International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:8, 2009

SBTAR: An Enhancing Method for Automate Test Tools

Noppakit Nawalikit, and Pattarasinee Bhattarakosol

***Abstract*—**Since Software testing becomes an important part of Software development in order to improve the quality of software, many automation tools are created to help testing functionality of software. There are a few issues about usability of these tools, one is that the result log which is generated from tools contains useless information that the tester cannot use result log to communicate efficiently, or the result log needs to use a specific application to open. This paper introduces a new method, SBTAR that improves usability of automated test tools in a part of a result log. The practice will use the capability of tools named as IBM Rational Robot to create a customized function, the function would generate new format of a result log which contains useful information faster and easier to understand than using the original result log which was generated from the tools. This result log also increases flexibility by Microsoft Word or WordPad to make them readable.

***Keywords*—**Software Automation Testing, Automated test tool, IBM Rational Robot.

I. Introduction

SOFTWARE Testing Automation (STA) is one kind of the software testing which use a computer program to perform the test instead of manual checking. Several tools were created to help testers to improve the quality of the software. In addition, it also saves the development time and cost by enhancing the productivity of testers and entire development project teams. Using an automate tool, 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 HP, to ensure functionalities of the verified software. These tools can run scripts which are developed by themselves or written by languages they support [1].

Generally, the results which generated by automate tools cannot present details in an understandable format. Similarly, testers cannot use original log immediately because the log does not contain only usable information, but also useless information which testers must screen them out, and manually generate new reports. Therefore, testers cannot use original default log to show what exact steps are, or which data that cause problems. Another problem is that the result is 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; it needs to install more applications and licenses to access this log.

Authors are with Department of Mathematics, Faculty of Science, Chulalongkorn University Bangkok 10330, Thailand (e-mail: n_nostep027@hotmail.com, pattarasinee.b@chula.ac.th).

In this paper, Rational Robot from IBM is used as a tool to implement a new module. The new function creates a new format of the result log in the ".doc" format. The testers can customize information which would like them to be shown in the result log. When testers use a new function in automated testing, it can expel the useless information problems which came with original result log and testers can use this result log immediately.

The remainder of this paper is organized as follows. In Section II, it is talking about related works. Rational Robot, an automated test tools from IBM is introduced in Section III. In Section IV, we describe the new method and an algorithm of the new function which is implemented using Rational Robot. The discussions are presented in Section V. Finally, the conclusion is given in Section VI.

II. Related Works

Previous researchers have studied the effectiveness of automated testing tools to develop a test. Their works mostly mention about replaying tools to perform regression testing on applications, or the way to efficient the tools by proposing the way to increase reuse and flexibility so as to reduce the cost. However, the research do not mention about how to utilize ability of the tools to make most efficient result. For software automation test using tools to capture and playback do not mean the whole story of "Test Automation", 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.

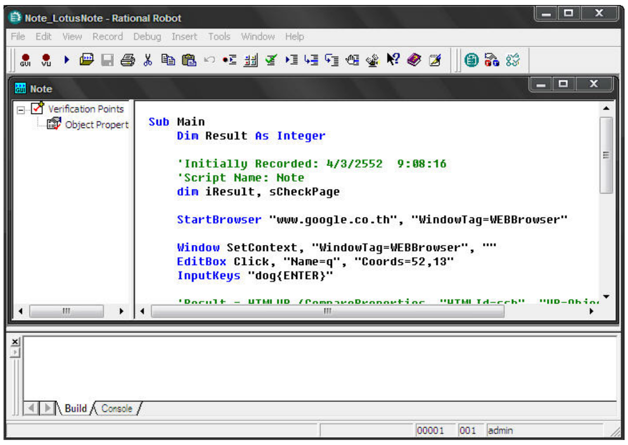
To use the Rational Robot as an automate tools to test the functionality of the software, testers have to record the script, perform an appropriate customization, then playback the script to obtain the result that comes as a picture of a test log which is generated by the Tools. Many problems will come with that test log, the biggest problem is that tester cannot use test log to communicate with the developer team efficiently because the details in the log which generated by tools may not have sufficient information. For example, in the log, generated by that of Rational tools, has only the name of scripts, date and times, a result (Pass, Fail or Warning). Information from this log may have some messages for descriptive, but developer still cannot know that what is the exact step, or data that is used in test which causes an error. The problem that will come after cannot effectively use the log which generate by the tools is Executive peoples or Project Management will not see the value of using tools to do software testing. Therefore, they will decline the use of these tools.

