An Empirical Study on the Occurrences of Code Smells in Open Source and Industrial Projects

Md. Masudur Rahman*
Institute of Information Technology, University of Dhaka
Dhaka, Bangladesh
bit0413@iit.du.ac.bd

Abdus Satter
Institute of Information Technology, University of Dhaka
Dhaka, Bangladesh
abdus.satter@iit.du.ac.bd

Md. Mahbubul Alam Joarder
Institute of Information Technology, University of Dhaka
Dhaka, Bangladesh
joarder@iit.du.ac.bd

Kazi Sakib
Institute of Information Technology, University of Dhaka
Dhaka, Bangladesh
sakib@iit.du.ac.bd

ABSTRACT

Background: Reusing source code containing code smells can induce significant amount of maintenance time and cost. A list of code smells has been identified in the literature and developers are encouraged to avoid the smells from the very beginning while writing new code or reusing existing code, and it increases time and cost to identify and refactor the code after the development of a system. Again, remembering a long list of smells is difficult specially for the new developers. Besides, two different types of software development environment - open source and industry, might have an effect on the occurrences of code smells. **Aims:** A study on the occurrences of code smells in open source and industrial systems can provide insights about the most frequently occurring smells in each type of software system. The insights can make developers aware of the most frequent occurring smells, and researchers to focus on the improvement and innovation of automatic refactoring tools or techniques for the smells on priority basis. **Method:** We have conducted a study on 40 large scale Java systems, where 25 are open source and 15 are industrial systems, for 18 code smells. **Results:** The results show that 6 smells have not occurred in any system, and 12 smells have occurred 21,182 times in total where 60.66% in the open source systems and 39.34% in the industrial systems. *Long Method, Complex Class* and *Long Parameter List* have been seen as frequently occurring code smells. The one tailed t-test with 5% level of significant analysis has shown that there is no difference between the occurrences of 10 code smells in industrial and open source systems, and 2 smells are occurred more frequently in open source systems than industrial systems. **Conclusions:** Our findings conclude that all smells do not occur at the same frequency and some smells are very frequent. The short list of most frequently occurred smells can help developers to write or reuse source code carefully without inducing the smells from the beginning during software development. Our study also concludes that industry and open source environments do not have significant impact on the occurrences of code smells.

CCS CONCEPTS

• **Software and its engineering**; • **Software systems**; • **Code smells**;

KEYWORDS

code smell, open source system, industrial system, empirical study

ACM Reference Format:
Md. Masudur Rahman, Abdus Satter, Md. Mahbubul Alam Joarder, and Kazi Sakib. 2022. An Empirical Study on the Occurrences of Code Smells in Open Source and Industrial Projects . In *ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (ESEM '22), September 19–23, 2022, Helsinki, Finland.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3544902.3546634

1 INTRODUCTION

Code smells are not programming errors or bugs, but these are the symptoms that affect code comprehensibility and maintainability [9]. Moreover, the presence of code smells indicates the probability of having bugs in future. Developers usually introduce code smells due to less awareness in design and program comprehensibility [2]. This situation of code smell occurrences is correct for both categories of software development environment - industrial (also known as in-house) and open source systems (or projects). For industrial environment, systems are developed with release pressure, time and budget constraints, same development strategies for the developers, etc. On the other hand, for open source environment, there exist little release pressure, time and budget constraints, and developers have their own development strategies. Therefore, due to these different development environments, there may have different occurrences of code smells in those systems.

Nowadays, software development is performed by reusing source code with little modification if required to reduce development time and cost. Source code can be reused from open source systems and previously developed industrial systems (if allowed). However, if developers are not aware of the presence of code smells in the code to be reused, such reuse might spread these smells across the systems, and so increase more maintenance time and cost. So, developers from both industrial and open source environment should be enlightened with the statistics of the presence of the smells in the

ESEM 2022, Sep. 19–23, 2022, Helsinki, Finland

ACM ISBN 978-1-4503-9427-7/22/09…$15.00
https://doi.org/10.1145/3544902.3546634

existing source code, specially the frequently occurring smells. For example, if a developer wishes to reuse code from open source systems (or industrial) and knows the probable frequently occurring smells in those systems in advance, s/he can refactor those smells (if applicable) before reusing the code in his/her system. Such type of activity will make the system more maintainable in the long run. In addition, it is very difficult for developers to write code considering all possible code smells. Hence, a list of frequently occurring code smells might help them to focus and be careful about only those smells while developing systems. As a result, it might be helpful to reduce the smells from the systems at the time of development. Therefore, it is important to find out which smells occur more frequently in which environments - industrial and/or open source, to make awareness about those smells among the developers of both environments.

