International Journal of Latest Research in Science and Technology **ISSN (Online):2278-5299**
Vol.1,Issue 2 :Page No.152-154 ,July-August(2012)
http://www.mnkjournals.com/ijlrst.htm

An Automation of Software Testing: A Foundation for the Future

Ravinder Veer Hooda
Assistant Professor (CSE Deptt.), Emax Institute of Engg. & Technology, Ambala (Haryana), India
ravi4vh@gmail.com

Abstract - Test automation is the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.[1] Commonly, test automation involves automating a manual process already in place that uses a formalized testing process.

I. INTRODUCTION

Although manual tests may find many defects in a software application, it is a laborious and time consuming process. In addition, it may not be effective in finding certain classes of defects. Test automation is the process of writing a computer program to do testing that would otherwise need to be done manually. Once tests have been automated, they can be run quickly and repeatedly. This is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can cause features to break which were working at an earlier point in time.

There are two general approaches to test automation:

• Code-driven testing. The public (usually) interfaces to classes, modules or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.

• Graphical user interface testing. A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behavior of the program is correct.

Test automation tools can be expensive, and are usually employed in combination with manual testing. Test automation can be made cost-effective in the long term, especially when used repeatedly in regression testing.

In Automation Testing the test Engineer or Software Quality Assurance person should have coding knowledge as they have to write down the test cases in form of code which when run and give output according to checkpoint inserted in it. Checkpoint is the point which is inserted to check any scenario.

One way to generate test cases automatically is model-based testing through use of a model of the system for test case generation but research continues into a variety of alternative methodologies for doing so.

What to automate, when to automate, or even whether one really needs automation are crucial decisions which the testing (or development) team must make. Selecting the correct features of the product for automation largely determines the success of the automation. Automating unstable features or features that are undergoing changes should be avoided.[2]

II. CODE-DRIVEN TESTING

A growing trend in software development is the use of testing frameworks such as the xUnit frameworks (for example, JUnit and NUnit) that allow the execution of unit tests to determine whether various sections of the code are acting as expected under various circumstances. Test cases describe tests that need to be run on the program to verify that the program runs as expected.

Code driven test automation is a key feature of agile software development, where it is known as test-driven development (TDD). Unit tests are written to define the functionality *before* the code is written. Only when all tests pass is the code considered complete. Proponents argue that it produces software that is both more reliable and less costly than code that is tested by manual exploration. It is considered more reliable because the code coverage is better, and because it is run constantly during development rather than once at the end of a waterfall development cycle. The developer discovers defects immediately upon making a change, when it is least expensive to fix. Finally, code refactoring is safer; transforming the code into a simpler form with less code duplication, but equivalent behavior, is much less likely to introduce new defects.

III. GRAPHICAL USER INTERFACE (GUI) TESTING

Many test automation tools provide record and playback features that allow users to interactively record user actions and replay them back any number of times, comparing actual results to those expected. The advantage of this approach is that it requires little or no software development. This approach can be applied to any application that has a graphical user interface. However, reliance on these features poses major reliability and maintainability problems. Relabelling a button or moving it to another part of the window may require the test to be re-recorded. Record and playback also often adds irrelevant activities or incorrectly records some activities.

A variation on this type of tool is for testing of web sites. Here, the "interface" is the web page. This type of tool also requires little or no software development. However, such a framework utilizes entirely different techniques because it is reading HTML instead of observing window events.

Another variation is scriptless test automation that does not use record and playback, but instead builds a model of the Application Under Test (AUT) and then enables the tester to create test cases by simply editing in test parameters and conditions. This requires no scripting skills, but has all the power and flexibility of a scripted approach. Test-case maintenance seems to be easy, as there is no code to maintain and as the AUT changes the software objects can simply be re-learned or added. It can be applied to any GUI-based software application. The problem is the model of the AUT is actually implemented using test scripts, which have to be constantly maintained whenever there's change to the AUT.

IV. WHAT TO TEST

Testing tools can help automate tasks such as product installation, test data creation, GUI interaction, problem detection (consider parsing or polling agents equipped with oracles), defect logging, etc., without necessarily automating tests in an end-to-end fashion.

One must keep satisfying popular requirements when thinking of test automation:

- Platform and OS independence
- Data driven capability (Input Data, Output Data, Metadata)
- Customizable Reporting (DB Access, crystal reports)
- Easy debugging and logging
- Version control friendly – minimal binary files
- Extensible & Customizable (Open APIs to be able to integrate with other tools)
- Common Driver (For example, in the Java development ecosystem, that means Ant or Maven and the popular IDEs). This enables tests to integrate with the developers' workflows.
- Support unattended test runs for integration with build processes and batch runs. Continuous integration servers require this.
- Email Notifications (automated notification on failure or threshold levels). This may be the test runner or tooling that executes it.
- Support distributed execution environment (distributed test bed)
- Distributed application support (distributed SUT)

V. DEFINING BOUNDARIES BETWEEN AUTOMATION FRAMEWORK AND A TESTING TOOL

Tools are specifically designed to target some particular test environment. Such as: Windows automation tool, web automation tool etc. It serves as driving agent for an automation process. However, automation framework is not a tool to perform some specific task, but is an infrastructure that provides the solution where different tools can plug itself and do their job in a unified manner. Hence providing a common platform to the automation engineer doing their job.

