Experimental Study of Four Selective Code Smells Declining in Real Life Projects

Md. Erfan, Bohnishikhan Halder, Sathi Rani Pal, Md. Shariful Islam, and Rahat Hossain Faisal

***Abstract*—Different types of codes which may increase the liability of bugs or defects in future to a system known as Code smell. These type of smell can be eliminated without changing the external outcome and modifying the internal structure of the system. There are existing several well known code smell detection tools which automatically identify the code smells. The research used PMD automatic code smell detector to find four code smells on various open source java projects. It uses 43 open source java projects and identify the selective smells to these projects. The experiment shows duplicate codes, unused imports, unused local variables and unused private methods are not present for 79%, 34.9%, 51.1%, and 86.04% projects respectively. In the paper, it also shows that the probability of occurring duplicate codes, unused imports, unused local variables and unused private methods are respectively 4.5%, 71%, 20.4% and 4.1% which indicates the four selective code smells are declining in real life projects day by day.**

***Index Terms*—Code smell, refactoring, detector and PMD.**

I. Introduction

Code smell is a type of code decay [1], [2] which hinders a developer for future maintenance of the system [3], [4]. It is good practice for developers to avoid the code smells because of maintaining in future of the system after development. Those smells are something wrong in the code of the system but do not create any change to the outcome. According to Fowler [5], [6], There are 22 state-of -the-art code smells which are identified by the Fowler [5], [6].To eradicate those smells, there exists several types of automated code smell detectors. Some well-known detection tools named PMD, CheckStyle, Infusion and JDeodorant . Those tools make easy for helps developers to find code smells. The research work identified four code smells including duplicate codes, unused private methods, unused local variables and unused imports using PMD in different java projects.

Duplicate Code is same code structure remaining more than one place in the system [7]-[9]. Fowler described same expression in two methods of the same class and same expression in two siblings subclasses are common duplication problems. These duplicate codes isare created by exact duplication, renaming, aliasing or other kinds which are occuring during modified and mixed with different code [10]. Duplicate code is considered as an occurrence by PMD where a portion of code occurring more than once. ItThe duplicate code problem is arranged by at least 25 tokens to the PMD.

Example

```
import.javax.swing.JOptionPane;
import.javax.swing.*;
public void function()
{
}
```

Here JOptionPane is a part of swing but it is written separately and also called swing library function. This is a code smell known as duplicate code.

Unused import, an import from another module is never used [11]. It is a code smell which creates for some irresponsibility of developers.

Example

```
import.javax.swing.*;
int sub()
    {
        return(b-c);
     }
```

Unused Local variables is a code smell which creates maintenance burden ofto a system. It is an assigned variable or in-parameter which is not read before it becomes undefined. It appears due to spelling mistakes of local variables in the system.

Example

```
public int Add(int sum)
{
    int sub;//unused local variable
    return sum;
}
```

In this example, this function adds values but sub variables are not used in this function. These sub variables are unused local variables.

Unused Private method which is declared in the class but never executed. It is one type of dead code which is inoperative and unnecessary. Each function to the example is declared but not used in this system.

Example

```
public class food
{
   String Eat(){ // Unused method }
}
```

In hereHerein, Eat the method Eat() is declared but there is no any statementare no statements inside the method. That is just a method which is not used is called Unused Private method.

Eduardo [12], found 84 code smells detection tools. Among them 29 tools are available for online download. They It can detect code smells for different programming languages such as java, C, C++, C# and so on. Some

Manuscript received October 9, 2018; revised December 7, 2018.

The authors are with Computer Science and Engineering, University of Barisal, Barisal, Bangladesh (e-mail: sams.csebu@gmail.com, mostafij.csebu@gmail.com, irfan.bucse@gmail.com, bohnishikha46@gmail.com, sathibucse08@gmail.com, shariful.islam@gmail.com, rhfaisal@gmail.com).

doi: 10.18178/ijiee.2024.14.4.1

automated code smells detection tools PMD, CheckStyle, JDeodorant, inFusion and so on.

inFusion is a smell detection tool which is standalone tool [13]. It is possible to detect different type ofThis tool can detect22 code smells and also. It is used to detect code smells for C, C++ and Java languages.

JDeodorant [14], [15] is an open source Eclipse plug-in tool. It detects only fivea good number of code smells Feature Envy, Type Checking, Long Method, God Class and Duplicate code. The detection techniques are for God class, Feature Envy and slicing techniques, for God Method [16].

PMD is also an open source Eclipse plug-in detection tool. The automated detection tool is limited to only It is for java language. In this tool, Abstract Syntax Tree is used for detecting code smells.

CheckStyle [17] is a static analysis tool. It can compute different metrics. It is also Eclipse plug-in tool for java language.