In order to conduct the analysis, an empirical study has been carried out on 40 Java systems, where 25 of them are popular open source systems and rest 15 are industrial systems. Open source systems have been selected based on the popularity, number of stars and contributors. Each industrial system has been developed following standard SDLC model and are currently used by clients. A list of 18 code smells has been selected for the investigation. An existing tool named DECOR [5] has been used for smell detection, because the tool shows good accuracy and has been used in the earlier studies [3, 9, 10]. To the best of our knowledge, this is the largest study aimed at analysing the frequency of code smell occurrences in both open source and industrial systems.

The results of the study show that 12 smells have been found in both open source and industrial systems. However, 6 smells - *Base Class Knows Derived Class, Functional Decomposition, Large Class, Message Chain, Swiss Army Knife and Traditional Breaker*, have not been seen in any systems. A total number of 21,182 occurrences of the 12 smells have been found where 39.34% are in the industrial systems and 60.66% are in the open source systems. Among the 12 smells, *Long Method* have been seen the most frequent smells as its occurrences are 37.46% and 38.07% of the total occurrences in the industrial and open source systems respectively. The next most prevalent smells are *Complex Class* and *Long Parameter List* as the occurrences of the first one are 36.59% and 36.46% of total occurrences of the smells in industrial and open source systems respectively. The presence of the later one is 19.04% of the total occurrences in the industrial systems and 11.62% in the open source systems. For every 41 methods in the industrial systems and 45 methods in the open source systems, one method is found as *Long Method*. Besides, one of every 6 classes in the open source systems is *Complex Class*, and one of every 8 classes in the industrial systems has been found as *Complex Class*. One method has been seen containing *Long Parameter* smell out of every 80 and 149 methods in industrial and open source systems, respectively. Other smells have not been seen so prevalent. One tailed t-test with 5% level of significance has shown that there is no difference between the occurrences of the smells in the industrial and open source systems except two smells - *Base Class Should Be Abstract* and *Refused Parent Bequest*. These smells are found more in open source systems than industrial systems as supported by the t-test.

2 RELATED WORK

Lots of works have been performed after defining 22 types of code smells by Fowler [2]. Most of the works have covered the detection strategies of the smells, and their impact on software systems. A very little work have been found about frequencies of smells in large scale in the literature.

JDeodorant [13], an Eclipse plug-in identifies six types of code smells, namely Feature Envy, Type Checking, Long Method, God Class, Duplicate Code and Refused Bequest and provides refactoring suggestions using static code analysis. Moha *et al.* [5] proposed a tool, namely DECOR that can identify 18 types of smells using rule card. Since it can identify a large number of smells, we have used this tool for this empirical study. Palomba *et al.* [8] used historical information of software releases to detect code smells.

Several studies have performed impact analysis of code smells on software maintainability using open source software systems [3, 9, 12, 14]. Yamashita *et al.* [15] showed inter-smell relations considering coupled smells on three systems. They incorporated static dependency analysis to investigate coupled and collocated smells. In another study, Palomba *et al.* [7] analysed the co-occurrence of code smells on the same code component thorough a large scale empirical study. They showed that almost 59% of smelly classes contain more than one type of code smells. Chatzigeorgiou *et al.* [1] investigates the presence and evolution of four types of code smells, namely Long Method, Feature Envy, State Checking and God Class using two source systems. Murphy-Hill *et al.* [6] showed that refactorings are performed frequently and almost 90% are manually, after conducting an extensive study using four data sets.

However, there is a noticeable lack of knowledge about the occurrences of code smell in industrial systems. Moreover, to the best of our knowledge, there exists no such empirical work regarding the comparative analysis of code smell occurrences between open source and industrial systems. To mitigate this research gap, the inclusion of industrial systems along with the open sources will provide a new dimension in the research area of code smell and refactoring.

3 EMPIRICAL STUDY DESIGN

The goal of the study is to identify the frequently occurring code smells in both types of software systems - open source and industrial, and compare the results between them, with the purpose of having insightful information about the smell occurrences.