There are various types of frameworks. They are categorized on the basis of the automation component they leverage. These are:

1. Data-driven testing
2. Modularity-driven testing
3. Keyword-driven testing
4. Hybrid testing
5. Model-based testing

VI. NOTABLE TEST AUTOMATION TOOLS

Tool name	Produced by	Latest version
HP QuickTest Professional	HP Software Division	11.0
HTTP Test Tool	Open source	2.0.8
IBM Rational Functional Tester	IBM Rational	8.2.1
LabVIEW	National Instruments	2011
Maveryx	Maveryx	1.2.0
Oracle Application Testing Suite	Oracle Corporation	12.1
QF-Test	Quality First Software GmbH	3.4.7
Ranorex	Ranorex GmbH	3.2.3
Rational robot	IBM Rational	2003
Selenium	Open source	2.20
SilkTest	Micro Focus	2011
eggPlant	TestPlant	2012
EiffelStudio AutoTest	Eiffel Software	7.0
SOAtest	Parasoft	9.2
TestComplete	SmartBear Software	8.7
Testing Anywhere	Automation Anywhere	7.5
TestPartner	Micro Focus	6.3
TPT	PikeTec GmbH	4.0
TOSCA Testsuite	TRICENTIS Technology & Consulting	7.3.1
Visual Studio Test Professional	Microsoft	2010
WATIR	Open source	3.0
Telerik Test Studio	Telerik, Inc.	2012.1

VI. CONCLUSION

In the broader sense, however, the future of software testing is strongly linked to automated testing, and how we deal with that. Development is changing and getting more streamlined, and is therefore getting done much faster. Testing will rely on automation to help keep up. If we don't grow the discipline of test automation, however, it won't be able to keep up.

REFERENCES

1 Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Management* (http:/ / www. wiley. com/ WileyCDA/ WileyTitle/ productCd-0470042125. html). Wiley-IEEE Computer Society Press. p. 74. ISBN 0-470-04212-5.

2 Brian Marick. "When Should a Test Be Automated?" (http:/ / www. stickyminds.com/sitewide.asp?Function=edetail& ObjectType=ART& ObjectId=2010). StickyMinds.com. . Retrieved 2009-08-20.

3 http:/ / www. tricentis. com/ en/ newsroom/ newsletter/ 201108-News

4 Elfriede Dustin, et al.: *Automated Software Testing.* Addison Wesley, 1999, ISBN 0-201-43287-0

5 Elfriede Dustin, et al.: *Implementing Automated Software Testing.* Addison Wesley, ISBN 978-0-321-58051-1

6 Mark Fewster & Dorothy Graham (1999). *Software Test Automation*. ACM Press/Addison-Wesley.
ISBN 978-0-201-33140-0.

7 Roman Savenkov: *How to Become a Software Tester.* Roman Savenkov Consulting, 2008, ISBN 978-0-615-23372-7

8 Hong Zhu et al. (2008). *AST '08: Proceedings of the 3rd International Workshop on Automation of Software Test* (http:/ / portal. acm. org/ citation. cfm?id=1370042#). ACM Press. ISBN 978-1-60558-030-2.

9 Mosley, Daniel J.; Posey, Bruce. *Just Enough Software Test Automation* (http:/ / www. amazon. com/ Just-Enough-Software-Test-Automation/ dp/ 0130084689/ ref=sr_1_5?s=books& ie=UTF8& qid=1337627825& sr=1-5). ISBN 0130084689.

10 Practical Experience in Automated Testing (http:/ / www. methodsandtools. com/ archive/ archive. php?id=33)

11 Automation Myths (http:/ / www. benchmarkqa. com/ pdf/ papers_automation_myths. pdf) by M. N. Alam 3_TargetLink_EmbeddedTester_en_701. pdf)

12 Test Automation: Delivering Business Value (http:/ / www. applabs. com/ internal/app_whitepaper_test_automation_delivering_business_value_1v00. pdf)

13 Test Automation Snake Oil (http:/ / www. satisfice. com/ articles/ test_automation_snake_oil. pdf) by James Bach

14 When Should a Test Be Automated? (http:/ / www. stickyminds. com/ r. asp?F=DART_2010) by Brian Marick

15 Why Automation Projects Fail (http:/ / martproservice. com/ Why_Software_Projects_Fail. pdf) by Art Beall

16 Guidelines for Test Automation framework (http:/ / info. allianceglobalservices. com/ Portals/ 30827/ docs/ test automation framework and guidelines. pdf)

17 Advanced Test Automation (http:/ / www. testars. com/ docs/ 5GTA. pdf)