiency

Figure 4.3 shows the average frequency of the rating obtained from 24 peoples measuring for the efficiency of the SBTAR method. The result has shown that there are 19 peoples are very satisfied because the results generated from the SBTAR method can improved their works with much effective.

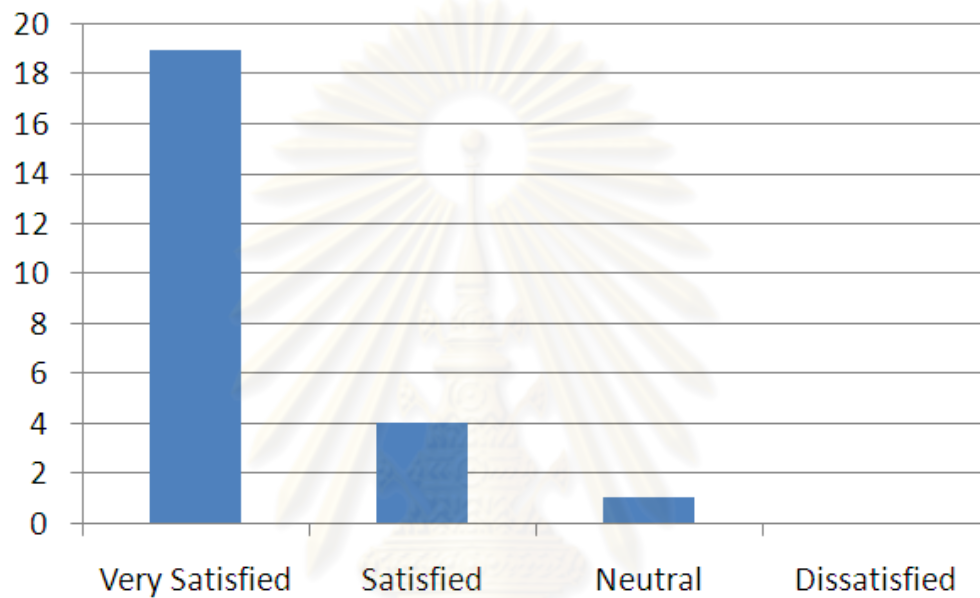


Figure 4.3: The average frequency of SBTAR in the part of Efficiency

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

#### 4.4.3 Reliability

Figure 4.4 shows the average frequency rating from 24 participants under the consideration of reliability of the SBTAR method. The bar chart shows that most people are very satisfied with the SBTAR method because it can perform a test without any errors or problems.

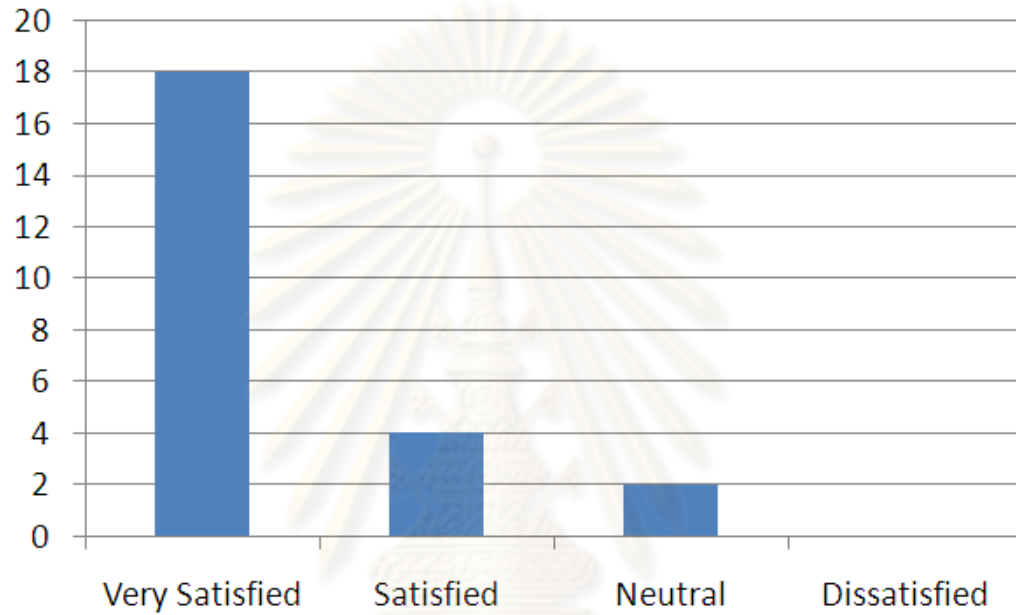


Figure 4.4: The average frequency of SBTAR in the part of Reliability

#### 4.4.4 Reusability

Figure 4.5 shows the result of average frequency rating obtained from 24 persons considering in the reusability issue of the SBTAR method. The majority of the participants agree that they are very satisfied because the implemented methods can be reused with other tasks easily.