3.1 Formulating the Research Goal

This study aims at investigating the presence of different code smells in open source and industrial systems by answering the following research questions:

RQ1 [Code smells in open source and industrial systems]: ***What types of code smells do occur in open source and industrial systems respectively?***

Developers can reuse source code from two sources - industrial systems and open source systems. Usually, developers follow standard software development process to develop industrial systems. On the other hand, usually, development process of most of the open source systems does not force developers to follow such rigid

steps like industrial system development. Due to the different nature of the two categories of systems and their development process, it is necessary for developers to know the types of code smells occur in each category of the systems independently. The answer to the question will help them to know whether they need to deal differently for each of type of systems while reusing source code.

RQ2 [Frequent code smells]: ***Which types of code smells do frequently occur in open source and industrial systems***

Literature contains many code smells, but not all of the smells occur frequently and are important for software development. Answering the question can help researchers and practitioners to know the frequently occurring code smells in both open source and industrial systems individually. The insights will motivate them to develop and improve refactoring tools and techniques for the smells on priority basis.

RQ3 [Differences between the occurrences of code smells in open source and industrial systems]: ***Are there any differences between the occurrences of the code smells in industrial and open source systems?***

The ideology of industrial development environment is to do business with the developed systems or earn money by delivering software meeting client requirements. On the contrary, open source systems are developed as contributions to the user and developer community without any business intentions. In addition, the environment, organizational setup and operations also differ from one another. The question aims to find whether the contrast of motive and operational process between industrial and open source development process has any impact on the occurrences of code smells.

3.2 Systems under Study

The context of the study consists of (i) software systems from both open sources and industries and (ii) code smells. In order to conduct the empirical study and answer the RQs, we have analyzed 40 large scale Java systems as our dataset: 25 well-known open source systems and 15 industrial systems. Both categories of the systems are large based on their size and development time. The average development time span for the open source systems is around 6.6 years, whereas for the industrial is 6 years. The industrial systems have been collected from six different software industries in four countries around the world. The description of the dataset are shown in Table 1. It is noted that, source code of the industrial systems studied in this research cannot be disclosed, as these are confidential to the companies. We provided the smell detection tool, DECOR [5] and three size-metric (Table 1) calculator tool, SourceMonitor[1] that produced the results to the corresponding industry personnel and they gave us the results without disclosing the system information, like - system, class and method name, system architecture etc. We also used the tool to calculate the metrics for the open source systems.

Table 1: Description of the dataset in the study

Category of Systems	Systems #	KLOC		NOM		NOC	
		Total	Average	Total	Average	Total	Average
Open Source	25	3276	131	222151	8886	36228	1449
Industrial (In-house)	15	2093	140	127631	8509	18479	1232

[1]https://www.derpaul.net/SourceMonitor/

3.3 Detection of the Code Smells

To carry out the investigation, a list of 18 code smells has been selected as shown in Table 2 (first column). A smell detection tool, DECOR [5] has been selected to identify the occurrences of each code smell in the systems. Source code of the only open source systems and results obtained through running the tools on the open source and the industrial systems have been given in the link[2].

4 RESULT ANALYSIS

To answer **RQ1**, the occurrences of code smells have been analyzed. The presence of code smells across the systems in the study along with frequencies has been studied to answer **RQ2**. The differences between the occurrences of the code smells in industrial and open source systems have been reported with t-test results in the answer of **RQ3**.

4.1 RQ1 [Code smells in open source and industrial systems]

According to the results of the study as shown in Table 2, out of the 18 code smells, 6 smells have not been found in both open source and industrial systems. These smells are *Base Class Knows Derived Class, Functional Decomposition, Large Class, Message Chain, Swiss Army Knife* and *Traditional Breaker*. Rest 12 smells have been seen present in both categories of the systems with different extents. The finding indicates that developers should be more careful on these smells, and researchers should give more emphasis on the improvement and invention of detection and refactoring techniques for the smells. To get an idea about the extent of the presence of the smells, the answer of **RQ2** describes the frequency of the occurrences of these smells from different granular levels.

4.2 RQ2 [Frequent code smells]

To understand what types of code smells frequently occurs, the spreadity for each smell that appears at least once in any of the systems in the study. Spreadity is a metric that defines the existence of a smell with respect to the total number of systems. The metric is calculated using the following equation.

$$spreadity(x) = \frac{n(x)}{N} \tag{1}$$

Here, x = code smell, $n(x)$ = number of systems containing smell x and N = total number of systems. Higher value of the metric for a smell indicates higher number of systems contain the smells. The maximum value will be 1 if all the systems contain the smell and it will be zero if none of the systems contain it.

