ISSN: 2454-1907
DOI: https://doi.org/10.29121/ijetmr.v5.i2.2018.147

IJETMR
International Journal of Engineering Technologies and Management Research
A Knowledge Repository

COMPARISON OF JAVA PROGRAMMING TESTING TOOLS

Shikha Gautam [*1]
[*1] ICT Research Lab, Department of Computer Science, IIIT-L, University of Lucknow, Lucknow, 226007, India

Abstract:
Various testing tools have been used to find defects and measure quality of software which have been developed in different languages. This paper provides the overview of various testing tools and analyzed Java programming testing tools because Java programming is very important due to its mature nature to develop software. Java testing tools are analyzed based on various quality attributes. Analysis shows that selection of testing tool depends on requirement.

Keywords: *Software Testing; Software Quality; Java; Requirement.*

Cite This Article: Shikha Gautam. (2018). "COMPARISON OF JAVA PROGRAMMING TESTING TOOLS." *International Journal of Engineering Technologies and Management Research*, 5(2), 66-76. DOI: https://doi.org/10.29121/ijetmr.v5.i2.2018.147.

1. Introduction

Software testing is a process of executing a program or application with the intent of finding the errors or quality of the product. Software testing is conducted when executable software exists. Testing used to find out errors and fix them to support and improve software quality. Testing check what all functions of software supposed to do and also check that software is not doing what it not supposed to do [1].

Software testing is more difficult process because of the vast array of programming languages, operating systems, and hardware platforms that have evolved. There are different types of software testing. Reviews, walkthroughs, and inspections are static testing; whereas executing programmed code with a given set of test cases is refer as dynamic testing. Software testing methods are divided into white box and black box testing. There are four levels of test: unit testing, integration testing, component interface testing, and system testing [2].

Testing is an important phase for software quality. Testing support quality by executing software code and find out errors. Typically testing is classified into three categories as shown in Table 1.

1) Functional Testing
2) Non-Functional Testing or Performance Testing
3) Maintenance

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

Table 1: Classification of Software Testing

Testing Category	Types of Testing
Functional Testing	Unit Testing UAT (User Acceptance Testing) Integration Testing Localization Interoperability Globalization
Non Functional Testing	Performance Load Flexibility Volume Functionality Scalability Usability
Maintenance	Regression Maintenance

Well tested software provides efficient resource utilization resulting in low cost. Thoroughly tested software ensures reliable and high-performance functioning of the software. Java is an Object-Oriented Programming (OOP), relatable, and much easier programming language which makes Java more flexible, modular, stable, scalable, robust, secure, and extensible programming language. So we select Java programming testing tool for comparison [3-4].

In section 2, overview of software testing tools is explained. In section 3, need for software testing tools is discussed. In section 4 analysis of Java testing tools is explained. Section 5 describes the conclusion of this paper.

2. Overview of Software Testing Tools

Software testing tool that measures important aspect of software may have few unexpected side effects also on that software. To develop software various programming languages are used such as: C, C++, Java, and Python. For software testing different testing tools are used as per requirement. Each and every testing tool has its own property and importance to test software products such as:

2.1. C and C++ Testing Tool

Embunit is a unit testing tool for programmers and testers to test software in C or C++. Embunit generates the unit test source code automatically. Embunit has been developed with flexibility and can be customized to create unit tests for virtually any hardware platform [5-6].

VectorCAST/C++ is a highly automated unit and integration test solution used by embedded developers to validate safety and business critical embedded systems [7].

CppTest is a portable, powerful, and simple, unit testing framework for automated tests in C++. Mainly focus on usability and extendability. In 2007 a semi-automatic prototype CppTest tool is

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

developed with the testing capability in three levels: (1) structural testing in method level, (2) state based class level testing through modeling state transition behaviors using an extended finite state machine (EFSM), and (3) system level black box testing with some traditional strategies [8].