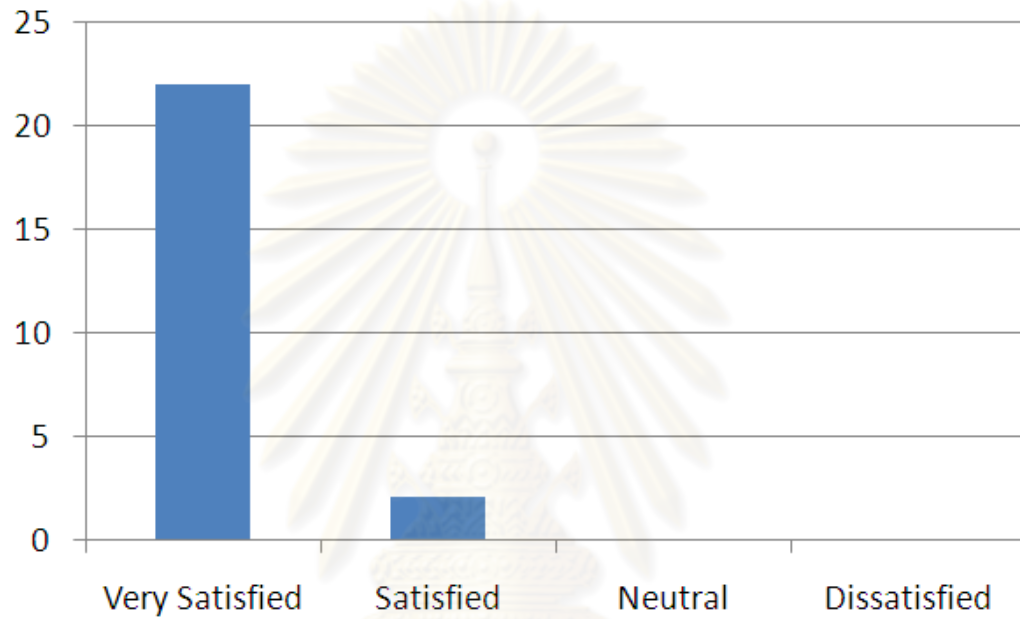


Figure 4.5: The average frequency of SBTAR in the part of Reusability

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย



#### 4.4.5 Overall satisfaction

Although the questionnaire measures individual criterion, there is a part that also measure the overall satisfaction of the SBTAR method. The result has shown that most participants are very satisfied. Thus, the SBTAR method can serve needs of every group.

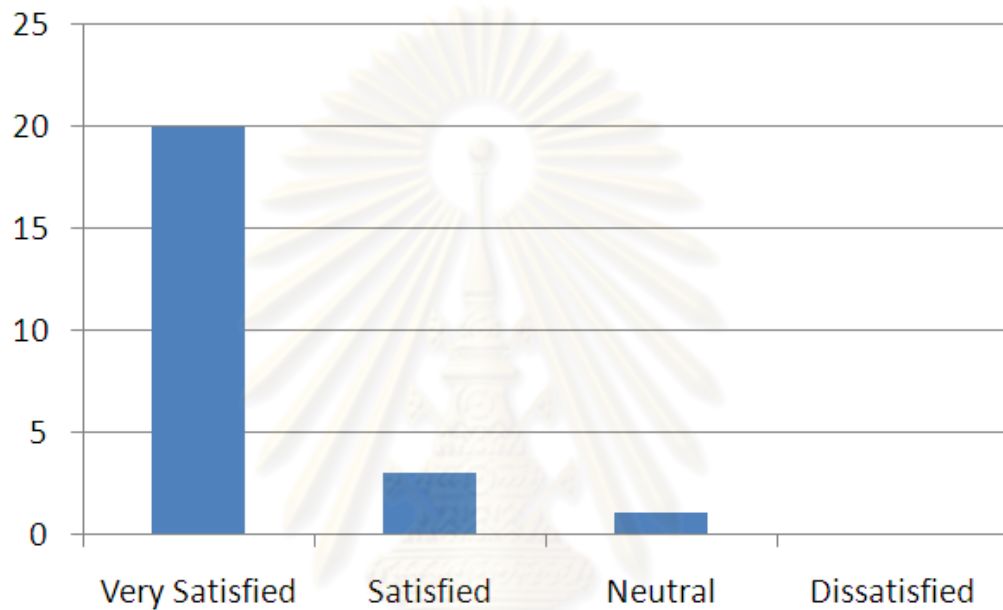


Figure 4.6: The average frequency of SBTAR in the part of Overall satisfaction

## CHAPTER 5

### DISCUSSION AND CONCLUSIONS

In this chapter, the discussion will be discussed in Section 5.1, and conclusions will be drawn in Section 5.2, followed by the future work for this thesis in Section 5.3.

