Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
REVIEW ARTICLE
Available Online at www.ijarcs.info

A Study on the Current Trends in Software Testing Tools

Siddharth Bhargava, Sahil Guleria, Ajita and Gaurang
School of Computer Science and Engineering
VIT University, Vellore, India

Abstract: This paper describes the recent developments that have been seen in the software testing tools, in the past two decades. With the current rapid growth observed in technology, hardware and software likewise, it has become imperative that software testing tools become equally dynamic, scalable and portable in nature, just like the product it is meant to test. The study investigates the various new and emerging trends which have been introduced in the field of testing and what is the general approach and choice of the developer or the tester while selecting a tool for a particular application.

Keywords: Software testing, Testing tools, dynamic, scalable and portable, selection of tool.

1. INTRODUCTION

Software testing forms an integral part of any Software Development Life Cycle. It focuses on verifying and validating a software process, product or the project itself. Based on a set of metrics, the tests can be used to determine various attributes of the software while at the same time, help pinpoint errors or bugs present in the software. Software testing helps develop a robust, fully functional and stable product.

1.1. Testing Strategies

Testing is a set of activities that can be preplanned and executed in a systematic manner. Test strategies help design a proper testing template, which consists of a set of steps into which the user can place specific test case design techniques and testing methods. Testing strategies have some generic characteristics which are always followed, conduct regular, effective technical reviews; begin at component level and works outward toward integration of entire system; different testing techniques suitable for different software engineering approaches; conducted by developers and an independent test group.

1.2. Selection of Testing Tools

With the wide variety in the testing strategies as well as the testing methodology that can be adapted for a given software, many different testing tools have been developed to fulfil the purpose of performing the necessary tests. As per Kaner, Falk, Nguyen, [15] a good testing tool is one which has a high probability of finding an error, is not redundant, should be independent of other tests and create no side effects in the system.

Constance Heitmeyer [6] in his paper describes testing tools as a medium to help validate a specification, provide mechanized support for verifying properties, reduce the time and e ort required to construct (and execute) a set of test cases. Tools can help find errors that human inspection often misses. A set of powerful tools can liberate people from doing the hard intellectual work required to produce high quality, high assurance software systems. In their paper, Tanja, Beatriz et al. [19] have suggested a methodological framework which can be used to evaluate the software testing techniques and ensure that they follow the desired guidelines and checklists. Thus, Testing the software not only requires a planned approach but also appropriate tools. The choice of the tools depends on the testing methods used for the given product, application or service it may provide. Khaled, Rafa et al. [14] have classified the tools as per the software testing methods. This helps in characterizing the tests which have limited automated tools. Thus, selection of tools depends on multiple factors and requires a proper analysis of the product and the software it uses.

2. CURRENT TRENDS

Technology dynamics show that modern technology is developing at an unprecedented rate. Changes in software, hardware and the interfacing have brought about evolution in the testing tools as well. Many modern testing tools have been integrated to become compatible with latest technology as well as the present-day world. Innovations in the field of code optimization, resource allocation and interfacing, has resulted in growth of the current testing tools into more user friendly, customizable and easy to use utilities. Variation introduced in the tools have resulted in broader applicability and more testing capability of the tool, itself.

2.1. Visualization

One major new development to the field of testing tools is the use of visualization to depict the output of the tests. In her paper, Sita Ramakrishnan [16] uses a visual tool, LIGHTVIEWS, to describe Object Oriented testing case studies using visual images, animation and interactive outputs. While visualizing the results may be an excellent method of displaying the test outputs, visualization or data representation can also be used to influence the input fed into the test and help increase the efficiency of the testing tools, as seen by Yu Xia Sun et al. [17] where they used XML representation of the source codes and tested them. Another reference is made by Benjamin Kormann, Dmitry Tikhonov, Birgit Vogel-Heuser [9] where they have used UML Sequence diagrams for testing out the Programmable Logic Controller (PLC). This has helped reduce the complexity of the software components.