10 years ago Daniel Marjamäki develops a versatile tool, Cppcheck that can check non standard code. Cppcheck is a static analysis tool for C and C++ code. Unlike C, C++ compilers, and many other analysis tools it does not detect syntax errors in the code. Cppcheck primarily detects the types of bugs that the compilers normally do not detect. The goal is to detect only real errors in the code [9].

TILO is an automated unit testing tool for C and C++ which is designed to help developers in creating and running unit tests. Unit tests will have to be coded, not written, and will be executed automatically by the tool [10].

2.2. Java Testing Tool

In 1987 five graduates of the California Institute of Technology who had been working on Caltech Cosmic Cube founded Parasoft Corporation. Parasoft is a static analysis tools that prevents defects and expose vulnerabilities. Automate number of defect prevention practices for C, C++, Java, and .NET languages. Basically used for security, reliability, performance, and maintainability. Original unit testing technology has been extended to code coverage analysis, regression testing, and traceability. In case of testing integration testing, API testing, load testing, penetration testing, and system testing is automated [11].

Judy is a mutation testing tool for Java. Judy detects classes, tests, and libraries by scanning file's content. Class paths are automatically calculated from found files [12].

CMTJava is a complexity measures tool for Java which is an easy to use code metrics tool. It is intended for mature software development organizations striving for productive development process resulting in high quality products [13].

SonarQube collects and analyzes source code, measuring quality, and providing reports for software projects. It combines static and dynamic analysis tools and enables quality to be measured continuously over time. SonarQube able to accurately measure technical debt may not accurately reflect the technical debt of a system. It is only one tool that evaluates technical debt [14].

2.3. Python Testing Tool

unittest unit testing framework was originally inspired by JUnit and has a similar flavor as major unit testing frameworks in other languages. It supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. Python unit testing framework sometimes referred as PyUnit [15-16].

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

pytest is a mature full featured Python testing tool that helps write better programs. The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries [17].

Automated testing of Django applications makes use of the fundamental test support built in to the Python language: doctests and unit tests Django provides an API and tools for that kind of integration. Django creates easy to use interface to handle the authentication [18].

Pylint is a Python source code analyzer which looks for programming errors, helps enforcing a coding standard, and sniffs for some code smells. Pylint has many rules enabled by default, way too much to silence them all on a minimally sized program. It's highly configurable and controls it from error within your code. Additionally it is possible to write plug-in to add your own checks [19].

3. Need for Software Testing Tools

Software testing is an activity to ensure whether the actual results match the expected results and to ensure that the software is defect free. It involves execution of a software component or system component to evaluate one or more properties. It can be either done manually or using tools. Software testing tool helps to identify errors, gaps or missing requirements to the actual requirements [20-21].

Software testing tool is needed to verify functionality, documentation, and operation as intended. Testing on multiple devices and operating systems can help ensure software works for many users as possible. It makes sure that the application performance is enough and users are satisfied with it. When the delivered product is of quality, it helps in gaining the confidence of the users. Software testing with strict test execution assures minimum maintenance cost. There cannot be any failures because it can be very costly in the later stages of the development [22-24].

Software testing tool is required for product quality and satisfied user. It also brings profit because user experience. Software needs to be understandable, simple, and easy to use. The biggest benefit of software testing leads to business optimization. Business optimization means more satisfied users, user retention, fewer costs of fixing a product, minimum costs of a user service, better quality, improved reputation, more reliable products, and brand image. Although it is sometimes difficult to explain the importance of testing, it shouldn't be measured purely in cost and time rather in the great value it brings [25-26].

4. Analysis

In this section Java testing tool are analyze. There are various testing tool which is used to incorporate software quality. Main Java testing tool is:

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

4.1. Testwell CMTJava- Code Complexity Measures Tool for Java

CMTJava helps to locate the problematic complex code functions and files from big code to assess them separately. Code complexity has effect on how difficult it is to test and maintain the program.