#### 5.1 Discussion

One important process in software development is the verification procedure where the various tests must be performed to ensure the performance of the delivered software. Most of software developers apply automated test tools to perform this process in order to save their time and cost. Unfortunately, these tools cannot fully support the testing process because the results from the test tools may be stored in the unusable format. Therefore, the testers must perform manually analysis process in order to capture and locate defects of the software. Consequently, the SBTAR method has implemented in order to improve the reporting procedure of the test.

There are some researches such as [3] and [11] using flexibility of XML (Extensible Markup Language) technology which is more widely applied in software development. Those researches use XML in the automated testing by implementing in the express test cases, test data and convert XML to an executable format for test automation tools. However, Microsoft Excel format is widely used and much flexible than XML because it provides many manipulation functions that testers can use to prepare their works example test case, test data and test result. Thus, the SBTAR method has applied the Microsoft Excel in the operations.

Furthermore, there are some researchers who use a single repository as a center database which every component can acquire information. Central database can improve automated test in the standardized issue. Additionally, testers can save the maintenance time only by changing an error statement in the database. As the consequence, this change will be available for other testers. Since a new method

allowing testers to produce a test suitable for native language was proposed by [12], this method applied the international knowledge base which is a centralized database as same as the SBTAR's concept. However, the SBTAR uses Microsoft Excel as its centralized database to store the information where the automated scripts can be used as input. Therefore, the SBTAR is much flexible than the proposed method in [12] because the flexibility is obtained from Microsoft Excel, which supports every languages.

This thesis proposed the SBTAR method to support the testing process. This method is different from other testing methods of other testing tools since it focuses in storing testing results rather than fast execution mechanisms with difficult readable format of the output log. As a result, the users of this new proposed module can derive and fix errors of the developed software correctly and easily in a short period of time. Thus, the objective of saving time and cost for using the automate test tools is completely achieved.

## 5.2 Conclusions

In order to obtain qualified software for clients, developers must perform a good software process, including the testing procedure, to guarantee the quality of the delivered software. Although the testing process is time consuming and is the tedious task for all testers, this process needs to be performed with a high skill of software tester groups. Most testers implement software testing tools, automatic software testing, to obtain the test results. The testing scenarios are used as inputs, and the expected results are the running transactions and events stored in a log file. The most common problem of the software test tools is that this common log file is difficult to be understood and used.

In order to eliminate the limitations above, this paper proposes the SBTAR, a mechanism to manage the result log obtained from the testing process to be in the form that testers can locate and solve bugs in the software quickly and correctly. The SBTAR method has implemented to supporting the automated regression test for System Integration Test (SIT), User Acceptant Test (UAT) or testing before deploy to

production. There are 2 implemented functions that work on different purpose. The first function is StatementTransformation; this function has a capability to acquire the information that stored in the centralize database (Microsoft Excel file). This function has aims to reduce the constrains of the data pool, increase the standardize of the error messages, provide flexibility for testers in preparing test cases and testing data. The second function is ScreenBasedTrackingARray. This function focuses on the format of the result log. Since there are some issues with the original result logs that it cannot present the failure of the application under the test in the understandable format, testers have to interpret the result before using in the problem analysis phase. Therefore, the ScreenBasedTrackingARray function generates a readable and an ease of use result log for users. Moreover, the result from the experimental survey indicates that the organization gains benefits of applying the SBTAR method because the cost and time in the testing procedure are reduced.

### 5.3 Future work

In this experiment, the method name SBTAR was implemented and running on IBM Rational Robot version 7.0. There are many companies that have implemented automated testing tools. Each software companies have proposed various solutions for users, even IBM launched a new product, IBM Rational Functional Tester, and the WinRunner from HP, etc. Therefore, developing the SBTAR to work with all tools will be the further study for improving the ability of every automated testing tool. In addition, a graphic user interface should be designed and implemented in order to support the testers to be convenience in preparing the test data and using the SBTAR method.