Therefore, the expectation from using this new method is that testers can perform an automate test easily and gain more effective works.

III. IBM Rational Robot

IBM Rational Robot is an application that used as Software automate testing tools which have capability to record activities that user perform, playback the recorded script and also provides an environment for editing script. Rational Robot can use for test Microsoft Windows client/server and Internet applications which running under Windows NT 4.0, Windows XP, Windows 2000 and Windows 98 environment. Rational Robot can extend ability by integrating with Rational TestManager to manage and playback a Rational Robot script and use it to manage test logs too. However, Rational Robot can be used as a standalone product.

In order to develop a test by Rational Robot, users can use Rational Robot to develop 2 kinds of scripts which are GUI scripts for functional testing, and sessions for performance testing. Rational Robot can automatically generate the scripts based on testers' actions performed in the application-under-test (AUT). After scripts were generated, testers can edit them using the Robot editor which provides color-coded commands for keywords as powerful integrated programming during scripts' development.



```
Sub Main
    Dim Result As Integer

    'Initially Recorded: 4/3/2552  9:08:16
    'Script Name: Note
    dim iResult, sCheckPage

    StartBrowser "www.google.co.th", "WindowTag=WEBBrowser"

    Window SetContext, "WindowTag=WEBBrowser", ""
    EditBox Click, "Name=q", "Coords=52,13"
    InputKeys "dog{ENTER}"
```

Fig. 1 IBM Rational Robot interface

Adding Features to Scripts

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

A. Library Source Files

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

- **.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 recordimg, sub-procedures and functions also can be added in to the script file.
- **.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 using Rational 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
'Script Name: Master