2.2. Cost Optimization and Resource Management

Optimization of the code and as well as efficient resource allocation and management has resulted in making low-cost testing tools which are more efficient and faster in computation and testing capacity. Dimitris Gizopoulos [4]

investigated, in his paper, how software-based self-testing can be used to produce low-cost testing of the embedded processor cores. Using software architecture-based regression testing, it is possible help reduce testing cost and access both high-level and low-level testability, as studied by Henry Muccini and others [13]. Another cost effective technique is using virtualization for setting up software test environments. In the paper by Youngjoo Woo, Seon Yeong Park and Euiseong Seo [21], they proposed using a virtual battery as a test subject and subjecting it to discharge of the virtual machines, as per the consumption of their resources. A new approach used in developing and testing Artificial Intelligence and expert systems have been proposed by Thair Mahmoud and Bestoun Ahmed [12], where they use fuzzy logic-based adaptive swarm optimization for the purpose of software testing.

2.3. Open Sourcing and Multi-Platform Integration

Open sourcing of testing tools has helped promote development of customizable tools which may be highly specific in nature or generalized in order to test multiple formats, on various platforms. For instance, TestTalk proposed in Chang Liu's paper [11], is capable of generating test descriptions which are platform-independent and tool-neutral. In the research conducted by Seyed Amir Emami et al. [2], it has been studied that open source testing tools are relatively cheaper and easily available for popular programming platforms. It states that there is active participation from the open source community in actively updating and maintaining the tools. These tools serve as an excellent alternative to their commercially licensed counterparts. An open source tool, AUSTIN, has been proposed as an alternative to ETF, the state-of-the-art test used for software testing of C programs [10].

Another example of implementing open source tools in an innovative way is through data-driven integration as explained by Wenming, Xiangling and Jianmei [20], where four open source testing tools were integrated and used to create a single complete test solution which potentially covers the entire test area. It also helps standardizes the testing process.

2.4. Automated and Intelligent Testing

With better understanding of the working of the testing tools, automated testing tools have become very popular among industries who need very precise tests capable of handling large volumes of data and functions. In the paper by Eugenia Diaz et al. [3], they proposed a tool which can automatically generate test cases in order to obtain branch coverage from the source code. This effectively reduces software testing time as compared to manual instrumentation. According to Chorng-Shiuh Koong et al. [8], automated testing functions can be developed to reduce the burden and increase the efficiency of the engineer.

An implementation of automated testing includes an automated black box testing tool for a parallel programming library as mentioned by Roy Patrick Tan et al. [18]. The tool enables to not only generate tests, but also perform the performance, stress and security tests as well. This has helped achieve high code coverage and gain a high confidence in the quality of the code. The test's framework can also be adapted for a number of other kind of tests. Another dynamic testing technique involves automated software testing using agents [1]. Xin Guo et al. [5] have proposed an intelligent multilingual software testing tool which can help software testers to test the product that is not in the tester's native language. The tool also includes an output verifier that helps to verify the applications output generated (by the test case) and matches it with the expected output.

Some other different techniques that have been developed as a modification to the current testing tools include Chin-Yu Huang's work [7] which suggests that software fault probability depends on the skill of test teams, program size, and software testability. Project managers should buy new automated test tool, technology or additional manpower which can provide a significant improvement in software testing and productivity. Another author, Shiyi Xu [22], has suggested use of fault dominance as a method to reduce faults in software testing and enhance effectiveness of the testing.

3. CONCLUSION

The study has shown some of the various techniques and innovations that have been introduced in the field of software testing tools, over the past two decades. New and modernized tools have come up with the ability to be scalable, portable and executable in various operating environments. The current trends in the testing tools prove that the need for the technology is of relevance and there would be continuous development being made in the field.

REFERENCES

[1] P. Dhavachelvan, G.V. Uma, and V.S.K. Venkatachalapathy. A new approach in development of distributed framework for automated software testing using agents. Knowledge-Based Systems Volume 19 Issue 4, Elsevier, 2006.

