An Investigation into the Effectiveness of Java Code Coverage Tools

Shaik Khasim Saheb, Devavarapu Sreenivasarao, Prof.T.V.Narayana Rao M Kiran Kumar
Department of Computer Science and Engineering
Sreenidhi Institute of Science and Technology, Hyderabad, India

Abstract— Coverage is a measure used in software testing. It decides the percentage in which the source code of a program has been tested. Today so many code coverage tools are available to check how much code is being tested. Some tools have their own advantages and limitations. In this paper, we propose to give a survey in such a way that which tool is giving effective results when compared with other tools in consideration of all scenarios. We also explain how to catch, analyze and testing the code of java programs. The major part of this survey examines the most commonly used coverage based testing tools. We also survey over-and-above features which are affiliated with code coverage based testing tools. Such features accomplish tools more advantageous and experimental, particularly for large-scale, real-life popular software applications. This study make-believe each tool has its individual features attached to its application domains. Therefore, this study can be used to select the right coverage testing tools based on various requirements.

Keywords—*code coverage, test cases, measurements, analysis and effectiveness*

I. INTRODUCTION

Since the booming aggressive burden surrounded by numerous software vendors, the desire for high quality software has expanded. Software quality shows an extensive aspect in industries whose products build on software for their application. Subsequently, there is advancing stress in these organizations to increase software quality. Software testing is a profession generally dedicated to take a decision and sometimes to increase software quality. Of course, estimating the time and resources that should be allotted to testing hold a deal among budget, time and quality. Obtaining an impressive and active software testing tool could be a life-saver for a project or a company. Moreover there is no single test tool applicable for all accessible systems and multinational sectors. Concluding what basis to implement when picking a clear-cut tool for a project is absolutely critical. For example, some tools concatenate seamlessly with your option of IDE (e.g. Eclipse) and produce easy to understand interfaces to comfort unit testing in the development phase, but have scalability problems. Those tools are fit for a small project, but not for a extensive real-life popular application that sometimes appended with huge percentage of legacy code. This survey examines individual fact that practitioners should recognize when selecting coverage based testing tool. It encloses the following issues. First, the theme of testing is very spacious. One exhaustive, but pathetic representation to classify testing proposals is the structural/behavioral or white-box/black-box model. Structural tests, also known as white-box tests, are based on how a system operates. They cover a thorough awareness of the execution of a pattern. Behavioral tests, also known as black-box tests, are hold on what a system is mandatory to do. They benefit conventional user synopsis lacking enquiry into the code. Because of their several views of the structure, black box and white box test tools are incomparable. We concentrate this review on tools that evaluate testing coverage. Coverage-based testing gives a avenue to compute the status of thoroughness of white-box testing. Second, there are various suitable test tools, both popular and open-source software. We chosen only those with code coverage characteristics. We justified five tools that meet our category. Facts about them are available in the public domain. Innermost or private coverage-based test tools are not in the scope of this survey. The rest of this survey is standardized as follows. Section 2 provides an outline of coverage. Section 3 shows about some Test Coverage tools. Section 4 discusses some important aspects of coverage measurement, along with programming languages, auxiliary features to code coverage. Section 5 presents various coverage criteria sustained by each tool. Section 6 concludes the paper in consideration with the commonly used features which are specific to the coverage based testing tools.

II. OUTLINE OF COVERAGE

Code coverage specifies to a software engineering approach whereby you trace quality and completeness of your collection of test cases by establishing simple metrics like the percentage of {classes, methods, lines, etc.} that got executed when the test suite ran. Practice represents that coverage percentages below, say, 60-70% belongs to incompletely tested software. You can imagine unknown bugs in such software. Because of this, "good" software package firms in still internal processes whereby a team cannot unharness a chunk of software package unless it passes unharness gates like "line coverage should be eightieth or higher"[3][6].