Why Code Complexity Analysis is Important

- Code complexity correlates with the defect rate and robustness of the application
- Complex code is difficult to test more errors in the final application
- Complex code is difficult to maintain
- For this reasons several standards (like ISO 26262) requires the enforcement of low code complexity
- Unnecessary complex code is often the reason for bad code quality and erroneous programs
- Complex code is difficult to test and to maintain
- As the costs of bad quality and erroneous programs can be very high, applications with a reasonable complexity helps us to save money.

What Is Measured by Testwell Cmtjava

Based on the static properties of the program code CMTJava gives estimates how error prone the program source code is due to its complexity, how long it will take to understand the code, what is the logical volume of the code, etc. As the project team has not usually time to inspect all the code produced by the project, CMTJava can assist in locating the modules which are most likely to cause problems in the future [27].

Testwell CMTJava analyses Java programs for the following metrics

a) Lines-of-code (LOC) metrics

LOCbl	number of blank lines
LOCcom	number of lines with comments
LOCphy	number of physical lines
LOCpro	number of lines with program code

b) Halstead's metrics

B	estimated number of errors
D	difficulty level, error proneness
E	effort to implement
L	program level (abstraction level of the program)
N	program length
N1	number of operators
N2	number of operands
n	vocabulary size (unique operators + unique operands)
n1	number of unique operators

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

n2	number of unique operands
T	implementation time / time to understand
V	volume: size of the implementation of an algorithm

c) McCabe cyclomatic number v(G)

Cyclomatic number v(G) describes the complexity of the control flow of a program.

d) Maintainability Index

Maintainability Index is calculated with certain formulae from lines of code measures, McCabe measures and Halstead measures. The measurement and track maintainability are intended to help reduce or reverse a system's tendency toward code entropy or degraded integrity, and to indicate when it becomes cheaper and less risky to rewrite the code instead to change it.

Testwell CMTJava Features

- Measures original non-pre-processed files
- Extremely fast analyses your applications in a couple of minutes
- Can handle many big files
- HTML or textual reporting measurements can be further processed by Excel
- GUI integration in Visual C++ Developer Studio
- Available on many platforms: Windows, Linux, HP, Solaris

Verybench: Front End with Several Graphical Views

Verybench is a graphical user interface for the code complexity measuring tool Testwell CMT++.

It enables managers, developers as well as quality and test engineers to collaborate easier on and to contribute easier to source code quality. It accelerates the way metrics are understood by an entire development team and not just by single individuals involved in improving the quality of software. Verybench comes with two kinds of main views: the Dashboard Views and the Metrics Views [28].

4.2. SonarQube

SonarQube is an open source platform for continuous inspection of code quality.

Features

- Offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities .
- Records metrics history and provides evolution graphs.
- Provides fully automated analysis, integrates with Maven, Ant, Gradle, MSBuild, and various continuous integration tools (Atlassian Bamboo, Jenkins, Hudson, etc.)

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

- Integrates with Eclipse, Visual Studio and IntelliJ IDEA development environments through the SonarLint plugins.
- Integrates with external tools: LDAP, Active Directory, GitHub, etc.
- Is expandable with the use of plugins.

SonarSource delivers what is probably the best static code analyzer for Java. Based on Java compiler front end, it uses the most advanced techniques (pattern matching, dataflow analysis) to analyze code and find code smells, bugs and security vulnerabilities. For any product SonarSource is developed for following principles: depth, accuracy, and speed. SonarJava has a great coverage of well established quality standards. The SonarJava capability is available in Eclipse and IntelliJ for developers (SonarLint) as well as throughout the development chain for automated code review with on premise SonarQube or online sonarqube.com [29-30].

4.3. Codacy

Codacy is an automated code review platform for Python, Ruby, PHP, Java, JavaScript, and Scala programming languages. It integrates several open source static analysis tools and also offers rules from other internally developed tools. Users can check code for security and duplication issues, accuracy, complexity, coding style, and other metrics. The platform also parses logs from code coverage tools and presents the evolution of code coverage side by side with the other software metrics.