In [13], the authors use three detection tools for finding code smells are inFusion, JDeodorant and PMD for finding three code smells God class, God method and Feature envy and comparing those tools with different open source projects. The author [10] reviews the current panorama of the tools for automatic code smell detection. Four representative code smell detectors applied to six different versions of GanttProject. Different detectors for the same smell do not meaningfully agree in their answers are suggested by the researcher.

In this research, four code smells duplicate codes, unused imports, unused local variables and unused private methods are analyzed. Those smells are identified by the automatic code smell detection tool named PMD. The PMD are executed to the IDE for 43a good number of open source java based open source projects. The research identifies the number of projects containing the selective smell. It also identifies the probability of occurring those smells to the projects. From our experimental analysis, it is ourthe final is that the selective four code smells are declining in real life projects day by day.

II. Related Work

Code smell is not any major problem in a system, a barrier for developers to properly understand the code. It increased the technical debt of the developers. For better readability and reduction of the debt from the developers, the smells need to be eradicated. There are many research works to identify the code smells from a system. There are also some code smell detection tools which automatically identify the smells in a code. There are several code smell detection tools named PMD, checkstyle, iPlasma and so on.

In the research work, [9] PMD code smell detection tool is used because of its actively development and maintenance. PMD is an open source code smell detector tool and used as command line, ant task, maven plug-in, IDE plug-in on Eclipse. The tool uses Abstract Syntax Tree for detecting bad smells form the code. In order to detect code smell, developers can directly run PMD on the code. It allows new rules from the users via XPATH rules or java based rules. Our targeted code smells are duplicate codes, unused imports, unused local variables and unused private methods detect automatically by the tool which is the main reason for selecting the tool. The tool detects the targeted smells quickly and efficiently for various open source projects.

The existing research works are either used existing smell detection tool or proposed a smell detection mechanism. State of the art research works are evaluated for open source projects with different version. In [13], three detection tools for finding code smells are inFusion, JDeodorant and PMD used to find three code smells God class, God method and Feature envy and comparing those tools with different open source projects. They These tools are used Mobile Media system to find the code smells and evaluate the result using 9 versions of a project. They want to more investigate more about others code smells and evaluated among them.

The author [1] identifies taxonomy for in six categories for 22 code smells. The research discusses about the procedure to absorb code smell by refactoring using Mantyla Survey. It derives the number of ways to remove selected smells but does not describe the way to identify in detail.

The researcher [10] reviews the current panorama of the tools for automatic code smell detection. Six different versions of open source java project, GanttProject are evaluated by four representative code smell detectors. Different detectors for the same smell does not meaningfully agree in their answers are their findings.

A smell detector is invented for both lab and industrial field by the research work [3]. It detects eleven types of code smells in both lab and industrial field. They compare with Checkstyle detector tool with their invented tool for a project to compare the results.

The author [18] proposed a mechanism called detection strategy. This mechanism is for helping developers and maintainers. Using this strategy, affected classes or methods can be directly localized. Even they tried to find 10 code smells in different projects.

Van Emden [19], using meta-model to detect code smell of software systems. Using this model, the author she detects code smells automatically and doing facial recognition in the software [20].

TABLE I: Specification For the Experiments

Specification	
Computers	Windows 10, Core i5, i7 with 8GB RAM.
IDE	Eclipse, Oxygen
Projects	Open source java projects
Smell Detector	PMD code smell detector

III. Experimental Setup

The experimental environment setup for this experiment are composed of automatic code smell detection tool PMD integrated with Eclipse, four laptops with different configuration and 43 open source java projects. PMD is an open source code smell detector tool which integrated to the Eclipse IDE for executing the open source java projects. These projects are open source and having different version. The selected projects are from small to very large size to properly analyze the experiments. As these projects are very large in size, different configuration of computer are used to

evaluate the result. The experiment is executed on the Eclipse Oxygen 64 bit version. The overall specification for the experiments are displayed in Table I.

IV. Evaluation and Finding

The objectives of the research are to examine that some code smells such as duplicate codes, unused imports, unused local variables and unused private methods are declining day by day in the real life projects. In order to fulfill our objectives, the research work uses PMD automatic code smell detection tool. The smell detection tool is integrated to the programming IDE Eclipse. It collects 43 open source java projects having different versions and sizes. The results for the overall projects are displayed in Table II.

TABLE II: Percentage of Projects without Smell

Name of Code smell	Total Project	No. of projects without smell	No. of smelly projects	Percentage of projects without smell (%)
Duplicate codes(DC)	43	34	9	79
Unused Imports(UI)	43	15	28	34.9
Unused local variables(ULV)	43	22	21	51.1
Unused private methods(UPM)	43	37	6	86.04

In the paper, four code smells are duplicate codes, unused imports, unused local variable and unused private methods information are collected. The collected information are gathered from the PMD detector tool executed to the IDE. The number of projects without containing any smells for DC, UI, ULV and UPM are 34, 15, 22 and 37 respectively. The percentage shown in Figure 1. of not occurring the selective smells to the projects are 79%, 34.9%, 51.1% and 86.4%.