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"
 MsgBox iText

 End Sub
```

Example of Header File (Mastertheader.sbh)

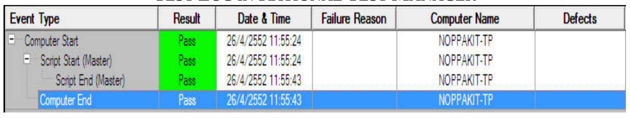
```
 Global Input1 as Integer
 Global Const Input2 as Integer =
10
 Declare Sub TestNumber Basic Lib
"MasterLibrary"(input1 as Integer,
input2 as Integer)
```

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 library source files can be called from procedures in other files (scripts and other library source files). 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.

Using Robot with Rational TestManager

Rational Robot can be integrated with other Rational tools to increase the capability of Automated testing, such as integrated with Rational Test Manager to manage all testing activities – planning, design, implementation, execution, and analysis. In the Test Manager, Testers can evaluate tests by examining the results of test execution in the test log, and by running various reports. Table 1 is the picture from Test Log in Rational Test Manager which generated after completed running the test.

TABLE I
TEST LOG IN RATIONAL TEST MANAGER



Event Type	Result	Date & Time	Failure Reason	Computer Name	Defects
Computer Start	Pass	26/4/2552 11:55:24		NOPPAKIT-TP	
Script Start (Master)	Pass	26/4/2552 11:55:24		NOPPAKIT-TP	
Script End (Master)	Pass	26/4/2552 11:55:43		NOPPAKIT-TP	
Computer End	Pass	26/4/2552 11:55:43		NOPPAKIT-TP	

Playing Back Scripts

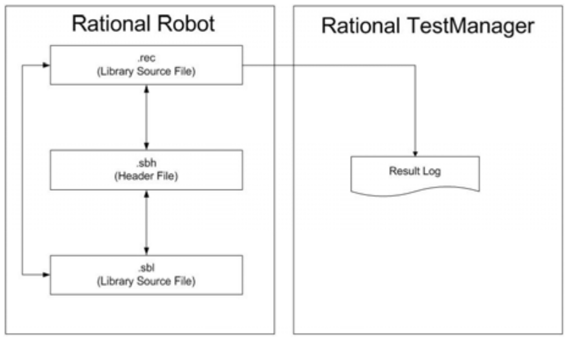


Fig. 2 Architecture of Rational Robot

When playing back a script, Rational Robot repeats the step from actions which performs while recording and automates 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 by manually testing. Thus, the testing time decrease, while both coverage and overall consistency increases.

Furthermore, the automate test can be enhanced by adding the custom procedure or function into Library file (.sbl). This method makes Rational Robot performs all tests much dynamically. But in the case that a customized procedure or function 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.

From the original result log, which generated from Rational Robot and Rational TestManager, may not be able to adapt to use for every place/situation. Fortunately, Rational Robot has capabilities which allow creating new functions and using another object program outside Rational Robot itself, such as Microsoft Word. The new module will using command to create a word document (.doc format) and wrote the information which specify in script put it in that word file to make a new format of result log. In next section will mention about the new module which create by use Rational Robot and how to use it to solve problem.

IV. Proposed Method

A new method will change the way of using the result log. The new report will be generated from mechanism which is implemented in the Library file. The new module will generate another file which has extension ".Doc" (Microsoft Word format), this type of file is very common which can be opened by the program name "WordPad", the tools that bundle with Microsoft OS, or by very famous office tools named "Microsoft Word". The architecture of the new method is show in Fig. 3.

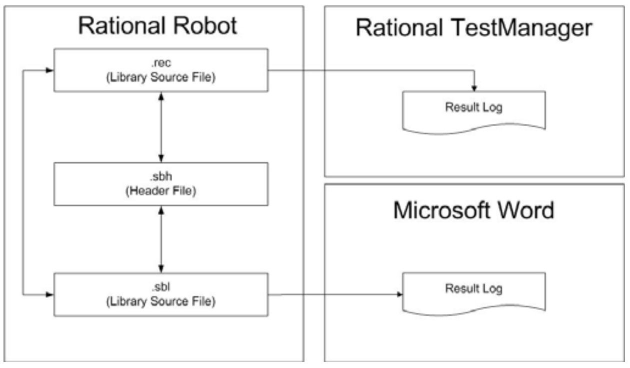
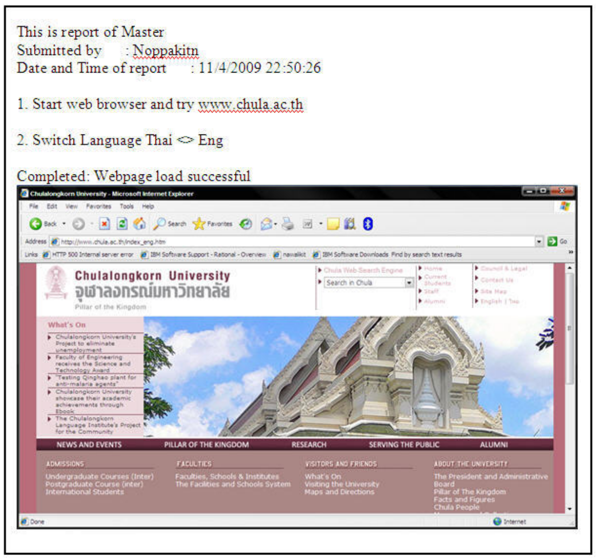


Fig 3 Architecture of New Practice adapt to Rational Robot

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

- name of script
- name of person who run the script
- date and time that running the script
- information of each test step
- test result (Pass or Fail) and description
- capture screen



This is report of Master
Submitted by : Noppakitn
Date and Time of report : 11/4/2009 22:50:26

1. Start web browser and try www.chula.ac.th

2. Switch Language Thai <> Eng

Completed: Webpage load successful

Fig. 4 Example of Result log from SBTAR function

From Fig. 4 is a result log created from new function, this result log generated from a small scenario because of example purpose then it contains only 2 test steps. All of this information can be distributed and easy for testers to understand and make a re-test manually. Additionally, developers can use this information to drive for locating causes and problems of the entire software processes.

A. Custom Function

The custom function starts with implementing a function in the library file (.sbl) because this function must be available for every script when needed. The name of this function which develops to serve this practice is "ScreenBasedTrackingARray" (SBTAR). The algorithm of SBTAR function is described as follows.

(1) Check output folder which using date as folder name if it is not available then create a new folder.
(2) Capture currently logon name of machine which is used for running script and keep it in variable.
(3) Create object of Microsoft Word application.
(4) Write information to Microsoft Word, start with name of script, username of person who run this script and current date and time.
(5) Loop writing each test step until last step.
(6) Print last screen by pushing "Prtsc" (PrintScreen) to keep the screen shot in clipboard and paste it at the end of file.

Once, implementation a new module in library file finished. It's the time for declaration in header file (.sbh). Fig. 5 shows how to declare function SBTAR in header file.