[2] Seyed Amir Emami, Jason Chin Lung Sim, and Kwan Yong Sim. A survey on open source software testing tools: A preliminary study in 2011. Fourth International Conference on Machine Vision (ICMV 2011), 2011.

[3] Raquel Blanco Eugenia Daz, Javier Tuya. A modular tool for automated coverage in software testing. Software Technology and Engineering Practice, Eleventh Annual International Workshop, 2003.

[4] Dimitris Gizopoulos. Low-cost, on-line self-testing of processor cores based on embedded software routines. Microelectronics journal, 2004 - Elsevier, 2004.

[5] Xin Guo, William Tay, Tong Sun, and Rodrigo Andres Urra. Intelligent multilingual software testing tool. Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference, 2008.

[6] Constance Heitmeyer. Managing complexity in software development with formally based tools. Electronic Notes in Theoretical Computer Science, Elsevier, 2004.

[7] Chin-Yu Huang. Performance analysis of software reliability growth models with testing-e ort and change-point. Journal of Systems and Software, 2005 - Elsevier, 2005.

[8] Chorng-Shiuh Koong, Chihhsiong Shih, Pao-Ann Hsiung, Hung-Jui Lai, Chih-Hung Chang, William C. Chu, Nien-Lin Hsueh, and Chao-Tung Yang. Automatic testing environment for multi-core embedded softwareatemes. Journal of Systems and Software, Volume 85 Issue 1, Elsevier, pages Pages 43{60, 2012.

[9] Benjamin Kormann, Dmitry Tikhonov, and Birgit Vogel-Heuser. Automated plc software testing using adapted UML sequence diagrams. IFAC Proceedings Volume, Elsevier, 2012.

[10] Kiran Lakhotia, Mark Harman, and Hamilton Gross. Austin: An open source tool for search based software testing of c programs. Information and Software Technology, 2013 - Elsevier, 2013.
[11] Chang Liu. Platform-independent and tool-neutral test descriptions for automated software testing. ICSE '00 Proceedings of the 22nd international conference on Software engineering, pages 713{715, 2000.
[12] Thair Mahmoud and Bestoun S. Ahmed. An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing. Expert Systems with Applications, 2015 - Elsevier, 2015.
[13] Henry Muccini, Marcio Dias, and Debra J. Richardson. Software architecture-based regression testing. Journal of Systems and Software, 2006 - Elsevier, 2006.
[14] Khaled M. Mustafa, Rafa E. Al-Qutaish, and Mohammad I. Muhairat. Classification of Software Testing Tools Based on the Software Testing Methods. Second International Conference on Computer and Electrical Engineering, 2009.
[15] H. Q. Nguyen, Kaner C., and J. Falk. Testing Computer Software (2d. ed). 1993.
[16] Sita Ramakrishnan. Lightviews - visual interactive internet environment for learning oo software testing. ICSE, 2000.
[17] Yu Xia Sun and Huo Yan Chen T.H. Tse. Lean implementations of software testing tools using xml representations of source codes. International Conference on Computer Science and Software Engineering, 2008.
[18] Roy Patrick Tan, Pooja Nagpal, and Shaun Miller. An automated black box testing tool for a parallel programming library. Software Testing Verification and Validation, 2009. ICST '09. IEEE International Conference, 2009.
[19] Tanja E.J. Vos, Beatriz Marn, Maria Jose Escalona, and Alessandro Marchetto. A Methodological Framework for Evaluating Software Testing Techniques and Tools. 12th International Conference on Quality Software, 2012.
[20] Guo Wenming, Fu Xiangling, and Feng Jianmei. A data-driven software testing tools integration system. Computational Intelligence and Software Engineering (CiSE), International Conference, 2010.
[21] Youngjoo Woo, Seon Yeong Park, and Euiseong Seo. Virtual battery: A testing tool for power-aware software. Journal of Systems Architecture, 2013 - Elsevier, 2013.
[22] Shiyi Xu. Reduction of faults in software testing by fault domination. Tsinghua Science & Technology, 2007 - Elsevier, 2007.