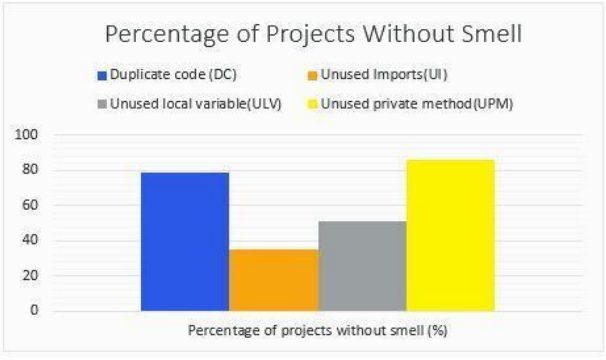

Fig. 1. Figure for without smelly projects.

The research work also identifies the probability of appearing DC, UI, ULV, UPM smells to the real life projects. The experimental data shows that the probability of occurring those smells are 4.5%, 71%, 20.4% and 4.1%. The lower probability value means that these types of smell have lower probability of occurring in the projects. The higher value of the probability indicates that the smells are appeared for several projects and have occurred several times in a project. The probability of occurring those selective code smells in 43 open source java projects are shown in the following Table III.

TABLE III: The Chance of Occurring Smell to the Project

Name of Code smell	Total Project	No. of projects without smell
Duplicate codes(DC)	43	4.5
Unused Imports(UI)	675	71
Unused local variables(ULV)	195	20.4
Unused private methods(UPM)	39	4.1

Unused Imports code smells are mostly occurred to most of the projects. Most of the projects contains the smell and increasing the probability. The DC and UPM code smells are not exist to the projects and have little amount of numbers. The overall findings for DC, UI, ULV and UPM are displayed in Fig. 2.

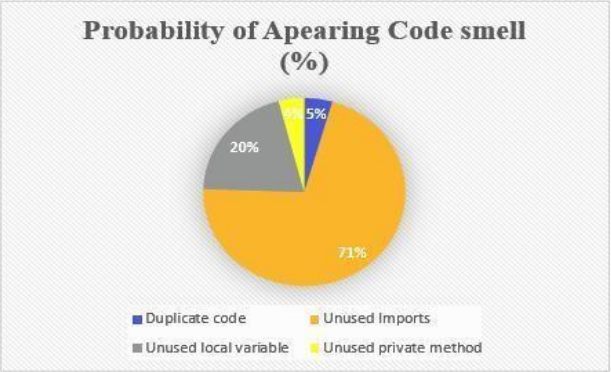

Fig. 2. Figure of the chance of probability of occurring smells in a project.

V. Conclusion and Future Work

The research identifies duplicate codes, unused imports, unused local variables and unused private methods using PMD on Eclipse IDE. The research examines the chance of occurring those smells on 43 real life open source projects. After calculating the percentage we find the results which are duplicate code 79%, unused private method 86.4%, unused import 51.1% and unused local variable 34.9% which are big amount as compare with other code smells. From our observation we conclude that the selective smells are declining for the real life projects day by day. The main criteria for this paper is showing the development of coding skills among developers day by day.

In future, we want to work using different types of code smell detectors and we will be trying to show that how vary the using of code smell in different countries. In that paper, we will discuss about some more code smells. We will select some projects and classify them into country wise. After selecting them, we will check them by different detectors and calculated the result based on countries. Now a days, developers are more aware of their developing source codes and it is developing day by day in developing countries for this reason we are interested in performing this experiment.

Acknowledgment

The authors would like to thank F. A. Md. Samsuddoha and F. B. Md. Mostafijur Rahman lecturer at University of

Barisal.

REFERENCES

[1] S. Counsell, H. Hamza, and R. M. Hierons, "An empirical investigation of code smell „deception‟ and research contextualisation through paul‟s criteria," *CIT. Journal of Computing and Information Technology*, vol. 18, no. 4, pp. 333-340, 2010.

[2] Y. Mehta, *Analyzing Code Smell Removal Sequences for Enhanced Software Maintainability*, Doctoral dissertation, National Institute of Technology Jalandhar, 2017.

[3] X. Liu and C. Zhang, *DT: A Detection Tool to Automatically Detect Code Smell in Software Project*, 2017.

[4] W. Li and R. Shatnawi, "An empirical study of the bad smells and class error probability in the post-release object-oriented system evolution," *Journal of Systems and Software*, vol. 80, no. 7, pp. 1120-1128, 2007.

