Comparing Bug Finding Tools for Java Open Source Software

Elmira Hassani Oskouei[1*], **Oya Kalıpsız**[1]

1 Yildiz Technical University, * Corresponding author, elmira.ha2006@gmail.com

Abstract

Software's are getting bigger and more complex and it is very important to improve defect-detection techniques. Software failure may have very critical consequences like economic loss. Using bug finding tools can reduce time and cost of testing software's. The importance of software testing process has caused developing of many tools to find bugs automatically in program source code in recent years. In this paper, we perform a comparison between different Java open source bug-finding tools over a wide variety of tasks. For our study, we used three well-known open source bug-finding tools which are PMD, FindBugs and Checkstyle. We ran these tools on a variety of open source Java programs and compare the results. Our results show that each of the tools can find different kind of bugs and there is no perfect tool that can be used instead of the other tools.

Keywords: PMD, Checkstyle, FindBugs, Java, Software testing tools.

Citation: Oskouei, E. H., Kalıpsız, O. (2018, October) *Comparing Bug Finding Tools for Java Open Source Software.* IMISC 2018 Conference Proceedings, 17-19.

Editor: H. Kemal İlter, Ankara Yıldırım Beyazıt University, Turkey

Received: August 19, 2018, **Accepted:** October 18, 2018, **Published:** November 10, 2018

Introduction

The importance of software test and quality has caused to developing many bug finder tools. Software's are getting bigger and more complex and it is very important to improve defect-detection techniques. Software failure may have very critical consequences like economic loss. Many researches have been done to help find bugs automatically (Malhotra, & Bahl, 2017) and easier by using static analysis bug finding tools. The deployment of an integrated environment for software testing tools is also important to increase businesses productivity in software development. The integrated environment is proposed to assist software-testing executions within projects on enterprises (Romano, De Souza, & Dacünha, 2015).

We address the question how bug finding tools can help to detect problems and if there is one tool that can use instead of all the tools to find the bugs. In addition, what type of defects can each tool detect and is there any common bug type that is found by all the tools.

After evaluating the results the below finding are listed.

- Each tool can find different type of defects and there are very few bugs that were found by all the three tools.
- There is no one tool that can be used instead of all the tools since each tool found different bugs.

In this study, three open source tools that are PMD, FindBugs and checkstyle are used after evaluating the available open source tools for Java programs. We ran these tools on four open source projects written in Java programs (Malhotra, 2015). All the defects that were found were classified into five different categories.

Method

To detect bugs in source code by static analysis bug finding tools are used (Manzoor, Munir, & Moayyed, 2012). These tools can sometimes help to detect very critical problems that can cause to failure of the software and they can reduce time and cost while developing the software.

In this section, the three tools that are used in the case study are described. These tools can analyze Java programs. All the open source programs that are used in this study are written with Java language and all three tools are published under an open source license. The basic properties of the tools are written in table 1.

Table 1. Basic properties of the tool.

Name	Version	Input	Technology	Plugin Availability	GUI Availability
PMD	4.0.14v2017	JavaCC	Syntax	Available	Unavailable
FindBugs	3.0.1v2015	Java Byte Code	Syntax, Dataflow	Available	Available
CheckStyle	7.6.0v2017	JavaCC	Syntax	Available	Unavailable

PMD

This is an open source code analyzer that can examine the Java source code (https://www.pmd.github.io). It uses a rule-based approach to analyze the source code and to indicate the possible bugs and mistakes like empty catch blocks, unused variables, copy and paste line of codes and etc. It is a static code analysis tool that indicates bugs without executing the code. PMD has many built-in rules to check the source code but it also allows writing rules so it can be used to check problems for specific environments. Additionally, it also allows users to execute custom analyses by

allowing them to develop new evaluative rules in a convenient manner. One of the reasons that make PMD quite popular is that it can be integrated with famous development environments such as Eclipse and it is also user-friendly. This tool is generally very effective and functional for both small and large set of codes.

FindBugs

This tool detects bugs by using a list of bug patterns (http://www.findbugs.sourceforge.net). FindBugs uses data flow and syntactic analysis to detect bugs. Static analysis of the byte code is how this tool can find bugs. It allows everyone to add new bug patterns so it is expandable. Like PMD, this tool can also be integrated with famous development environments such as eclipse.

Checkstyle

This tool checks the Java code according to a code standard like sun code conventions (http://www.checkstyle.sourceforge.net). The best thing about this tool is, it can be adjustable according to any coding standard. Its operation is based on validation rules. The latest versions of this code are able to identify class design problems, duplicated code, or bug patterns.

We want to give a quick overview of the four projects. The projects chosen are development projects from the telecommunications company Sahand Iran with various development efforts and sizes. All these projects were developed using the Java programming language and can be classified as web information systems as they all use HTML and web browsers as their user interface.

Case Study

We ran all the three tools on the projects with special care aim to get appropriate results as much as possible. We check each warning one by one in order to make sure if it is a real defect or not. To do this, each part of the code related to the warning was checked by experienced developer. We used four projects in this case study but for further details and more accurate results more projects and tests are necessary. Also, three tools were used for this study which can be expandable to more tools. For defect categorization, we used the standard categories (Wagner, Jurjens, Koller, & Trischberger, 2005) which are described in from 1 to 5. The defects in category 1 are the most critical and in category 5 the least critical. The categories are:

1. Defects that lead to a crash of the application.
2. Defects that cause a logical failure.
3. Defects with insufficient error handling.
4. Defects that violate the principles of structured programming
5. Defects that reduce the maintainability of the code.