Incidentally, reaching for a 100% coverage is not profitable either. you only get plenty less quality improvement for significantly a lot of effort to succeed in such perfection. Coverage near 85-90% is "good

enough" for all sensible functions. The topic of that coverage metric is "better" may be somewhat spiritual. There are tutorial studies showing that, for instance, path coverage at a definite level detects somewhat additional bugs than, say, line coverage at a similar level. we predict the particular metric definition isn't that necessary. We'd rather empower all developers in our team with a free and quick tool so they'll track their own coverage (of some kind) early and regularly. associate degree tough developer can look into the coverage report that links to the ASCII text file, drill down a little, look into the "red" areas, and work out that, if any, areas of the merchandise he left somewhat under-tested. Code coverage analysis is that the method of: Finding areas of a program not exercised by a group of check cases, making extra check cases to extend coverage, and determinative a quantitative live of code coverage, that is Associate in Nursing indirect live of quality. Associate in Nursing optional side of code coverage analysis is distinguishing redundant check cases that don't increase coverage. A code coverage instrument automates this method. you employ coverage analysis to assure quality of your set of tests, not the standard of the particular product. you are doing not typically use a coverage instrument once running your set of tests through your unharness candidate. Coverage analysis needs access to check program open source and infrequently needs recompiling it with a special command. This paper discusses the small print you ought to think about once attending to add coverage analysis to your take a look at arrange. Coverage analysis has sure strengths and weaknesses. you want to select from a spread of activity ways. you ought to establish a minimum share of coverage, to work out once to prevent analyzing coverage. Coverage analysis is one in all several testing techniques you ought to not believe it alone. Code coverage analysis is usually referred to as take a look at coverage analysis. the 2 terms ar synonymous . the tutorial world a lot of usually uses the term "test coverage" whereas practitioners a lot of usually use "code coverage". Likewise, a coverage instrument is usually referred to as a coverage monitor[1][5].

III. TEST COVERAGE TOOLS

This section presents state-of-art coverage based mostly tools that perform code coverage analysis. There square measure several take a look at coverage tools that square measure offered in literature and web, commercially or work version. Following square measure some take a look at coverage tools[4].

A. JavaCodeCoverage

JavaCodeCoverage is associate degree open supply bytecode analyser tool for take a look at coverage analysis for Java code which needs neither the language descriptive linguistics nor the open source. A very important side of JavaCodeCoverage is that it stores the coverage info for individual legal action thereby facilitating careful coverage analysis. Another vital side of JavaCodeCoverage is that it records all very important code elements and take a look at coverage info in open supply info code MySQL .

B. JFeature

JFeature is associate degree open supply feature/requirement coverage tool that facilitates that specialize in needs as code is developed. It lets leverage from customary development practices to urge a lot of insight into the wants lined by the code. it's a plug-in for the Eclipse IDE and conjointly permits user to import needs and match them to JUnit take a look at cases at intervals Java application.

C. JCover

JCover is a code coverage instrument for Java programs. It provides a mechanism to get applied mathematics info on the coverage of associate degree application throughout a check run. It is often wont to calculate the share of code that was dead, proportion not dead, what sources weren't employed in files then on. JCover supports statement and branch coverage.

D. Cobertura

Cobertura is a free open supply Java tool that calculates the proportion of code accessed by tests. It will be wont to establish that elements of Java program area unit lacking check coverage. It will be dead from pismire or from the command[2].

E. Emma

It is associate open source tool for measurement and news code coverage for Java. It will instrument categories for coverage either offline (before they're loaded) or on the fly (using associate instrumenting application category loader). Supported coverage varieties are category, method, line and basic block.

F. Clover

Clover is accessible as either associate Eclipse or plan plugins or victimization hymenopter script. It supports statement, method, class, and package coverage. This tool provides correct, configurable coverage analysis. Coverage news is in XML, HTML, or via a Swing graphical user interface. It is a low cost coverage tool.

G. Quilt

Quilt may be a Java software system development tool that measures coverage, the extent to that unit checking exercises the software system below test It is optimized to be used with the JUnit unit take a look at package, the hymenopteron Java build facility, and also the maven project management toolkit. Quilt intercepts code because it is being loaded and alters it. It doesn‘t work on the ASCII text file level. It manipulates compiled categories and their bytecode, the code of the Java Virtual Machine, the JVM.