Codacy is configurable, allowing code patterns to be enabled or disabled. It integrates with GitHub, Bitbucket, JIRA, YouTrack, Heroku, HipChat, and Slack. Codacy also offers an Enterprise version that runs in the clients' servers and integrates with Jenkins, GitHub Enterprise, Bitbucket Server, and GitLab. Codacy is free for open source software projects, and offers paid accounts for software developers who wish to have private repositories [31].

4.4. Coverity

Coverity is a brand of software development products from Synopsys, consisting primarily of static code analysis and dynamic code analysis tools. The tools enable engineers to find defects and security vulnerabilities in source code written in C, C++, Java, C#, and JavaScript [32].

Coverity Code Advisor is a static code analysis tool for C, C++, C#, Java, and JavaScript. It is derived from the Stanford Checker, a research tool for finding bugs through static analysis. Static code analysis finds defects and security vulnerabilities in source code without executing the code. Also known as Static Application Security Testing (SAST), it is used to improve software quality and security with automation in IoT, Automotive, Medical, Enterprise, Cloud, Mobile, Social, Shared Economy, Analytics, and mission critical software development lifecycle [33].

4.5. LDRA

LDRA Testbed provides the core static and dynamic analysis engines for both host and embedded software. LDRA Testbed provides the means to enforce compliance with coding

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

standards such as MISRA, JSF++ AV, CERT C, CWE, and provides visibility of software flaws that might typically pass through the standard build and test process to become latent problems. In addition, test effectiveness feedback is provided through structural coverage analysis reporting facilities which support the requirements of the DO-178B standard up to and including Level-A. Quality metrics such as Halstead complexity measures, Cyclomatic complexity, Knots metric are designed to verify that code is clear, maintainable, and testable. The quality report in the LDRA tool suite presents both a summary and detailed breakdown of quality metrics which are deduced during static analysis [34].

LDRA TBvision is the interactive environment for LDRA Testbed that lets you easily visualize coding standards compliance and quality metrics and rapidly address identified flaws at the source code level. With 40 years of code analysis experience in safety critical, security critical, and business critical applications, LDRA Testbed and TBvision provide complete confidence for your software development project.

- LDRA Increase Productivity with Automated Unit/Integration Testing with TBrun
- Delivering Software Quality and Security through Test, Analysis and Requirements Traceability

TBrun automates and increases test throughput and repeatability to significantly increase overall test effectiveness. Software development managers seeking to develop the highest quality code in the most cost effective manner are leveraging automated unit and integration testing to avoid the potential delays caused by inefficient manual low level testing strategies. These traditional techniques often are inadequate and postpone the discovery and correction of defects until late in system test where they are most expensive to fix [35].

4.6. Yasca

Yasca is an open source program which looks for security vulnerabilities, code quality, performance, and conformance to best practices in program source code. It leverages external open source programs, such as FindBugs, PMD, JLint, JavaScript Lint, PHPLint, Cppcheck, ClamAV, Pixy, and RATS to scan specific file types, and also contains many custom scanners developed for Yasca. It is a command line tool that generates reports in HTML, CSV, XML, MySQL, SQLite, and other formats. It is listed as an inactive project at the well known OWASP security project, and also in a government software security tools review at the US Department of Homeland Security web site [36-37].

4.7. AppPerfect- Java Code Testing

Locating and fixing performance problems during source code development time is arguably the cheapest way to resolve problems. As your project goes past the development phase, in to testing and deployment, the cost of fixing problems grows exponentially. By conducting source code analysis and successfully identifying and correcting all such issues, software developers can eliminate the risk and potential costs early in the software development cycle [38].