Findings

In this section, we present all the results from the case study. Table 2 shows all the defect types and their categories found by the tools over the projects.

As shown on table 2, most of the warnings belong to category 5 (maintainability of the code). It is very clear that each tool finds different type of bugs and very few defects were detected by all the three tools. Checkstyle could not detect the defects that belong to the first category. PMD and Findbugs detect problems from all the categories.

Table 2. Defects found by the tools.

Bug Type	Category	PMD	FindBugs	Checkstyle
Division by zero	1	1	0	0
Database connection is not closed	1	3	1	0
Return value of function ignored	2	22	15	14
Exception caught but not handled	3	45	30	17
Null-pointer exception not handled	3	0	6	0
Concatenating string with + in loop	4	0	18	0
Checking equality using == orl =	4	10	3	0
For- instead of simple while loop	5	18	0	0
Parameter not used	5	44	0	0
Unused local variable	5	355	0	0
Variable initialized but not read	5	0	215	115
Needless if-clause	5	34	0	0
Stream not closed on all paths	5	0	33	0
Needless semicolon	5	45	0	0
Multiple functions with same name	5	0	24	0
Null dereference	5	33	40	0
Possible unexpected exception	5	45	24	0
Possible deadlock	5	23	14	0
Unreachable code due to constant guard	5	3	1	0
Private method not used	5	28	28	0
Empty finally block	5	1	0	0
Length may be less than zero	5	2	0	0
Should be a static inner class	5	20	26	15
Unnecessary return statement	5	215	189	0

The number of the bugs found by each tool is graphically shown on Fig. 1. Findbugs detect total of 15 types of bugs and PMD total of 19 types of bugs and Checkstyle total of 4 types of bugs.

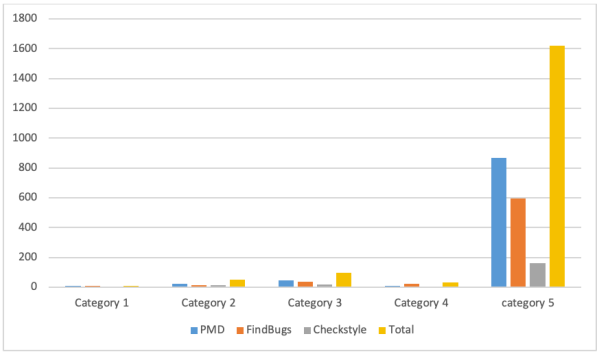

Figure 1. A graphical comparison of the number of defects found by each tool and in total.

Bug-finding Tools in Contrast with Review

On one of the projects an informal review was done by the developers of the project. The reviewers inspected the code in a review meeting. All of the defects disclosed by the review are listed in the table 3.

All the bugs found by the tools, were detected by the review too. In addition, there are some defects found which were not detected by the tools. However, PMD found 45 needless semicolons which were not found by the review. On the other hand, the review detected 5 defects of type “Length may be less than zero” whereas the tools only found 2. In addition, Findbugs found one “Unreachable code due to constant guard” but it was not recognized in the review.

Most of the logical defects or wrong results were not found by the tools. These types of faults are only found in the review by following the test cases and codes.

In summary, tools are not as successful as the review because the review is able to find logical and much more defects. Also, the type of the defects found by the review is much more than the tools. However, it is beneficial to use the tools at the begging to remove the defects that are in common because this process is much cheaper and faster.

Nevertheless, we notice a few number of false positive from the three tools and this results in a lot of work for the developers. This means, most of the tools need improvements to reduce these false positives as much as possible.

Table 3. Defects found by the review.

Defect Type	Category	Occurrences
Division by zero	1	1
Database connection is not closed	1	3
Inappropriate result	2	5
Needless if-clause	5	34
Wrong result	2	4
Database connection inside loop opened and closed	4	1
Wrong delete of Input	2	3
Leading zero	2	2
Data Range Inappropriate and not checked	2	5
Wrong Input	2	3
Length may be less than zero	2	2
Incomplete comparison	2	1
Negative conclusion	2	1
Other logical errors	2	6

Discussion and Conclusions

The work presented is a case study using some open source projects and bug finding tools to evaluate their performance and results in comparison to each other and also a review which was done on the same projects. The bug finding tools mostly are not able to verify the logic of the software therefore most of the defects found are related to category 5 that is related to maintainability of the code. These tools look for certain patterns and simple dataflow. On the other hand, there are defects that can only be revealed by review or test which are logical.

In summary, after evaluating the results, we can notice that there is no one perfect tool that can we use instead of all the other ones. Each tool can find different type of defects which are not in common. In addition, using tools is not efficient enough because there are many logical faults that are only found in review. This shows that both can be used together.

The main conclusion is that developers need to improve the bug finding tools and also try to reduce the false positive ratio so in this case; these tools can save costs and time while used together with other test techniques.

References

B. L. Romano, R. B. De Souza and A. M. Da cünha. (2015). Deploying integrated environment for software testing tools. 12th International Conference on Information Technology New Generations.

Numan Manzoor, Hussan Munir, and Misagh Moayyed. (2012). Comparison of static analysis tools for finding concurrency bugs. IEEE 23rd International Symposium on Software Reliability Engineering Workshops.

R. Malhotra and L. Bahl. (2017). A defect tracking tool for open source software. 2nd International Conference for Convergence in Technology (I2CT).

R. Malhotra, Empirical Research in Software Engineering, USA: CRC Press. (2015).

Stefan Wagner, Jan Jurjens, Claudia Koller, and Peter Trischberger. (2005). Comparing bug finding tools with reviews and tests. Unpublished.