[5] M. Fowler and K. Beck, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley Professional, 1999.

[6] E. V. Emden and L. Moonen, "Java quality assurance by detecting code smells," in *Proc. Ninth Working Conference on Reverse Engineering*, 2002, pp. 97-106, IEEE.

[7] M. V. Mantyla, J. Vanhanen, and C. Lassenius, "Bad smells-humans as code critics," in *Proc. 20th IEEE International Conference on Software Maintenance*, 2004, pp. 399-408, IEEE.

[8] R. Kumar, J. Singh, and A. Kaur, *An Empirical Study of Bad Smell in Code on Maintenance Effort*, 2016.

[9] V. Lelli, A. Blouin, B. Baudry, F. Coulon, and O. Beaudoux, "Automatic detection of GUI design smells: The case of blob listener," in *Proc. the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2016, June, pp. 263-274, ACM.

[10] F. A. Fontana, P. Braione, and M. Zanoni, "Automatic detection of bad smells in code: An experimental assessment," *Journal of Object Technology*, vol. 11, no. 2, pp. 1-38, 2012.

[11] H. Neukirchen and M. Bisanz, "Utilising code smells to detect quality problems in TTCN-3 test suites," *Testing of Software and Communicating Systems*, pp. 228-243, 2007.

[12] E. Fernandes, J. Oliveira, G. Vale, T. Paiva, and E. Figueiredo, "A review-based comparative study of bad smell detection tools," in *Proc. the 20th International Conference on Evaluation and Assessment in Software Engineering*, 2016, June, p. 18, ACM.

[13] T. Paiva, A. Damasceno, J. Padilha, E. Figueiredo, and C. Sant‟Anna, "Experimental evaluation of code smell detection tools," 2015.

[14] N. Tsantalis, T. Chaikalis, and A. Chatzigeorgiou, "JDeodorant: Identification and removal of type-checking bad smells," in *Proc. 2008 12th European Conference on Software Maintenance and Reengineering, CSMR 2008*, April 2008, pp. 329-331, IEEE.

[15] M. Fokaefs, N. Tsantalis, and A. Chatzigeorgiou, (2007, October). "Jdeodorant: Identification and removal of feature envy bad smells," in *Proc. IEEE International Conference on Software Maintenance, ICSM 2007*, 2007, pp. 519-520, IEEE.

[16] E. Figueiredo, N. Cacho, C. Sant'Anna, M. Monteiro, U. Kulesza, A. Garcia, and F. Dantas, "Evolving software product lines with aspects: an empirical study on design stability," in *Proc. the 30th International Conference on Software Engineering*, May 2008, pp. 261-270, ACM.

[17] S. Jancke and D. Speicher, "Smell detection in context," University of Bonn, 2010.

[18] R. Marinescu, "Detection strategies: Metrics-based rules for detecting design flaws," in *Proc. 20th IEEE International Conference on Software Maintenance*, September 2004, pp. 350-359, IEEE.

[19] E. V. Emden and L. Moonen, "Java quality assurance by detecting code smells," in *Proc. Ninth Working Conference on Reverse Engineering*, 2002, pp. 97-106, IEEE.

[20] J. Schumacher, N. Zazworka, F. Shull, C. Seaman, and M. Shaw, "Building empirical support for automated code smell detection," in *Proc. the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, September 2010, p. 8, ACM.

Md. Erfan is a lecturer of computer science and engineering in University of Barisal. He studies in the Institute of Information Technology (IIT), University of Dhaka. City: Barisal, Country: Bangladesh

Bohnishikha Halder is a student from University of Barisal. Date of Birth: 27/09/1995. She studies in Computer Science and Engineering from University of Barisal. City: Barisal, Country: Bangladesh

Sathi Rani Pal is a student from University of Barisal. Date of Birth: 19/04/1997. She studies in Computer Science and Engineering from University of Barisal. City: Barisal, Country: Bangladesh Bangladesh.

MD. Shariful Islam received his BS and MS in computer science from the University of Dhaka, Bangladesh.He obtained his PhD degree in wireless networking from the Department of the Computer Engineering, School of Electronics and Information, Kyung Hee University, South Korea in 2011. He is now working as a professor in the Institute of Information Technology (IIT), University of Dhaka, Bangladesh. He has been teaching a good number of courses related to computer networks, wireless and mobile systems, security, information technology project management, etc. to graduate and undergraduate students of reputed universities. He has research interests in wireless networking, wireless mesh networks, Information security, cloud computing, etc. He has published a good number of research papers in international conferences and journals.

Rahat Hossain Faisal is an assistant professor and the Chairman of computer science and engineering in University of Barisal. City: Barisal, Country: Bangladesh