Figure 1 depicts the spreadity for the 12 smells and rest 6 smells are intentionally omitted from the figure, since they have not occurred in any of the systems and the aim of the RQ2 is to find the most frequent smells. *Complex Class* and *Long Method* have been found in all the systems of both the open source and industrial having spreadity 1. *Long Parameter List* has been found in all the open source systems, but 93% of the industrial systems contain the smell. *Many Field Parent Attributes but Not Complex, Refused Parent Bequest, Spaghetti Code* and *Speculative Generality* seem less frequent in both types of systems. Another important observation

[2]https://tinyurl.com/mrtbc2xb

Table 2: Comparative result analysis between industrial and open source systems

Code Smell (S_i)	Industrial system						Open Source system						p-value
	Total Smell Frequency (fs_i)	Average Smell Frequency (Per System)	*Spreadity* = System# having S_i / Total System#	$ISF_i(LOC)$ *Score* = $LOC / (fs_i)$	$ISF_i(NOM)$ *Score* = $NOM / (fs_i)$	$ISF_i(NOC)$ *Score* = $NOC / (fs_i)$	Total Smell Frequency (fs_i)	Average Smell Frequency (Per System)	*Spreadity* = System# having S_i / Total System#	$ISF_i(LOC)$ *Score* = $LOC / (fs_i)$	$ISF_i(NOM)$ *Score* = $NOM / (fs_i)$	$ISF_i(NOC)$ *Score* = $NOC / (fs_i)$	
Anti Singleton (AS)	86	5.73	0.67	24336	1484	215	213	7.61	0.8	15380	1043	170	0.114
Base Class Knows Derived Class (BCKDC)	0	0.00	0.00	NA	NA	NA	0	0.00	0	NA	NA	NA	NA
Base Class Should Be Abstract (BCSBA)	5	0.33	0.13	418578	25526	3696	59	2.11	0.52	55524	3765	614	**0.008**
Blob	82	5.47	0.47	25523	1556	225	72	2.57	0.72	45499	3085	503	0.460
Class Data Should Be Private (CDSBP)	238	15.87	0.93	8794	536	78	783	27.96	0.96	4184	284	46	0.382
Complex Class (CC)	3049	203.27	1.00	686	42	6	4685	167.32	1	699	47	8	0.160
Functional Decomposition (FD)	0	0.00	0.00	NA	NA	NA	0	0.00	0	NA	NA	NA	NA
Large Class (LC)	0	0.00	0.00	NA	NA	NA	0	0.00	0	NA	NA	NA	NA
Lazy Class (LzC)	141	9.40	0.60	14843	905	131	581	20.75	0.84	5638	382	62	0.222
Long Method (LM)	3122	208.13	1.00	670	41	6	4892	174.71	1	670	45	7	0.080
Long Parameter List (LPL)	1587	105.80	0.93	1319	80	12	1493	53.32	1	2194	149	24	0.286
Many Field Attributes But Not Complex (MFABNC)	5	0.33	0.20	418578	25526	3696	13	0.46	0.2	251992	17089	2787	0.211
Message Chains (MC)	0	0.00	0.00	NA	NA	NA	0	0.00	0	NA	NA	NA	NA
Refused Parent Bequest (RPB)	7	0.47	0.07	298985	18233	2640	26	0.93	0.36	125996	8544	1393	**0.016**
Spaghetti Code (SC)	7	0.47	0.13	298985	18233	2640	21	0.75	0.4	155995	10579	1725	0.110
Speculative Generality (SG)	4	0.27	0.13	523223	31908	4620	11	0.39	0.28	297808	20196	3293	0.420
Swiss Army Knife (SAN)	0	0.00	0.00	NA	NA	NA	0	0.00	0	NA	NA	NA	NA
Tradition Breaker (TB)	0	0.00	0.00	NA	NA	NA	0	0.00	0	NA	NA	NA	NA
Total	**8333**	**555.53**	**1.00**	**251**	**15**	**2**	**12849**	**458.89**	**1**	**255**	**17**	**3**	**0.142**