```
 Declare Function
ScreenBasedTrackingARray BasicLib
"MasterLib"        (sDocPath As
String,_
        sFileName As String,_
        iCount As Integer,_
        sErrMessage As String,_
        vValues() As Variant)
```

Fig. 5 Declaration statement of SBTAR function

After completing the implementation of the custom function in the library file and declaring the function in the Header File, the next step is to perform Script customization.

Script Customization

In the script file must be customized by specifying the value to variables and adding them to the script.

The first parameter which is "sDocPath" will be assigned the path or location that uses to store the log file. In Fig. 6, is an example of the value specification for "sDocPath".

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

Fig 6 An example of value specification for "sDocPath"

The second parameter is "iCount". It will be used to represent the sequence of test step then it needs to be reset its value to "0" every times. For resetting the value just assign the variable "iCount" = 0.

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

Fig. 7 Add description for each test step

After resetting "iCount" to 0 then we start to add description for each test step by assigning a sentence to keep in array variable name "vValue()". In this function iCount must adding 1 for every test step because it will using as indicator of array element. Last step put the call statement to use SBTAR function. To using this function users must use the Call Statement to call the function written in library file and must provide values for 5 parameters which are

1. "sDocPath" is parameter for keeping a value of location which is used to store the result log.
2. "sFileName" is parameter for keeping a value of name of the result log.
3. "iCount" is parameter for keeping a value of number of the testing steps that are used in a test.
4. "sErrMessage" is parameter for keeping a description/reason of the result.
5. "vValue()" is an array for keeping values of descriptions of each testing step.

Fig. 8 is call statement and required parameter for use SBTAR function.

```
Call
ScreenBasedTrackingARray(sDocPath,
sFileName, _
iCount, msgtext, vValue())
```

Fig 8 Example of call statement for SBTRA

V. Discussion

One important process in software development is the verification procedure where 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 safe their time and cost. Unfortunately, these tools cannot fully support the testing process according that 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 the defect of the software.

This paper 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 understanding 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.

VI. Conclusion

In order to obtain a 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 perform with a high skill of software tester group. 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 the log file. The common problem from most software test tools is that this common log file is difficult to understand and use. This paper proposes 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 benefits of applying this method are that cost and time in the testing procedure is reduced, with results are generated and used easily.

Acknowledgment

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.

References

[1] ZHU Xiaochun, ZHOU Bo, 1LI Juefeng, GAO Qiu, "A Test Automation Solution on GUI Functional Test", Eighth ACIS International Conference on Volume 3, Issue, July 30 2007-Aug. 1 2007 Page(s):1124 – 1128.

[2] IBM Rational Robot, http://www-01.ibm.com/software/awdtools/tester/robot/support/index.html

[3] Rational User Group, http://tech.groups.yahoo.com/group/RationalUsers/

[4] Karhu, Katja; Repo, Tiina; Taipale, Ossi; Smolander, Kari, "Empirical Observations on Software Testing Automation", Software Testing Verification and Validation, 2009. ICST apos;09. International Conference on Volume , Issue , 1-4 April 2009 Page(s):201 – 209

[5] Michael, J.B.; Bossuyt, B.J.; Snyder, B.B., "Metrics for measuring the effectiveness of software-testing tools", Software Reliability Engineering, 2002. ISSRE 2002. Proceedings. 13th International Symposium on Volume , Issue , 2002 Page(s): 117 – 128

[6] Cheng-hui Huang; Huo Yan Chen, "A Tool to Support Automated Testing for Web Application Scenario", Systems, Man and Cybernetics, 2006. SMC apos;06. IEEE International Conference on Volume 3, Issue , 8-11 Oct. 2006 Page(s):2179 – 2184

[7] Bering, C.A.; Covey, J.H., "Software testing-concepts and approach", Aerospace and Electronics Conference, 1991. NAECON 1991., Proceedings of the IEEE 1991 National Volume , Issue , 20-24 May 1991 Page(s):750 - 756 vol.2

[8] O. Taipale, K. Smolander, and H. Kälviäinen, "Cost Reduction and Quality Improvement in Software Testing," in Software Quality Management Conference, Southampton, UK, 2006.

[9] E. Dustin, J. Rashka, and J. Paul, Automated software testing: introduction, management, and performance. Boston: Addison-Wesley, 1999.

[10] Mark Fewster, "Common Mistakes in Test Automation," Grove Consultants 2001.