REVIEW of AUTOMATED SOFTWARE TESTING with MACRO SCHEDULER

Priyanka Mittal
ASET Department, AMITY University
Noida, UP, INDIA
pri_mittal@rediffmail.com

Rajni Sehgal
ASET Department, AMITY University
Noida, UP, INDIA
rsehgal@amity.edu

Abstract— **Software testing is the most crucial and expensive task of project development. Testing is done in each development phase and can be done in hardware or software. The testing phase took over the 50% of the project resources. In this paper, we focus on automation testing using macro scheduler. Automated testing streamlines the QA process and saves precious resources, Time and money and improves the effectiveness of software testing. Macro Scheduler excels at testing software through its high level functions to test a program. The time saved by running test scripts over manual checklists is enormous and bugs can be found more quickly.**

Keywords— Software Testing, Test Case generation, Automated Testing, Macro Scheduler, Test Scripts**.**

I. INTRODUCTION

Software testing is one of the main task in project development .It can be a time consuming task. Traditionally, the applications are verified manually going through checklist to verify that the requirements of the software was met correctly through QA technicians and/or programmers. To save considerable time and resources, the Automated Testing using Macro Scheduler is used. Macro scheduler had provided many innovative new features, such as screen image recognition technology, to ensure our products can automate more software, more easily. Microsoft Visual Basic Scripting Edition can be incorporates by macro scheduler. Macro Scheduler macros can contain both Macro Script and VBScript to provide infinite possibilities.

II. PROCESS OF MACRO SCHEDULER

A. What is macro scheduler?

Macro Scheduler is ideal for automated software testing and costs a fraction of the price of most dedicated SQA tools. Macro Scheduler has been used by organizations large and small for enterprise-wide automated application testing. Macro Scheduler's powerful, yet easy to use, language allows for event driven user simulation to drive applications of any kind and gather time critical data. By working at the user level Macro Scheduler bridges that gap, allows automation of disparate systems and gathers more realistic data as a result of user level simulation rather than depending on internal hooks.

B. Procedure to create macro in excel

1. Right click on any sheet tab and choose *View Code*, to open the Visual Basic Editor.
2. In the Project Explorer at the left of the screen, find the workbook.
3. Look for a *Modules* folder, and open it. (If there is no Modules folder, go to Step 6.)
4. for each module in the folder:
i) Right-click on the module name.
ii) Choose *Remove Module1* (the name of your module may be different)
iii) Click *No* when asked if you want to Export.
5. Open the *Microsoft Excel Objects* folder.
6. for each worksheet, and for *This Workbook*:
i) Double-click on the object name, to open its code module. In this sample, you'd double-click on *Sheet1 (Sheet1)*
ii) On the keyboard, press Ctrl+A to select all the code (even if the code module looks empty)
iii) Press the Delete key.
7. Look for a *Forms* folder, and open it.
8. Delete any User Forms that it contains.
9. Look for a *Class Modules* folder, and open it.
10. Delete any class modules that it contains.
11. Close the Visual Basic Editor.
12. Save the changes to the workbook.

Buttons to Run Excel Macros

When a button is drawn onto a sheet the assign macro is not displayed. When right-clicking on the button the "Assign Macro" context menu item is not present. There are buttons from the Forms toolbar and there are buttons from the Control

Toolbox. If "Assign Macro" is not an option then it's from the Control Toolbox.

Choose "View code" and call your macro from it like this:

```
Private Sub CommandButton1_Click()
  Call Macro1
End Sub
```

C. Benefits

1. It Automate and Streamline Software Processes.
2. It Save Time, Solve Problems, Simplify Work.
3. It Cut Development Costs.
4. Improve Reliability and Accuracy.
5. Huge Return on Investment.

D. Features

1. It has Comprehensive, Reliable Automation for any Software or Business Process.
2. Keyboard & Mouse Macro Recorder with Advanced Window Sensing Technology.
3. Easily Create Scripts in Minutes.
4. Convert Macros to EXE files for distribution on other PCs.
5. Unique See Screen Image Recognition functions for Automation of any process.
6. Flexible Scheduler & Unique Auto Logon Technology For Unattended Operation.
7. Over 250 Powerful Built-in Script Commands, Loops, Conditionals etc, Plus Microsoft VBScript.

E. Role of Macro Scheduler

In this paper, we have used macro scheduler procedure to automate our testing process. In this way we have switch from manual testing to automated testing. Use of macro Scheduler have improved the time consumed, bug testing increase the no of test cases performed.

III.MANUAL TESTING V/S AUTOMATION TESTING

A. Testing

Software testing can define as the process of validating and verifying that a software program/application/product:

1. meets the business and technical requirements that guided its design and development;

2. Works as expected; and

3. Can be implemented with the same characteristics.

B. Manual Testing

It is a testing technique where the software is tested manually by test engineer, who prepare all the test cases and executes manually step by step on the application, and indicates whether a particular step was accomplished successfully or whether it failed, performs manual testing,. At the time of manual testing tester only need test case and the related information, for the execution of desired test case. While testing our application, complete test provides an opportunity to manually create and manage tests. In a project, when manual testing item is added, a collection of steps with detailed information and description can be created to perform the desired application when being tested. According to test strategy of test plan, test case is also written for all type of testing. In software, on the basis of design document, test engineer can write test case. Any testing includes manual testing. In the initial phase of software development, when the interaction between software and its user are not stable enough, it is useful.