Java Code Testing involves testing Java source code against industry standard coding rules and ensuring that Java source is stable, optimized, consistent, portable, and is compliant with coding

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

standards. Automated Java Code Testing is more efficient and less error prone as compared to manual code reviews. Java Code Testing accomplishes a thorough and accurate review that checks for all defined cases. An automated Java Code Testing boosts development, saves cost, and enhances the application quality by performing a test function or code review with greater consistency and in much lesser time. Performing such accurate reviews manually is tiresome, tedious and expensive. Automated Java Code testing also gives the ability and flexibility to a project manager to enforce coding standards and statistical quality control measures. The indicators of code size, complexity, and density can be computed using automated Java Code testing, which helps in the quality control of the code [39].

Analysis of Java programming testing tools are depicted in Table 2. Each and every tool is used for specific objective and requirement. If maintainability required then Testwell CMTJava and LDRA Testbed are selected. For accuracy SonarQube and Codacy are used. In case of security Codacy, Coverity, and Yasca tool are used.

Table 2: Analysis of Java Programming Testing Tools

Java Programming Testing Tools	**Quality Attributes**
Testwell CMTJava	Maintainability
SonarQube	Accuracy
Codacy	Accuracy and Security
Coverity	Security
LDRA Testbed	Clarity, Maintainability, and Testability
Yasca	Security and Performance
AppPerfect	Stability, Optimization, Consistency, and Portability

LDRA Testbed also used for clarity and testability. For performance Yasca tool is suitable. AppPerfect is used for stability, optimization, consistency, and portability [40].

5. Conclusion

This paper provides the overview of various testing tools and compare important Java testing tool. Java testing tools are analyzed based on various quality attributes. Analysis shows that selection of testing tool depends on requirement. If maintainability require then Testwell CMTJava and LDRA Testbed are selected. For accuracy SonarQube and Codacy are used. In case of security Codacy, Coverity, and Yasca tools are used. LDRA Testbed also used for clarity and testability. For performance Yasca tool is suitable. AppPerfect is used for stability, optimization, consistency, and portability.

Acknowledgment

Ms. Shikha Gautam is thankful to Prof. Brijendra Singh, Ph.D. supervisor for proper guidance to do creative research work and IIIT Lucknow to give such a beautiful and research-oriented environment.

ISSN: 2454-1907
DOI: 10.5281/zenodo.1179382

References

[1] Bertolino, "Software testing research: Achievements, challenges, dreams," In Future of Software Engineering, *IEEE Computer Society*, pp. 85-103, May 2007.

[2] V. R. Basili and R. W. Selby, "Comparing the effectiveness of software testing strategies," *IEEE transactions on software engineering*, Vol. 12, pp. 1278-1296, Dec 1987.

[3] J. A. Whittaker, "What is software testing? And why is it so hard?" *IEEE software*, Vol. 17, No. 1, pp.70-79, Jan 2000.

[4] http://www.guru99.com/software-testing-introduction-importance.html

[5] http://www.embunit.com

[6] M. J. Karlesky, W. I. Bereza and C. B. Erickson, "Effective test driven development for embedded software," In *IEEE International Conference on Electro/information Technology*, pp. 382-387, May 2006.

[7] http://www.vectorcast.com/software-testing-products

[8] Mao and Y. Lu, 2007, "CppTest: A Prototype Tool for Testing C/C++ Programs," In *IEEE Second International Conference on Availability, Reliability and Security*, ARES 2007, pp. 1066-1073, April 2007.

[9] Marjamäki, "Cppcheck: a tool for static C/C++ code analysis," 2007 http://cppcheck.sourceforge.net/

[10] https://www.codeproject.com/Articles/11453/TILO-An-Automated-Unit-Testing-Tool-for-C-and-C

[11] G. Chatzieleftheriou and P. Katsaros, "Test-driving static analysis tools in search of C code vulnerabilities," In *IEEE 35th Annual Computer Software and Applications Conference Workshops* COMPSACW-2011, pp. 96-103, July 2011.

[12] L. Madeyski and N. Radyk, "Judy–a mutation testing tool for Java," *IET software*, Vol. 4, No. 1, pp.32-42, Jan 2010.