H. CodeCover

CodeCover is a free testing tool for Java programmers. it's totally integrated into Eclipse and performs supply instrumentation for coverage mensuration particularly for condition coverage. It helps to extend take a look at quality.

I. InsECT

InsECT (Instrumentation Execution Coverage Tool), is a system developed in Java to get coverage info for Java programs. It instruments Java category files at the byte code level. The aim of InsECT is to produce elaborate coverage info concerning Java programs.

J. Jester

Jester is for Java code and JUnits tests. It finds code within the software package that's not coated by tests. Jester‘s approach is named mutation testing or machine-driven error seeding.

K. Hansel

It is associate extension to JUnit. Hansel offers terribly helpful data that what proportion of the code that a check is meant to check is covered? It deals with branch coverage of the category.

L. JBlanket

JBlanket could be a tool for assessing and up technique coverage of unit action at law. it's meant for each stand alone and consumer server programs.

M. Coverlipse

Coverlipse is associate Eclipse plug-in for code coverage visual image. The coverage results square measure shown when a JUnit check run. It supports branch, block and all-uses coverage.

N. BullseyeCoverage

It is a C and C++ code coverage instrument tool that tells what proportion of ASCII text file was tested . It pinpoints areas that require attention to be reviewed. Supported coverage varieties square measure operate and condition/decision. BullseyeCoverage supports the widest vary of platforms of any code coverage instrument including Windows and Linux.

O. NCover

NCover is associate open ASCII text file coverage tool for .NET platform. It provides a awfully powerful and versatile tool set which might integrate into build method and facilitate to deliver higher quality code. It tells regarding what number times every line of code was dead throughout a selected run of the applying. It supports technique and sophistication coverage.

P. Testwell

CTC++ Testwell CTC++ could be a powerful instrumentation-based check coverage and dynamic analysis tool for C and C++ code. It shows the coverage all the thanks to the changed condition/decision coverage (MC/DC) level as needed by comes. The tool is lightweight however still contains all the essential ―must‖ options of associate business strength testing tool.

Q. eXVantage

It is a tool suite for code coverage testing, debugging and performance identification. It supports Java and C/C++ platforms. eXVantage uses ASCII text file instrumentation for C/C++ and byte code instrumentation for Java. It analyzes the program in such some way that it will choose the smallest amount variety of probes to be inserted into the object program, that‘s why it's the best off-line instrumentation overhead.

R. OCCF

OCCF (Open Code Coverage Framework) supports multiple programming languages. A sample tool is created for C, Java and alternative languages victimization OCCF. The researchers developed a tool which will live four coverage criteria. They reduced prices by reusing common code, and obtained consistent measurements by supporting multiple languages, versatile measurements through increasing options, and complete mensuration‘s by inserting measurement code into the source code.

S. JAZZ

JAZZ may be a structural testing tool. It will branch, node, and def-use coverage and implements a GUI, check planners, dynamic instrumentation, and a check instrument. Jazz is incorporated in Eclipse for the Intel x86. Instrumentation is dynamically inserted on demand because the program executes. Instrumentation is additionally deleted at the time it's now not required.

IV. COVERAGE MEASUREMENT

All tools enclosed during this survey have coverage mensuration capability. This section compares these tools for 3 vital coverage tool characteristics: (i) supported programming languages, (ii) program instrumentation overhead and (iii) further options complementary to code coverage.

A. Supported languages

The selection of languages reflects every company‘s target industries. Corporations that offer tools for system package, or embedded package vendors tend to focus additional on supporting C/C++. Table one shows an entire list of the tools and therefore the languages that they support. Such tools square measure designed to introduce minimum performance overhead in order that the tool is usable in period environments.