TABLE III
MANUAL TESTING V/S AUTOMATED TESTING

S.no	Manual testing	Automated testing
1.	Suitable for little repetition such as exploratory testing or late development verification testing.	Suitable for several repetitions.
2.	Manual testing is not reusable.	Automated testing is completely reusable.
3.	Provides limited visibility.	Provides global visibility.
4.	Ends up being an integration test.	One can test unit, system and module.
5.	It is covered in limited cost.	It is costly then manual.
6.	It takes more time to cover all cases.	Easy to cover up all cases in a limited time period.
7.	It is boring and same for whole period.	New things can learn and it is interesting.

C. Automated Testing

It is a technique, where the script can be run on any testing tool by test engineer. For new test engineer it is not an easy process to test the software using script in automated tools. So for writing a good script against any test case, the engineer should have a good programming knowledge. The plan is followed by these people to make various scripts for various testing. It is

reusable, once a test case is created then it can be use for future reference. It has a good feature of Email Notifications, on failure or threshold levels there is an automated notification. This may be the test runner or tooling that executes it. It Support unattended test runs for integration with build processes and batch runs. Continuous Integration servers require this, support distributed execution environment and distributed application support.

D. Coding Process with Manual Tests

i) Write code
ii) Uploading the code to some place
iii) Build it
iv) Running the code manually (in many cases filling up forms etc step by step)
v) Check Log files, Database, External Services, Values of variable names, Output on the screen etc
vi) If it does not work, repeat the above process

E. .Coding Process with Automated Unit Tests

i) Write code to pass the tests
ii) Auto-compile and run
iii) If tests fail -> make appropriate modifications
iv) If tests pass -> repeat for next method

F. Coding Process with Automated Functional Tests

i) Write one or more test cases
ii) Auto-compile and run to see the tests fail
iii) Finish writing code (with all unit tests passing)
iv) Write a Functional Test using any tool
v) Auto-compile and run
vi) If tests fail -> make appropriate modifications
vii) If tests pass -> move ahead

IV. CONCLUSION

Automated testing streamlines the QA process and saves precious resources, time and money. As the project continues along it s life cycle it is easy to run previously created test scripts and add new test cases. The time saved by running test scripts over manual checklists is enormous and bugs can be found more quickly. Test scenarios can easily be run repeatedly for added confidence in your product. Developers, QA engineers and customers all benefit. With Macro Scheduler s powerful GUI automation routines, output functions, VBScript capability and complex expressions it is easy to build advanced test scenarios for all application.

[1] Mike Andrews, James A. Whittaker, How to Break Web Software: Functional and Security Testing of Web Applications and Web Services, Addison Wesley.

[2] The Web Testing Companion: Insider's Guide to Efficient and Effective Tests, Wiley, 2003.

[3] Adam Barr, Find the Bug: A Book of Incorrect Programs, Addison-Wesley Professional, 2004.

[4] Beck, Kent, Test-Driven Development: By Example, Addison-Wesley, Boston, MA, 2003.

[5] Boris Beizer, Black Box Testing: Techniques for Functional Testing of Software and Systems, John Wiley & Sons, Inc., New York, 1995.

[6] Robert V. Binder, Testing Object-Oriented Systems: Models, Patterns and Tools, the Addison-Wesley Object Technology Series, Addison-Wesley Professional, 1999.

[7] Rex Black, Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing, 2nd Edition, Wiley, 2002.

[8] Rex Black, Critical Testing Processes: Plan Prepare, Perform, Perfect, Addison-Wesley Professional.

[9] Ilene Burnstein, Practical Software Testing, Springer, 2003.

[10] Lee Copeland, A Practitioner's Guide to Software Test Design, Artech House Publishers, Boston, MA, 2004.

[11] Rick D. Craig, Stefan P. Jaskiel, Systematic Software Testing, Artech House Publishing, Norwood, MA, 2002.

[12] Crispin, Lisa, Tip House, Testing Extreme Programming, Addison-Wesley, Boston, MA, 2003.

[13] Robert Culbertson, Chris Brown, Gary Cobb, Rapid Testing, Software Quality Institute Series, Prentice Hall PRT, Upper Saddle River, NJ, 2002.

[14] Aristides Dasso, Ana Funes, Verification, Validation and Testing in Software Engineering, Idea Group Publishing, 2006.

[15] Rodger D. Drabick, Best Practices for the Formal Software Testing Process: A Menu of Testing Tasks, Dorset House Publishing Company, 2003.

[16] Paul Hamill, Unit Test Frameworks, O'Reilly Media, Inc, 2004.

[17] Paul C. Jorgensen, Software Testing: A Craftsman's Approach, 2nd Edition, CRC, 2002.