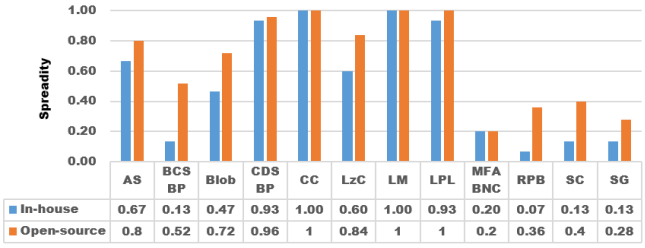

Figure 1: Spreadity of each code smell

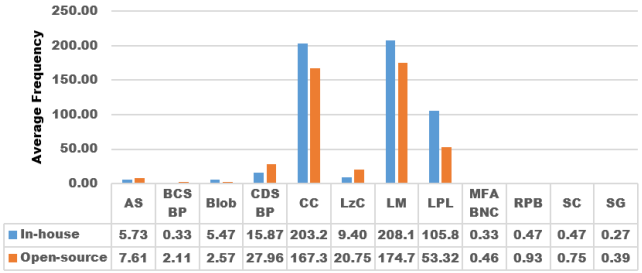

Figure 2: Average frequency of each code smell

is that the spreadity of the code smells are comparatively less in the industrial systems than open source ones.

The analysis is further extended to investigate the frequency of the occurrences of the smells in open source and industrial systems. Figure 2 depicts the average frequency of the occurrences of each smell per industry and open source system. Like spreadity, *Long Method* tops with the frequency of 208.13 per industrial system and 174.71 per open source system. *Complex Class* has followed the smell with the second highest appearances, 203.27 and 167.32 per industrial and open source system, respectively. *Many Field Parent Attributes but Not Complex, Refused Parent Bequest, Spaghetti Code* and *Speculative Generality* are less frequent both in open source and industrial systems which is also supported in Figure 2. Although the spreadity is comparatively higher in the open source systems as shown in Figure 1, the average frequency seems higher in the industrial systems than the open sources as shown in Figure 2.

The size of the systems under the study differs from one another. A large system is more likely have more code smells than smaller ones. So, the frequency of the occurrences of the smells have been further studied using a normalized metric - *inverse smell frequency*, $ISF_i(metric)$ score for a particular smell i in terms of a software

metric (Equation 2). We have inspired to use this metric from the Information Retrieval technique, *inverse document frequency, idf* calculation [4].

$$ISF_i(metric) = \frac{metric\ value}{smell\ count_i} \quad (2)$$

Here, *metric value* - a particular software metric value, such as - LOC, NOM, NOC, etc. *smell count*$_i$ - is for a particular code smell i, such as - Long Method smell count, Blob smell count, etc.

From this metric, we have derived more three metrics for the study - inverse smell frequency (for a particular smell i) in terms of Line of Code ($ISF_i(LOC)$), Number of Methods ($ISF_i(NOM)$) and Number of Classes ($ISF_i(NOC)$), respectively. $ISF_i(LOC)$ defines the average number of lines of code for which a smell occurs. $ISF_i(NOM)$ dictates the average number of methods for which a smell occurs. $ISF_i(NOC)$ asserts the average number of classes for which a smell is found. The lower the value of the metrics for a smell indicates the more frequent of the occurrence of the smell. The metrics can also assist to reveal after how many LOC, NOM and NOC a particular smell can occur. This insights can enlightened the developers with more clearer view about the frequency of each type of the code smells.