Table I.Coverage Tools and the Languages to Which They Apply

Tool Name	C++/C	Java	Other
Cobertura		X	
Emma		X	
Clover		X	.net
JavaCodeCoverage		X	
JFeature		X	
Clover		X	
JCover		X	
Quilt		X	
Code Cover		X	COBOL
Jester		X	
Hansel		X	
BullseyeCoverage	X		
NCover			.net
Testwell CTC++	X		
eXVantage	X	X	
OCCF	X	X	X
JAZZ	X		

In follow, debugging invariably follows testing. a number of the coverage tools give debugging help, like JCover, JTest. Their solutions square measure all completely different. for instance JCover has the power to try and do coverage differencing and comparison to reveal the error code.

B. Instrumentation overhead

Evaluation of Code Coverage is that the downside of characteristic the parts of a program that didn't execute in one or a lot of runs of a program. Developers and testers use code coverage to confirm that each one or well all statements in a very program are dead a minimum of once throughout the testing method. measure code coverage is very important for testing and confirmatory code throughout each development and porting to new platforms. historically code coverage measure tools are designed victimization static code instrumentation. throughout program compilation or linking, these tools insert instrumentation code into the binary possible file. The inserted instrumentation provides counters to record that statements ar dead. The code inserted into the executable remains in the executable throughout the execution even though once a statement has been executed, the instrumentation code produces no additional coverage information. Moreover, these tools conservatively instrument all functions prior to the program execution even though some of them may never be executed. Leaving useless instrumentation in place increases the execution time of the software being tested especially if the program is long running and has many frequently executed paths (as most server programs due).

1) Off-line program analysis and instrumentation

Overhead Source code instrumentation, employed by most of the tools as well as BullseyeCoverage, Intel Code Coverage Tool, linguistics styles and TestWork, needs recompilation, however provides a lot of direct results and is a lot of variable to a large form of processors and platforms. It can't be used once the ASCII text file isn't obtainable, as is usually the case for third party code. C/Cþþ tools like Dynamic Memory Systems‘ Dynamics, use runtime instrumentation, that makes them possible in a very production atmosphere. They will be a lot of economical in terms of compilation time, however less transportable. The Java coverage tool Koalog Code Coverage doesn't need instrumentation, and so no recompilation is required . It operates with the assembly binaries exploitation the Java right Interface, that is an element of the Java Platform computer program design (JPDA). Koalog Code Coverage is platform freelance, however needs a JPDA compliant Java Virtual Machine (JVM). Agitar‘s mischief-maker runs the code in a very changed JVM, conjointly employing a dynamic instrumentation approach. eXVantage uses ASCII text file instrumentation for C/Cþþ and bytecode instrumentation for Java. As compared to the opposite sixteen tools, it's the very best off-line instrumentation overhead as a result of it analyzes the program in such some way that it will choose the smallest amount variety of probes to be inserted into the computer program.

2) Run-time instrumentation overhead

Companies that offer tools for system code or embedded code tend to focus a lot of on reducing run-time overhead, so their tools are often usable in period environments, e.g. CodeTEST. TCAT claims that its TCAT C/ Cþþ Version three.2 maintains its overhead for execution size magnitude relation at one.1– 1.8 and swiftness magnitude relation at one.1–1.5, linguistics styles claims one.1–1.3, variable in step with language and compiler, among the simplest in our survey. Clover claims that their fastness overhead is extremely variable, betting on the character of the appliance underneath take a look at, and therefore the nature of the tests.

TABLE 2. Instrumentation

Tool Name	Source Code Instrumentation	Byte Code Instrumentation
JavaCodeCoverage		X
JFeature	X	
JCover	X	
Cobertura		X
Emma		X
Clover	X	
Quilt		X
Code Cover	X	
Jester		
GroboCodeCoverage		X
Hansel		
BullseyeCoverage	X	
NCover	X	
Testwell CTC++	X	
eXVantage	X	X
OCCF	X	
JAZZ	X	

