IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

A Study on Different Tools for Code Smell Detection

S.James Benedict Felix.[1*], Viji Vinod[2]

[1]Computer Science, Research Scholar, Bharathiar University, Coimbatore, India
[2] Computer Applications, Prof & Head, Dr.MGR Educational and Research Institute University, Chennai, India

Available online at: www.ijcseonline.org

Accepted: 13/Jul/2018, Published: 31/Jul/2018

Abstract— Code and design smells are the poor result to recurring implementation and design problems. They may hinder the progress of a system by building it hard for software engineers to carry out transform. Detection of code smells is very challenging for code developers and their informal definition leads to the completion of detection techniques and tools. Several refactoring tools have been developed. A bad smell is a sign of some setback in the code, which requires refactoring to deal with. Various tools are offered for detection and deduction of these code smells. These tools are different significantly in detection methodologies.

Keywords— Code smell detection tools, Detection techniques, MobileMedia (MM), Health Watcher (HW)

I. INTRODUCTION

Once code smells are placed in a system they can be eliminated by refactoring the source code. Refactoring [1] is a technique to construct a computer program more readable and maintainable. This paper analyzed four code smell detection tools, namely JDeodorant, infusion, PMD, and JSpIRIT. These four detection tools were chosen because they evaluate Java programs, they can be setup and installed from the given downloaded files, they detect the smells in target systems. Other tools were discarded for various reasons. For example, Checkstyle has not detected instances of smells in any of the target systems, so it was discarded. Instead, it supports only visualization features.

Section I contains the introduction of code smell detection, Section II contains code smell detection tools and detection techniques, Section III contains the experimental studies, Section IV contains the related works of the code smell detection tools, and Section V concludes research work with future directions.

II. CODE SMELL DETECTION TOOLS

Table 1 shows the fundamental information about the evaluated tools [3]. The column Tool has the names of the analyzed tools as reported in the tools corresponding websites. The column Version indicates the version of the tools that were used in the experiments. The column Type specifies if the tool is available as a plugin for the Eclipse IDE or as a separate tool. Languages column have the programming languages that can be evaluated by the tools, with Java being the general language among them. The column Refactoring shows whether the tool offers the feature of refactoring the code smell detected, which is available only in JDeodorant. The column Export signifies if the tool allows exporting the results to a file, a feature present only in inFusion and JDeodorant that export the results in an HTML file and an XML file, respectively.

Table 1 Code smell detection tools

Tool	Version	Type	Langu-ages	Refactoring	Export	Detection Technique
JDeodorant	5.0.0 2015	Eclipse	Java	Yes	Yes	Refactoring Opportunities
Infusion	1.8.6.2015	Standalone	Java, C,C++	No	Yes	Software Metrics
PMD	5.3.0 2015	Eclipse	Java, C,C++	No	No	Software Metrics
JSpIRIT	1.0.0 2014	Eclipse	Java	No	No	Software Metrics

The tool JDeodorant2 is an open source Eclipse plugin for Java that detects four code smells: God Method, God Class, Feature Envy, and Switch Statement. A tool inFusion is a standalone tool for Java, C, and C++ that detects 22 code smells, including the three smells of our interest namely God Method, God Class, and Feature Envy. inFusion is no longer available for download at this moment, as a commercial product. The code smells detection techniques were primarily based on the detection strategies described by Lanza and Marinescu. The tool PMD3 is an open source tool for Java and an Eclipse plugin that detects many code smells including the God Class and God Method. The detection of code smells techniques are based on metrics. For God Class, the detection strategies of a single metric are used: LOC (lines of code). Finally, JSpIRIT4 is an Eclipse plugin for Java that identifies and prioritizes ten code smells, including the three code smells namely God Method, God Class, and Feature Envy.

III. EXPERIMENTAL STUDY

With the object-oriented [2] methodology, the experimental study analyzed by two Java systems namely MobileMedia and Health Watcher. This experimental study describes these two target systems because of the code smell experts responsible for analyzing the code to identify code smells. The manual identification of code smells is a complicated task. Therefore, intimate knowledge of the system and its domain make possible the comprehension of the source code. This allowed the experts to focus code smell instances instead of trying to understand the system, its dependencies, and other domain-related specificities. In this paper, other reasons for choosing the two systems: (i) Access to their source code, allowing us to manually retrieve code smells, (ii) their code is readable, facilitating for instance, the task of identifying the functionalities implemented by methods and classes, (iii) these systems were beforehand used in other maintainability-related studies.

3.1 MobileMedia (MM)

This system is a software product line (SPL) for applications. This system is manipulating the photo, audio, and video on mobile devices. Our study involved nine object-oriented versions (1 to 9) of MobileMedia, ranging from 1 to 3 KLOC. Table 2 shows for each version of MobileMedia the number of methods, classes, and lines of code [4]. This paper observes versions 1 to 9 there was an increase of 2057 lines of code, 166 methods, and 31 classes.

3.2 Health Watcher (HW)

This system is a typical Web-based information application that allows civilian to register complaints regarding health issues. It is a nontrivial and real system that uses technologies common in day-to-day software development, such as GUI, persistence, concurrency, RMI, Servlets, and JDBC (Greenwood et al. 2007). This paper analyse ten object-oriented versions (1 to 10) of Health Watcher, ranging from 5 KLOC to almost 9 KLOC. We can observe that from version 1 to version 10 there was an increase of 2706 lines of code, with the addition of 41 classes and 270 methods.