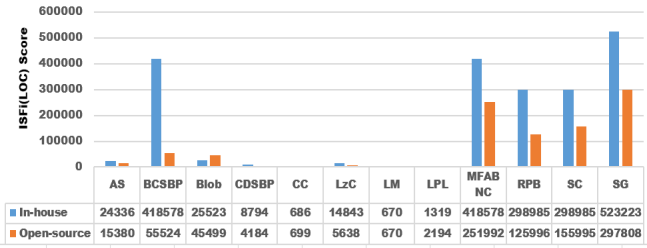

Figure 3: $ISF_i(LOC)$ **score of each code smell**

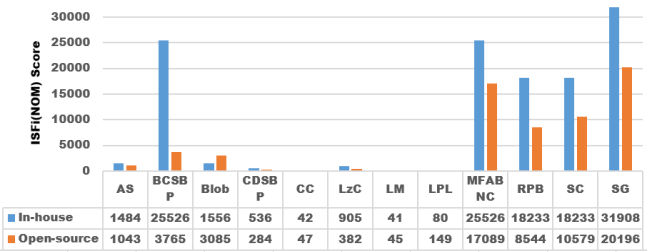

Figure 4: $ISF_i(NOM)$ **score of each code smell**

Figure 3, 4 and 5 illustrate the score of $ISF_i(LOC)$, $ISF_i(NOM)$ and $ISF_i(NOC)$ for all the smells under the study, except the six smells which have not been appeared in any system as described in the answer of **RQ1**. According to the figure, for every 670 line of code in the industrial systems and also in the open source systems, *Long Method* occurs. Again, one *Long Method* is found for every 41 methods in the industrial systems and 45 methods in the open source systems. In terms of number of classes, for every 6 classes in the industrial systems and 7 classes in the open source systems, one *Long Method* is seen. The second most frequent code smell is *Complex Class* as for every 686 lines of code in the industrial

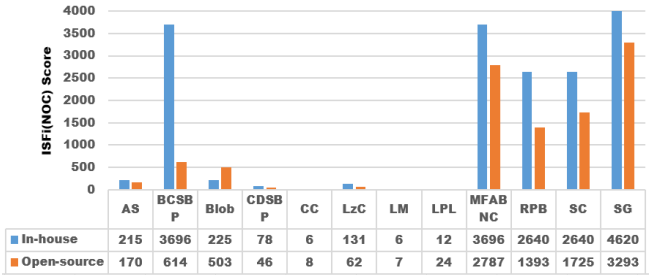

Figure 5: $ISF_i(NOC)$ **score of each code smell**

systems and 699 lines of code in the open source systems, the smell prevails. In fact, the smell is so frequent that for every 6 classes in the industrial systems and 8 classes in the open source systems, one class is found as *Complex Class*. Third position is occupied by *Long Parameter List* as one method is found to have the smell in every 80 methods in the industrial systems and 149 methods in the open source systems, respectively. *Lazy Class* and *Anti Singleton* are quite frequent as they occur 141 and 86 times in the industrial system, respectively, and 581 and 213 times in the open source systems, respectively. Besides, one of every 907 classes in the industrial systems and 284 classes in the open source systems has been identified as *Lazy Class*. Other smells are not so prevalent as the observations of the study highlighted in Table 2.

4.3 RQ3 [Differences between the occurrences of code smells in open source and industrial systems]

It has been seen in Figure 1 that the spreadity of the smells is more in the open source systems than the industrial systems. Conversely, Figure 2 illustrates that the smells are more frequent in the industrial systems than the open source systems. This research question aims to identify whether the occurrences of code smells differ between open source and industrial systems.

A statistical hypothesis testing, one tailed t-test with 5% level of significance has been conducted where the null hypothesis is - there is no difference between the occurrences of a particular code smell in open source and industrial systems. The alternative hypothesis is - the occurrences of a particular smell in open source systems is greater than in industrial systems. The test has been carried out for each smell for which at least one existence has been found in any of the systems in the study. For a particular smell, frequency of the smell has been normalized by dividing the total occurrences of the smell in a system with the total number of line of code of the system. The normalized data have been divided into two groups - one group contains the normalized system-wise frequencies for the open source systems and another group contains the normalized frequencies for the industrial systems. The results of the t-test has been shown in Table 2. The column p-value depicts the results for each of the smell under the study, where NA indicates the test has not been conducted for the smell as it has not occurred in any of the systems. According to the p-value, the alternative hypothesis has been rejected for all the smells except *Base Class Should Be Abstract* and *Refused Parent Bequest* in favor of null hypothesis. This rejection

dictates that there is no difference between the occurrences of the smells in open source and industrial systems. The acceptance of the alternative hypothesis for the smells *Base Class Should Be Abstract* and *Refused Parent Bequest* states that these smells occur more frequently in open source systems than industrial ones.

5 THREATS TO VALIDITY

The construct, external and reliability threats to validity [11] of this empirical study are discussed in this section.

Threats to construct validity: The potential threats are related to the accuracy of the tool used in this study to identify code smells from source systems [11]. The inaccurate results might lead to a different phenomenon than our investigation. However, to mitigate this threat we have used the tool that has been evaluated in previous studies having good accuracy.