V. COVERAGE MEASUREMENT CRITERIA

There square measure an outsized type of coverage activity criteria knowledge coverage, statement (line) coverage, block coverage, call (branch) coverage, path coverage, function/method coverage, category coverage and execution state house coverage. Among them, statement (basic block) coverage, call coverage, function/method coverage and sophistication coverage are enforced by some coverage tool vendors. the {remainder} remain principally of interest to researchers, due to their redoubled complexness and problem of use. Practitioners generally don't use different criteria like knowledge coverage as a result of it's more durable to enhance knowledge coverage. For example, supported our observations, it's terribly onerous to induce higher than some % knowledge coverage. software package tool corporations, particularly little software package vendors, look for immediate come on their investment, and therefore the level of sophistication of their users directs their focus onto usability, over painstakingness, or accuracy. Table three lists tools with their coverage measuring criteria. In Table 3, statement coverage means that the proportion of (executable) statements dead, whereas block coverage measures coverage of basic blocks, wherever a basic block may be a sequence of non-branching statements. The results for statement coverage and block coverage may take issue, however they're ordinarily listed within the same class and so within the same column in Table three for simple comparison. eXVantage and Intel Compiler Code-Coverage live block coverage. The number of tools, e.g. Koalog, give the

TABLE 3.Levels of coverage measurement provided by tools.

Tool Name	Statement/ Line/Block	Branch/ decision	Method/ function	Class
JavaCodeCoverage	X	X	X	
Jfeature			X	
Jcover	X	X	X	X
Cobertura	X	X		
Emma	X		X	X
Clover	X	X	X	X
Quilt	X	X		
Code Cover	X	X		
Jester				
GroboCodeCoverage				X
Hansel		X		
BullseyeCoverage		X	X	
Ncover			X	X
Testwell CTC++		X		
eXVantage	X	X	X	
OCCF	X	X		
JAZZ		X		

choice of selecting the scope for coverage calculations, as an example, the statement coverage in an exceedingly technique, a category or a package. Clover permits users to customise the scope by providing refined method- and statement-based filtering of coverage results this could facilitate to come up with additional informative reports and can be more mentioned in Section five. Line coverage and statement coverage dissent once over one statement might contribute to one line‘s coverage score or one statement takes over one line. trefoil uses statement coverage. However, most of the venders don't distinguish between statement and line coverage, thus we tend to list them in Table three.

There square measure many variations of condition/decision coverage, however we tend to list all of them within the third column while not differentiation. a call is that the whole expression that affects the flow of management within the CFG and is treated as one node within the CFG. A condition/branch could be a sub-expression in an exceedingly call expression, connected by logical-and and logical-or operators. BullseyeCoverage and Metrowerk's CodeTEST live changed condition/decision coverage for C/Cþþ, that provides a decent balance of usability and painstakingness. Branch coverage provides the amount of branches dead below take a look at. Clover, Cobertura and TCAT/Java support branch coverage for Java. Methodology coverage reports for every methodology or operate whether or not or not it's invoked. Category coverage reports {a category|a category} as lined if a minimum of one line in this class is dead. Neither methodology nor category coverage provides fine roughness, however they are doing give an summary of testing quality.Software Research Inc's TCAT/JAVA uses an algorithm, called _All Paths Generator', which is consumed, to calculate simple path coverage. It is designed for use on critical applications where test completeness is required. The applicability of different measures is affected by the style of the code, such as the size of a method or a function, and the density of branching. For example, method coverage, which counts a method as covered if at least one line in that method is executed, is more suitable for software that consists of many small methods rather than a few large methods.

Supported Languages	**Tool name**	**Measurements**					
		Statement/line/block	Branch/decision	Method/function	Class	GUI	Reports
Java	Clover	X	X	X		X	X
	Cobertura	X	X			X	X
	EMMA	X		X	X	X	X
	JCover	X	X	X	X	X	X
	Koalog	X					
	JavaCodeCoverage	X	X	X		X	X
	JFeature			X		X	X
	Clover				X	X	X
	Quilt	X	X				
	Code Cover	X	X			X	X
	Jester				X	X	
	GroboCode Coverrage				X		
	Hansel		X				
C/C++	Code TEST	X	X				
	BullseyeCoverage		X	X		X	X
	Testwell CTC++		X				X
Java and C/C++	eXVantage	X	X			X	X
	OCCF	X	X				X

VI. Conclusion

This paper evaluates some test coverage tools. We have compared five features language support, instrumentation, coverage measurement, GUI and reporting. In our opinion, these are the best criteria to test the coverage tools. Table 5 summarizes our analysis. Each tool has some strong and weak points. Users and developers can select the tool according to their need. We hope our work will help in more usage and selection of tools.