IV. COMPARATIVE STUDIES OF CODE SMELL DETECTION TOOLS

In this section summarizes the detection of the code smells in the two target systems using infusion, JDeodorant, JSpIRIT, and PMD tools.

4.1 MobileMedia (MM)

Table 2 listed the number of code smells identified by each tool in the nine versions of MobileMedia. For God Class, the tool JDeodorant reports the maximum number of classes, reporting 85 classes, and the other tools report less than 9. For God Method, the tool JDeodorant reports 100 methods. For God Method, JSpIRIT reports 27 God Methods, while infusion and PMD report similar numbers, 16 and 17, respectively. For Feature Envy, JSpIRIT reports the highest number of methods, reporting 74 methods, followed by JDeodorant reporting 69 methods. Lastly, inFusion reports only 9 instances of Feature Envy. Considering the total of smells reported, however, inFusion is the most conservative tool, with a total of 28 code smell instances for God Class, God Method, and Feature Envy. PMD is less conservative because it totally detects 24 instances of God Method and God Class. PMD does not detect Feature Envy. JDeodorant is the most aggressive and it is detecting totally 257 instances of various methods. That is, JDeodorant is the most conservative tools and detects more amounts of smells than inFusion and PMD. However, JSpIRIT is the tool that reports totally 110 code smells for the nine versions of the MobileMedia system.

4.2 Health Watcher (HW)

Table 2 shows the total number of code smell instances identified by every tool in the ten versions of Health Watcher. For God Method, JDeodorant is the most aggressive which reports 599 code smells. The other tool JSpIRIT reports fewer methods, with reporting 30 methods, PMD reporting 13, and infusion reporting none. PMD does not detect Feature Envy. For Feature Envy, JSpIRIT reported 111 methods, while Jdeodorant reported 90 and infusion reported 48. PMD is the second conservative tool, which is detecting a total number of 46 instances of God Class and God Method. Jdeodorant is the more aggressive tool, which is detecting a total number of 787 instances. That means, it detects 16 times the amount of smells of the tools infusion and PMD.

Table 2: Total number of code smell detection by each tool

Code Smell	**Infusion**		**Jdeodorant**		**JSpIRIT**		**PMD**	
	MM	HW	MM	HW	MM	HW	MM	HW
God Class	3	0	85	98	9	20	8	33
God Method	17	0	100	599	27	30	16	13
Feature Envy	8	48	69	90	74	111	----	----
Total	28	48	254	787	110	161	24	46

RELATED WORK

There are many papers analyzed code smell detection tools. A list of detection tools was proposed in a systematic literature review by Fernandes et al. [5]. Generally tools are evaluated individually and considering only a few smells. Fontana et al. were used six versions of a system to evaluate four tools, Checkstyle, inFusion, JDeodorant, and PMD. This paper analysed inFusion, JDeodorant, and PMD, calculating the agreement among these tools similarly to Fontana et al. Chatzigeorgiou and Manakos and Tufano et al. (2015) also analyzed multiple versions of systems to investigate the evolution of code smells.

Many papers proposed different approaches to detect code smells in software. Oizumi et al. (2016) proposed that code smells are related, appearing together in the source code to make different design problems. Another study by Fontana et al., (2015) applied 16 different machine-learning algorithms in 74 software systems to detect four code smells in an attempt to avoid some common problems of code smell detectors [6]. This paper extends our previous work by including the tool JSpIRIT and the Health Watcher system to increase the confidence of our results.

V. Conclusion

The comparison of code smell detection tools is a difficult task because these tools are based on informal definitions of the smells. The different interpretations of code smell by researchers and developers lead to tools with distinct detection techniques, results, and consequently, the amount of time spent with validation. In this paper, MobileMedia and Health Watcher are used as target systems, to evaluate the accuracy and the agreement of the tools inFusion, JDeodorant, JSpIRIT, and PMD. The accuracy was measured by calculating the recall and the precision of tools in detecting the code smells from the reference list.

For all smells in both systems, JDeodorant identified most of the correct entities but reports many false positives. A lower precision and a higher recall increase the validation effort but capture most the affected entities. On the other hand, inFusion, JSpIRIT, and PMD had higher precision, reporting more correct instances of smelly entities. A higher precision with a lower recall means that the tools do not report some of the affected entities.

References

[1] Altman DG (1991)," Practical statistics for medical research". Chapman & Hall, London

[2] Brown WJ, Malveau RC, Mowbray TJ, Wiley J (1998)," AntiPatterns: Refactoring software, architectures, and projects in crisis".Wiley

[3] Wiley Chatzigeorgiou A, Manakos A (2010) " Investigating the evolution of bad smells in object-oriented code". In: Proceedings of the 7th international conference on the quality of information and communications technology. IEEE, pp 106–115.

[4] DeMarco T (1979),"Structured analysis and system specification". Yourdon, New York

[5] Fernandes E, Oliveira J, Vale G, Paiva T, Figueiredo E (2016) ,"A review-based comparative study of bad smell detection tools". In: Proceedings of the 20th international conference on evaluation and assessment in software engineering (EASE'16).ACM, article18.

[6] Fontana FA, Mäntylä M, Zanoni M, Marino A (2015) Comparing and experimenting machine learning techniques for code smell detection. Empir Softw Eng 21(3):1143–1191. doi:10.1007/s10664-015-9378-4

[7] Amanda Damasceno,", On the evaluation of code smells and detection tools", Journal of Software engineering research and development. DOI 10.1186/s40411-017-0041-1. Oct. 2017.5:7