Threats to external validity: Three potential threats have been identified of this study while generalizing the findings of the study from the sample. First, we have used Java systems in our study, and there is a possibility that the results would be different for other object-oriented languages, like - C#. Second, results cannot be generalized to other types of code smells. Finally, we cannot extrapolate our results to more other open source and industrial systems. Since it is not available to have industrial systems for this research domain, analyzing a lot of systems is difficult. However, we have collected 15 industrial from various industries of different countries instead of one industry or country and 25 open source systems which might mitigate the threats to some extent.

Threats to reliability validity: The potential threats concerns to the replication of the analysis performed in this study using the industrial systems. Since these systems are confidential, we are unable to provide information about these. However, the results can be reproduced by using the dataset for open source systems using the tool mentioned in Section 3.

6 CONCLUSION

The study attempts to identify the most frequent occurring code smells in both types of systems - open source and industrial. The inclusion of the industrial systems along with the open source ones makes the study more interesting in a sense that most of the smell-occurrences are similar to both types of the systems. In addition, it is interesting that 6 out of 18 smells have not occurred in any system, whereas three smells - *Long Method, Complex Class* and *Long Parameter List* have been the most frequent ones in both types of systems. Therefore, these findings will assist developers to reduce the smells while developing systems, and researchers to innovate the smell refactoring techniques from the viewpoint of frequency analysis. Our future plan is to find the reasoning of why the smells occur frequently while the others are not.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of Information and Communication Technology (ICT) Division, Ministry of Posts, Telecommunications and Information Technology, Bangladesh through Grant/Fellowship Number: 56.00.0000.052.33.005.21-2, 18-01-2022.

REFERENCES

[1] Alexander Chatzigeorgiou and Anastasios Manakos. 2014. Investigating the evolution of code smells in object-oriented systems. *Innovations in Systems and Software Engineering* 10, 1 (2014), 3–18.

[2] Martin Fowler. 2018. *Refactoring: improving the design of existing code.* Addison-Wesley Professional.

[3] Foutse Khomh, Massimiliano Di Penta, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. 2012. An exploratory study of the impact of antipatterns on class change-and fault-proneness. *Empirical Software Engineering* 17, 3 (2012), 243–275.

[4] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. 2010. Introduction to information retrieval. *Natural Language Engineering* 16, 1 (2010), 100–103.

[5] Naouel Moha, Yann-Gaël Guéhéneuc, Laurence Duchien, and Anne-Francoise Le Meur. 2009. Decor: A method for the specification and detection of code and design smells. *IEEE Transactions on Software Engineering* 36, 1 (2009), 20–36.

[6] Emerson Murphy-Hill, Chris Parnin, and Andrew P Black. 2011. How we refactor, and how we know it. *IEEE Transactions on Software Engineering* 38, 1 (2011), 5–18.

[7] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. A large-scale empirical study on the lifecycle of code smell co-occurrences. *Information and Software Technology* 99 (2018), 1–10.

[8] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Denys Poshyvanyk, and Andrea De Lucia. 2014. Mining version histories for detecting code smells. *IEEE Transactions on Software Engineering* 41, 5 (2014), 462–489.

[9] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering* 23, 3 (2018), 1188–1221.

[10] Fabiano Pecorelli, Fabio Palomba, Dario Di Nucci, and Andrea De Lucia. 2019. Comparing heuristic and machine learning approaches for metric-based code smell detection. In *Proceedings of the 27th International Conference on Program Comprehension (ICPC).* IEEE, 93–104.

[11] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons.

[12] Dag IK Sjøberg, Aiko Yamashita, Bente CD Anda, Audris Mockus, and Tore Dybå. 2012. Quantifying the effect of code smells on maintenance effort. *IEEE Transactions on Software Engineering* 39, 8 (2012), 1144–1156.

[13] Nikolaos Tsantalis, Theodoros Chaikalis, and Alexander Chatzigeorgiou. 2018. Ten years of JDeodorant: Lessons learned from the hunt for smells. In *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering (SANER).* IEEE, 4–14.

[14] Aiko Yamashita and Leon Moonen. 2013. Exploring the impact of inter-smell relations on software maintainability: An empirical study. In *Proceedings of the 35th International Conference on Software Engineering (ICSE).* IEEE, 682–691.

[15] Aiko Yamashita, Marco Zanoni, Francesca Arcelli Fontana, and Bartosz Walter. 2015. Inter-smell relations in industrial and open source systems: A replication and comparative analysis. In *Proceedings of the 2015 IEEE International conference on software maintenance and evolution (ICSME).* IEEE, 121–130.