## REFERENCES

- [1] Fewster, M., Graham, D. **Software Test Automation**. Addison-Wesley Professional, Aug 1999.
- [2] Ramler, R., Wolfmaier, K. Economic Perspectives in Test Automation: Balancing Automated and Manual Testing With opportunity Cost. **Proceedings of the 2006 international workshop on Automation of software test (2006)**: 85 - 91.
- [3] Tang, J., Cao, X. Towards Adaptive Framework of Keyword Driven Automation Testing. **Proceeding of the IEEE International Conference on Automation and Logistics (2008)**: 1631 – 1636.
- [4] Jin, L. Automated Functional Testing of Search Engine. **AST '09, ICSE Workshop on Automation of Software Test (2009)**: 97 - 100.
- [5] Wissink, T., Amaro C. Successful Test Automation for Software Maintenance. **22nd IEEE International Conference on Software Maintenance (2006)**: 265 - 266.
- [6] Volokh, E. AUTOMATED TESTING – WHY AND HOW. **INTEREX Conference (1990)**.
- [7] Ha Kim, E., Chae Na, J., Moon Ryoo, S. Test Automation framework for Implementing Continuous Integration. **ITNG '09, 6th International Conference on Information Technology (2009)**: 784 – 789.
- [8] Hui, J., Yuqing, L., Pei, L., Shuhang, G., Jing G. LKDT: A Keyword – Driven Based Distributed Test Framework. **International Conference on Computer Science and Software Engineering (2008)**: 719 – 722.
- [9] Xiaochun, Z., Bo, Z., Juefeng, L., Qiu, G. A Test Automation Solution on GUI Functional Test. **INDIN 2008, 6th IEEE International Conference on Industrial Informatics (2008)**: 1413 – 1418.
- [10] Holmes, A., Kellogg, M. Automating Functional Tests Using Selenium. **Proceeding of Agile Conference (2006)**: 6 pp. – 275.

- [11] Mu, B., Zhan, M., Lanfang, H. Design and Implementation of GUI Automated Testing Framework Based on XML. **WCSE '09, WRI World Congress on Software Engineering** (2009): 194 – 199.
- [12] Guo, X., Tay, W., Sun, T. Intelligent Multilingual Software Testing Tool. **ICNSC 2008, IEEE International Conference on Networking, Sensing and Control** (2008): 751-755.
- [13] Yu, W., Patil, G. A Workflow-Based Test Automation Framework for Web Based Systems. **ISCC 2007, 12th IEEE Symposium on Computers and Communications** (2007): 333-339.
- [14] Berner, S., Weber, R., K.Keller, R. Observations and Lessons Learned from Automated Testing. **Proceedings of the 27th international conference on Software engineering** (2005): 571 579.
- [15] Stephenson, M., Lynch, T., Walters, S. Using Advanced Tools to Automate the Design, Generation and Execution of Formal Qualification Testing. **AUTOTESTCON '96, Test Technology and Commercialization** (1996): 160 – 165.
- [16] Fiora, T.W, AU, Baker, S., Warren, I., Dobbie, G. Automated usability Testing Framework. **Proceedings of the ninth conference on Australasian user interface - Volume 76** (2008): 55 – 64.
- [17] Maogui, H., Jinfeng, W. Application of Automated Testing Tool in GIS Modeling. **WCSE '09, WRI World Congress on Software Engineering** (2009): 184 – 188.
- [18] Wu, T., Wan, Y., Xi, Y., Chen, C. Study on the automatic test framework based on three-tier data driven mechanism. **ICIS 2009, Eighth IEEE/ACIS International Conference on Computer and Information Science** (2009): 996 – 1001.
- [19] Kim Ha Eun., Jong Chae Na., Seok Moon Ryoo. Implementing an Effective Test Automation Framework. **COMPSAC '09, 33rd Annual IEEE International** (2009): 534 – 538.

- [20] Gallagher, K., Hall, T., Black, S. Reducing Regression Test Size by Exclusion. ICSM 2007, IEEE International Conference on Software Maintenance (2007): 154 – 163.
- [21] Paul C, Jorgensen. SOFTWARE TESTING: A Craftsman's Approach, Second Edition. CRC Press, May 2002.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย





APPENDIX

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## Appendix A

### Satisfaction Survey

#### SBTAR: An Enhancing Method for Automate Test Tools

**Description:** The SBTAR is a method which can enhance usability of automated test tools in a part of a result log. We would like to know your opinion on our method. Please indicate your level of satisfaction on each characteristic of the method.

	Very Satisfied	Satisfied	Neutral	Dissatisfied
1. The method easy to adapt to your work.				
2. The method can help you finish your work faster.				
3. The method can help your work easier.				
4. The result of this method can be used to make you work efficiently.				
5. This method can be run completely by use only a small time.				
6. This method can work without any error or problems.				
7. The method easily applies to test many kind of applications.				
8. The method easy to install and use on every machines.				
9. Overall, how satisfied are you with the SBTAR method?				

What are your roles or positions? And how do you involve in software development life cycle?

-----  
-----

Have you ever use any automation tools in your job?  Yes  No

If yes, what's the tool that you use? How satisfied with that tools?

-----  
-----

What are your recommendations?

-----  
-----

## VITA

In 2006, Mister Noppakit Nawalikit graduated in Computer Business and minoring in Marketing from Prince of Songkla University, Trang, Thailand

### Publication

Nawalikit, N., and Bhattarakosol, P., "SBTAR: An Enhancing Method for Automate Test Tools" Proceedings of 2009 International Conference on Computer Science and Software Engineering (WASET), August 26<sup>th</sup>-28<sup>th</sup>, 2009



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย