Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software Testing and Its Tools

Priyanka
Software engineering Department,
Bits Bhiwani, India

Lect. Vipin Arora
Computer science Department
BITS Bhiwani, India

Abstract-Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we can not completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade –off between budget, time and quality.

Fault based testing strategies test software by focusing on specific, common types of faults. The coupling effect hypothesizes that test data sets that detect simple types of faults are sensitive enough to detect more complex types of faults. This paper describes empirical investigations into the coupling effect over a specific class of software faults. All of the results from this investigation support the validity of the coupling effect. The major conclusion from this investigation is the fact that by explicitly testing for simple faults, we are also implicitly testing for more complicated faults, giving us confidence that fault-based testing is an effective way to test software.

Keywords-software testing ,software testing tools ,testing tools, testing and its tools.

I. INTRODUCTION

Introduction-Because of the fallibility of its human designers and its own abstract, complex nature, software development must be accompanied by quality assurance activities. It is not unusual for developers to spend 40% of the total project time on testing. For life-critical software (e.g. flight control, reactor monitoring), testing can cost 3 to 5 times as much as all other activities combined. The destructive nature of testing requires that the developer discard preconceived notions of the correctness of his/her developed software.

Software Testing can be defined as: Testing is an activity that helps in finding out bugs/defects/errors in a software system under development, in order to provide a bug free and reliable system/solution to the customer.

II. SOFTWARE TESTING FUNDAMENTALS

Testing objectives include :

1. Testing is a process of executing a program with the intent of finding an *error*.
2. A good test case is one that has a high probability of finding an as yet undiscovered error.
3. A successful test is one that uncovers an as yet undiscovered error.

Testing should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. A secondary benefit of testing is that it demonstrates that the software appears to be working as stated in the specifications. The data collected through testing can also provide an indication of the software's reliability and quality. But, testing cannot show the absence of defect -- it can only show that software defects are present.

III. SOFTWARE TESTING TOOLS

1.1 Unit Test Tools:

These tools, frameworks, and libraries help to automate unit test execution, which is usually performed by the developer, usually using interfaces below the public interfaces of the software under test.

1. Rational Test RealTime Unit Testing:

Kind of Tool:

Rational Test RealTime's Unit Testing feature automates C, C++, Ada 83 and 95 software component testing.

Organization:

IBM Rational Software
http://www-306.ibm.com/software/awdtools/test/realtime/

Software Description:

Rational Test RealTime Unit Testing performs black-box/functional testing, i.e. verifies that all units behave according to their specifications without regard to how that functionality is implemented. The Unit Testing feature has the flexibility to naturally fit any development process by matching and automating developers' and testers' work patterns, allowing them to focus on value-added tasks. Rational Test RealTime is integrated with native development environments (Unix and Windows) as well as with a large variety of cross-development environments.

Platforms:

Rational Test RealTime is available for most development and target systems including Windows, Unix (Solaris, HP-UX, Linux)

2. C++Test:

Kind of Tool :

A C/C++ unit testing tool that automatically tests any C/C++ class, function, or component.

Organization :

ParaSoft Corporation
http://www.parasoft.com/

3. JUnit:

Kind of Tool :

A regression testing framework used by developers who implement unit tests in Java. (freeware)

Organization :

JUnit.org:
E-mail: junit@objectmentor.com
http://www.junit.org/

Software Description:

JUnit is a regression testing framework written by Erich Gamma and Kent Beck. It is used by the developer who implements unit tests in Java.

Platforms :

Platforms running Java.

4. JsUnit (Hieatt)

Kind of Tool :

JavaScript unit testing framework (freeware)

Organization :

Edward Hieatt
E-mail: edward@jsunit.net
http://www.jsunit.net/

Software Description:

JsUnit is a Unit Testing framework for client-side (in-browser) JavaScript. It is essentially a port of JUnit to JavaScript.

Licensed under the GNU Public License 2.0, GNU Lesser Public License 2.1, and the Mozilla Public License 1.1

Platforms :

Platforms supporting JavaScript 1.4 or higher.

5. MinUnit:

Kind of Tool :

Minimal Unit Testing Framework for C (freeware)

Organization :

Jera Design
E-mail: jbrewer@jera.com
http://www.jera.com/techinfo/jtns/jtn002.html

Software Description:

A minimal unit testing framework for C. It doesn't use malloc, so it may be more suitable for certain kinds of embedded systems.

Platforms :

Any platform with an ANSI C compiler.

6. VectorCAST:

Kind of Tool :

Embedded Software

Organization :

Vector Software, Inc.
http://vectorcast.com/

Software Description:

VectorCAST is a software module test system that automates the testing of individual software components prior to system test. Automation includes: building of complete test harnesses, generation and execution of test cases, code coverage, and pass/fail reports. VectorCAST is customized to the target CPU, cross compiler and run-time environments for leading embedded development tools from: Wind River Systems, Green Hills, TI, Rational, Cosmic, Tasking, Mentor Graphics, Analog Devices, and MetaWare

Platforms :

Windows 2000, XP, Linux RedHat, SuSE, Debian, Mandrake, Sun-Solaris 2.5.1 and above, IBM AIX 5.3 and above

7. CTA++ :

Kind of tool :

C++ test harnessing tool, unit/integration testing

Organization :

Testwell Oy
http://www.testwell.fi/

Software Description :

CTA++ (C++ Test Aider) is a tool for unit testing C++ classes, libraries and subsystems. CTA++ facilitates effective testing characterized as: easy-to-use and powerful arrangement to model the test suite into test cases, various forms of assertions for automating the test result checking, clear PASS/FAIL reporting on test cases and the whole test session, making the test runs visible, compact HTML browsable reporting of test results, regression testing, reading actual and expected values from command line or from compact textual data files, support for stub functions, reusing test cases of base class when testing inherited classes, testing multi-threaded code, testing all the advanced features of C++ (inheritance, overloading, exceptions, private parts, etc.), and more. Read more from http://www.testwell.fi/ctadesc.html

Platforms :

Windows-2000/NT/9x, Solaris, HPUX, Linux

8. Test Mentor - Java Edition :

Kind of Tool :

Java component, unit and function test automation

Organization :

SilverMark, Inc.
http://www.silvermark.com/Product/java/stm/

Software Description:

A functional test and test modeling tool for Java developers & QA Engineers to use as they develop their Java classes, clusters, subsystems, frameworks, and other components, either deployed on the client or the server during unit and i ntegration testing.

Platforms :

Client (user-interface) runs on Windows platforms only, test execution on al l Java platforms

9. Aunit:

Kind of Tool:

Ada unit testing framework (freeware)

Organization :

ACT Europe
8 rue de Milan
75009 Paris
France
Phone: +33 1 49 70 67 16
Fax: +33 1 49 70 05 52
E-mail: sales@act-europe.fr
http://libre.act-europe.fr/aunit/

Software Description :

AUnit is a set of Ada packages based on the xUnit family of unit test frameworks. It's intended as a developer's tool to facilitate confident writing and evolution of Ada software. It is purposely lightweight, as one of its main goals is to make it easy to develop and run unit tests, rather than to generate artifacts for process management. The framework supports easy composition of sets of unit tests to provide flexibility in determining what tests to run for a given purpose.

Platforms :

Unix, Windows

10. Check :

Kind of Tool :

A unit test framework for C (freeware)

Organization :

SourceForge
http://check.sourceforge.net/

Software Description:

Check features a simple interface for defining unit tests, putting little in the way of the developer. Tests are run in a separate address space, so Check can catch both assertion failures and code errors that cause segmentation faults or other signals. The output from unit tests can be used within source code editors and IDEs.

Check was inspired by similar frameworks that currently exist for most programming languages; the most famous example being JUnit for Java.

Licensed under the GNU LGPL.

Platforms :

POSIX-compliant systems.

11. CppUnit:

Kind of Tool :

C++ unit test tool (freeware)

Organization :

SourceForge

http://cppunit.sourceforge.net/

Software Description:

CppUnit is the C++ port of the famous JUnit framework for unit testing. Test output is in XML or text format for automatic testing and GUI based for supervised tests.

Licensed under the GNU LGPL.

Platforms:

BeOS, MacOS, Windows, Linux, possibly others.

12. csUnit:

Kind of Tool :

"Complete Solution Unit Testing" for Microsoft .NET (freeware)

Organization :

csUnit.org

E-mail: info@csunit.orghttp://www.csunit.org/

Software Description:

csUnit is a unit testing framework for the Microsoft .NET Framework. It targets test driven development using .NET languages such as C#, Visual Basic .NET, Visual J# and managed C++.

Licensed under the GNU GPL.

Platforms :

Microsoft Windows

13. dotunit:

Kind of Tool :

Unit test framework for .net (freeware)

Organization :

Christian Sepulveda

E-mail: csepulv@atdesigntime.com

http://dotunit.sourceforge.net/

Software Description:

dotunit is a port of JUnit to the Microsoft .net platform. This testing framework allows for automated unit and functional tests which are vital for refactoring and regression testing.

Distributed under the BSD license.

Platforms :

Windows

14. DUnit:

Kind of Tool:

xUnit Testing Tool (freeware)

Organization :

DUnit group

http://dunit.sourceforge.net/

Software Description:

DUnit is a port of the popular JUnit tool to Borland's Delphi (Object Pascal).

Platforms :

Windows, Linux

15. GJTester:

Kind of Tool :

General Java testing tool

Organization :

TreborSoft

http://www.gjtester.com/

Software Description :

GJTester provides a powerful GUI to aid developers in building test cases and test scripts. It allows the testers to accomplish unit and regression test without programming effort. The tool is useful for testing CORBA, RMI and other server technologies as well.

Platforms:

Platforms supported by Java.

IV. CONCLUSION

- Software testing is an art. Most of the testing methods and practices are not very different from 20 years ago. It is nowhere near maturity, although there are many tools and techniques available to use. Good testing also requires a tester's creativity, experience and intuition, together with proper techniques.
- Testing is more than just debugging. Testing is not only used to locate defects and correct them. It is also used in validation, verification process, and reliability measurement.
- Testing is expensive. Automation is a good way to cut down cost and time. Testing efficiency and effectiveness is the criteria for coverage-based testing techniques.
- Complete testing is infeasible. Complexity is the root of the problem. At some point, software testing has to be stopped and product has to be shipped. The stopping time can be decided by the trade-off of time and budget. Or if the reliability estimate of the software product meets requirement.
- Software testing is an activity which is aimed for evaluating.
- Testing is used to locate defects and correct them and also in validation, verification process, and reliability measurement. There are commercial and free testing tools as well as open source testing tools. Among them, Open Source Testing Tools seems to the most popular. Comparatively it provides more flexibility, more eases of usage and allows tool customization also. In this paper several popular and notable open source software testing tools have been categorized, analyzed, compared and their respective strengths and weaknesses discussed. The future of the testing tools is indicative that they will become less complex, more easy to use more adaptive and faster too.

V. FUTURE SCOPE

The primary function of software testing is to detect bugs in order to correct and uncover it. The scope of software testing includes execution of that code in various environment and also to examine the aspects of code - does the software do what it is supposed to do and function according to the specifications? As we move further we come across some questions such as "When to start testing?" and "When to stop testing?" It is recommended to start testing from the initial stages of the software development.

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions.

Today, software has changed from a standard, single, windows-based platform to multiple platforms. When considering about growth of software test automation tools we can see that there will be new areas which will enhance user friendliness, integration between tools, combining different tools to fit user needs.

With script-less automation, new tools will be shaped so that users could setup automation testing in a simple and logical way. It means that technical skills and programming knowledge will not be required any more. Another aspect is that Visual Test Automation is the future for test object recognition in the modern software application development and testing world. It contains algorithms built on OCR (object character recognition), ICR (image character recognition) and gesture recognition technology (interpreting human gestures). Technology based on visual based object recognition will allow testers to execute scripts cross-client and cross-platform. Automated testing tools will become less complex, easy to use and adaptive. Users will not need weeks to learn how to use testing tools. This kind of software testing will be even faster and will give exact results.

REFERENCES

[1] http://www.cs.cmu.edu/afs/cs/project/edrc-ballista/www/Ballista COTS Software Robustness Testing Harness homepage.

[2] Victor R. Basili, Richard W. Selby, Jr. "Comparing the Effectiveness of Software Testing Strategies",Technical Report, Department of Computer Science, University of Maryland, College Park, 1985.

[3] Boris Beizer, Software Testing Techniques. Second edition. 1990

[4] A very comprehensive book on the testing techniques. Many testing techniques are enumerated and discussed in detail. Domain testing, data-flow testing, transactin-flow testing, syntax testing, logic-based testing, etc.

[5] Beizer, Boris, *Black-box Testing: techniques for functional testing of software and systems*. Publication info: New York : Wiley, c1995. ISBN: 0471120944 Physical description: xxv, 294 p.: ill. ; 23 cm.
This book is a comprehensive introduction to various methods of testing, using intuitive examples. Complete coverage of all important testing techniques, and up-to-date. The focus is black-box/functional testing. The author is an internationally known software consultant with almost four decades of experience in the computer industry.

[6] Joe W. Duran, Simeon C. Ntafos, "An Evaluation of Random Testing", IEEE Transactions on Software Engineering, Vol. SE-10, No. 4, July 1984, pp438-443.

[7] Hetzel, William C., *The Complete Guide to Software Testing, 2nd ed.* Publication info: Wellesley, Mass. : QED Information Sciences, 1988. ISBN: 0894352423.Physical description: ix, 280 p. : ill ; 24 cm.
[8] This book is a good guide to software testing. But it may not be as complete a guide as it was 10 years ago.
[9] William E. Howden. Functional program Testing and Analysis. McGraw-Hill, 1987.
[10] http://www.numega.com/devcenter/bc.shtml
[11] Introduction of the tools BoundsChecker by NuMega
[12] Norman Parrington and Marc Roper, *Understanding Software Testing*, Published by John Willey & Sons, 1989. ISBN:0-7458-0533-7; 0-470-21462-7
[13] http://www.rational.com/products/purify_unix/index.jtmpl
[14] Introduces the Purify tool.
[15] http://www.rstcorp.com/definitions/software_testing.html
[16] A definition of and introduction to software testing.
[17] A standard for testing application software, William E. Perry, 1990
[18] This book summarizes and standardizes many testing techniques.
[19] Software-reliability-engineered testing practice (tutorial);John D. Musa; Proceedings of the 1997 international conference on Software engineering , 1997, Pages 628 - 629
[20] Philip Koopman, John Sung, Christopher Dingman, Daniel Siewiorek, Ted Marz. Comparing Operating Systems Using Robustness Benchmarks. 16th IEEE Symposium on Reliable Distributed Systems, Durham, NC, October 22-24, 1997, pp.72-79.
[21] Philip Koopman, John DeVale. Comparing the Robustness of POSIX Operating Systems. Proceedings of FTCS'99, 15-18 June 1999, Madison, Wisconsin.
[22] Ballista paper: multi-version comparison method to find silent and hindering failures.
[23] Kropp, N. P.; Koopman, P. J.; Siewiorek, D. P. Automated robustness testing of off-the-shelf software components. Twenty-eighth Annual International Symposium on Fault-Tolerant Computing (Cat. No.98CB36224)
[24] Testing the Robustness of Windows NT Software
[25] A simple heuristic method to test the robustness of some NT and GNU library.
[26] John DeVale, Philip Koopman & David Guttendorf. The Ballista Software Robustness Testing Service. Proceedings of TCS'99, Washington DC.
[27] Describes the Ballista testing web server and client architecture.
[28] Harry Koehnemann, and Timothy Lindquist; Towards target-level testing and debugging tools for embedded software; Conference proceedings on TRI-Ada '93 , 1993, Page 288
[29] Argues that current debugging method is not efficient for embedded software and propose an improved method.
[30] Smith, C. U. Performance Engineering of Software Systems. Addison-Wesley, 1990.