[13] N. Rai, "An Automated Tool for Computing Software Metrics," In *Undergraduate Research Symposium*, p. 18, 19 April 2006.

[14] M. R. Dale and C. Izurieta, "Impacts of design pattern decay on system quality," In Proceedings of the *8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, p. 37, Sep 2014.

[15] https://docs.python.org/3/library/unittest.html

[16] X. Bai and M. Sadeghi, "IT Module Based Test Automation Framework," In *Eighth International Conference on Information Technology: New Generations*, ITNG 2011, pp. 263-267, April 2011.

[17] A. Pajankar, "pytest- In Python Unit Test Automation," Apress, pp. 87-100, 2017.

[18] J. McGaw, "Django Testing- In Beginning Django E-Commerce, Apress, pp. 299-321, 2009.

[19] S. Sripada, Y. R. Reddy and A. Sureka, "In support of peer code review and inspection in an undergraduate software engineering course," In *IEEE 28th Conference on Software Engineering Education and Training*, CSEET 2015, pp. 3-6, May 2015.

[20] A. Podgurski and L. A. Clarke, "A formal model of program dependences and its implications for software testing, debugging, and maintenance," *IEEE Transactions on software Engineering*, Vol. 16, No. 9, pp. 965-979, Sep 1990.

[21] B. Hailpern and P. Santhanam, "Software debugging, testing, and verification," *IBM Systems Journal*, Vol. 41, No. 1, pp. 4-12, Jan 2002.

[22] D. Gelperin and B. Hetzel, "The growth of software testing," *Communications of the ACM*, Vol. 31, No. 6, pp. 687-695, June 1988.

[23] P. Karhapää, A. Haghighatkhah and M. Oivo, "What Do We Know about Alignment of Requirements Engineering and Software Testing?" In Proceedings of the *ACM 21st International Conference on Evaluation and Assessment in Software Engineering*, pp. 354-363, June 2017.

[24] R. Kaur and I. Singla, "Optimization of Software testing techniques," *International Journal for Scientific Research & Development*, Vol. 2, Issue 03, pp. 1762-1765, Mar 2014.
[25] S. R. Jan, S. T. Shah, Z. U. Johar, Y. Shah and F. Khan, "An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," *International Journal of Scientific Research in Science, Engineering and Technology*- IJSRSET, pp. 2395-1990, March-April 2016.
[26] V. Garousi and M. V. Mäntylä, "When and what to automate in software testing? A multi-vocal literature review," *Information and Software Technology*, Vol. 76, pp. 92-117, Aug 2016.
[27] http://www.verifysoft.com/en_cmtx.html
[28] http://www.verifysoft.com/verybench/index.html
[29] https://www.sonarqube.org/
[30] G. Campbell and P. P. Papapetrou, "SonarQube in action," Manning Publications Co, 2013.
[31] https://www.codacy.com/
[32] www.coverity.com/
[33] https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html
[34] www.ldra.com/en/testbed-tbvision
[35] J. B. Michael, B. J. Bossuyt and B. B. Snyder, "Metrics for measuring the effectiveness of software-testing tools," In Proceedings on *IEEE 13th International Symposium Software Reliability Engineering*, ISSRE 2003, pp. 117-128, 2002.
[36] http://www.scovetta.com/yasca.html
[37] M. Scovetta, "YASCA-Yet Another Source Code Analyzer," 2009.
[38] http://www.appperfect.com/index.php
[39] K. Daimi, S. Banitaan and K. Liszka, "Examining the performance of java static analyzers," In Proceedings of the *International Conference on Software Engineering Research and Practice* (SERP), The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing, WorldComp, Jan 2013.
[40] A. P. Mathur, "Performance, effectiveness, and reliability issues in software testing," In Proceedings of the *IEEE Fifteenth Annual International Computer Software and Applications Conference*, COMPSAC'91, pp. 604-605, Sept 1991.

*Corresponding author.
E-mail address: shikhagautam90@gmail.co/shikhagautam@ iiitl.ac.in