Supported Languages	Tool name	Instrumentation		
		Source code instrumentation	Byte code instrumentation	On the fly (dynamic)
Java	Clover	X		
	Cobertura		X	
	EMMA		X	X
	JCover	X	X	
	Koalog			X
	JavaCodeCoverage		X	
	JFeature	X		
	Clover	X		
	Quilt		X	
	Code Cover	X		
	Jester		Other	
	GroboCodeCoverrage		X	
	Hansel		Other	
C/C++	Code TEST	X		
	BullseyeCoverage	X		
	Testwell CTC++	X		
Java and C/C++	eXVantage	X	X	
	OCCF	X		

Table 4 provides guidelines for developers to select coverage testing tools. Overall, much research in the area of software coverage testing has been realized and used in industrial software production. We hope our work will contribute to more usage of tools to improve software testing and as well as it will helps to the new researchers, can choose a path to start their research accordingly.

Table 4:Tool Selection

Requirements	Tools
Real-time/low overhead	Dynamic, eXVantage
High coverage	Agitar, Parasoft Jtest
Multi-language support	PurifyPlus, Semantic Designs
Multi-platform (C++ only)	BullseyeCoverage, Semantic Designs

VII. REFERENCES

[1] R. Lingampally, A. Gupta, P. Jalote. "A Multipurpose Code Coverage Tool for Java," In Proceedings of the 40th Annual Hawaii International Conference on System Sciences, IEEE Computer Society, 261b, 2007.

[2]Zhu, H., Hall, P.A.V. and May, J.H.R. (1997) Software unit test coverage and adequacy. ACM Comput. Surv., 29, 366–427

[3] Myers, Glenford J., The art of software testing, New York :Wiley, c1979.

[4] Li, J.J. (2005) Prioritize Code for Testing to Improve Code Coverage of Complex Software. Proc. 16th IEEE International Symposium on Software Reliability Engineering, IEEE Computer Society.

[5] Li, J.J., Weiss, D. and Yee, H. (2006) Code-coverage guided prioritized test generation. Inf. Softw. Technol., 48, 1187–1198.

[6] Wong, W.E. and Li, J. (2005) An Integrated Solution for Testing and Analyzing Java Applications in an Industrial Setting, Proc. 12th Asia-Pacific Software Engineering Conference (APSEC'05). Vol. 00, 576–583. IEEE Computer

[7] **Mr. Shaik Khasim Saheb** working asAassistant professor ,CSE in Sreenidhi Inistitute of Science and Technology. He has completed M.Tech (CSE) in VIT University and his research area include Software Testing and Image Processing, Computer Networks, Information Security and Big Data. He has published 3 Papers in various International Journals.

[8] **Mr. Devavarapu Sreenivasarao** working as Assistant professor,CSE in Sreenidhi Inistitute of Science and Technology. He has completed M.Tech (CSE) in JNTUH, Hyderabad and his research interests include Software Testing ,Image Processing, Computer Networks, Information Security, Big Data and Cloud Computing. He has published 7 Papers in various International Journals.

[9] **Prof. T.V. Narayana Rao** working as Professor, CSE in Sreenidhi Inistitute of Science and Technology, Hyderabad. His research area includes Software Testing, Network Security and Image Processing, Big Data and Cloud Computing. He is member on editorial boards of many national and international journals and has published 99 Papers in various International Journals.

[10] **Mr. M Kiran Kumar** working as Assistant professor in Sreenidhi Inistitute of Science and Technology. He has completed M.Tech (CSE) in HCU, Hyderabad and his research area includes Software Testing and Image Processing, Computer Networks, Information Security and Big Data . He has published 3 Papers in various International Journals.