Understanding the Relationship between Missing Link Community Smell and Fix-inducing Changes

Toukir Ahammed, Moumita Asad and Kazi Sakib
Institute of Information Technology, University of Dhaka, Bangladesh

Keywords: Community Smell, Fix-Inducing Change, Software Bug.

Abstract: Missing link smell implies the situation when developers contribute to the same source code without communicating with each other. Existing studies have analyzed the relationship of missing link smells with code smell and developer contribution. However, the relationship between missing link smell and the introduction of bug has not been explored yet. This study investigates how missing link smell is related with Fix-Inducing Change (FIC). For this purpose, Spearman's rank correlation is measured between the number of smelly commits and FIC commits. The result shows that missing link smell and FIC are positively correlated. Furthermore, it is found that bugs introduced in smelly commits are mostly major type in terms of severity.

1 INTRODUCTION

Community smells can be defined as organizational and social anti-patterns in a development community (Tamburri et al., 2015). Community smells may lead to the emergence of social debt which denotes unforeseen project costs connected to a sub-optimal software development community. Although community smells may not be an immediate obstacle for software development, these can affect software maintenance negatively in the long run (Palomba et al., 2018c). Missing link is one of the most commonly reported community smells which occurs if developers do not communicate with each other while working collaboratively (Magnoni, 2016).

Missing link community smell implies the lack of communication among developers that can create a knowledge gap in the development community (Tamburri et al., 2019). As a software product can be thought of as the combined effort of all developers, the lack of communication and cooperation can negatively affect mutual awareness and trust among developers (Magnoni, 2016). Previous studies found that community smells including missing link smell are related to code smells (Giarola, 2018) and have an impact on code smell intensity (Palomba et al., 2018b; Palomba et al., 2018c). Since code smells are found to be successful indicators of bugs in software systems (Khomh et al., 2012; Palomba et al., 2018a), the relationship between community smells and bugs needs to be investigated.

In previous studies, the definition and detection of missing link smell in open-source projects have been studied. A few studies have explored the impact of missing link smell on different software artifacts such as code smell. Magnoni proposed the identification pattern of missing link community smell (Magnoni, 2016). Tamburri et al. examined the relationship between community smells and different socio-technical factors, e.g., socio-technical congruence, turnover, etc. (Tamburri et al., 2019). They considered missing link, organizational silo, black cloud and radio silence community smell. Palomba et al. investigated the impact of missing link smell and four other community smells on code smell intensity (Palomba et al., 2018c). Catolino et al. analyzed the role of four community smells including missing link smell on gender diversity and women's participation in open-source communities (Catolino et al., 2019). Although existing studies focused on analyzing community smell from different perspectives, there has been no study investigating the relationship between missing link smell and bug introduction so far.

In this context, the current study analyzes the relationship between missing link smell and Fix-Inducing Changes (changes that introduce error into the system). For analysis, seven diverse and open-source projects such as ActiveMQ and Cassandra are selected based on several criteria (e.g., availability of developer mailing list). First, missing link smells are identified in each project finding cases where a collaboration link does not have its communication counterpart. Then, the developers involved with each smell are identified by extracting the instance of smell. Then, commits that have been submitted by developers involved in missing link smell are marked as smelly commits. Besides, commits that represent FIC are identified by analyzing the project repository. Finally, a correlation analysis is performed between the number of smelly commits and FIC commits using Spearman's rank correlation coefficient (Spearman, 1987). To understand the severity of bugs that are introduced by developers who are involved in missing link smell, FIC commits that are submitted by smelly developers have been linked to the bug repository. After linking FIC to the bug repository, the information of severity is extracted from the bug report.

The result of the study shows that there is a significant positive correlation between the number of smelly commits and FIC commits. The study also finds that bugs occurring in smelly commits are related to major loss of functionality.

2 BACKGROUND

This section provides the basic idea of Missing Link Community Smell and Fix-Inducing Change.

2.1 Missing Link Community Smell

Missing link community smell refers to the situation when two developers collaborate in a part of source code but do not communicate with each other (Magnoni, 2016). This smell can be identified by finding those collaborations for which there is no communication found in the defined communication channel, e.g., mailing list. The occurrence of missing link smell is described below with a sample software development community taken from (Ahammed et al., 2020).

Figure 1 represents a software development community of six developers. Developers are connected through the solid line if they communicate with each other. The dashed line connects developers to the source code on which they work. Two types of Developer Social Network (DSN) can be generated from this development community to identify missing link smell. Firstly, a communication DSN can be generated from Figure 1 by considering only communication links which is displayed in Figure 2. Then, a collaboration network can be generated by linking developers who work in the same part of the source code. Figure 3 represents the collaboration DSN for the considered development community. For example, developer *A* and developer *B* work in the same source code file (Figure 1), so they are connected in the collaboration DSN (Figure 3).

Missing link smell now can be identified by comparing the collaboration network with the communication network. It can be easily observed that one link, $E-F$, in the collaboration network (Figure 3) does not have the corresponding counterpart in the communication network (Figure 2). Hence, it represents an instance of missing link smell between developer *E* and developer *F*.

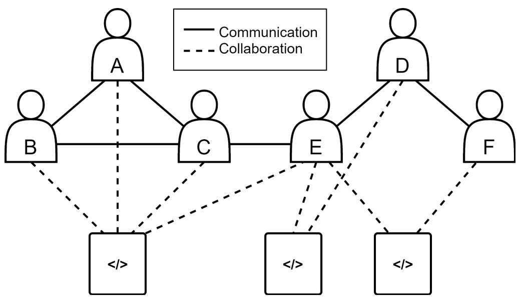

Figure 1: A Sample Software Development Community.

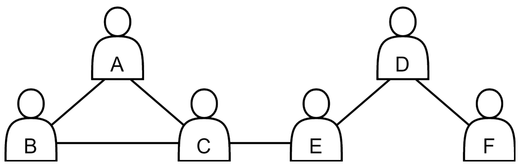

Figure 2: Communication DSN.

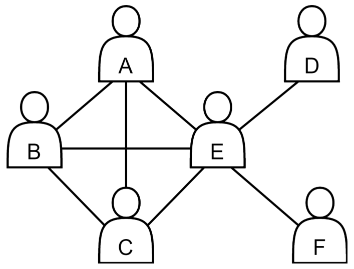

Figure 3: Collaboration DSN.

2.2 Fix-Inducing Changes (FIC)

Fix-Inducing Changes (FIC) are the changes to code that create error in the system and cause a fix later (Śliwerski et al., 2005). FIC can be identified starting from the point where the bug is fixed (Kim et al., 2008). The changes that fix a bug is called Fixing Changes (FC). FC commits can be found by searching the commit message for bug fixing keywords such as "Fix", "Bug" or "Patch". FC commit contains the change that is modified or deleted to fix the bug. By tracking the origin of these changes, FIC commits can be identified where the buggy code is introduced.

3 RELATED WORK

In recent times, community smells are studied to incorporate the organizational and social aspects of the software development community in software engineering research. Some studies focused on defining different types of community smell (Tamburri et al., 2013; Tamburri et al., 2015) while others focused on identifying these smells in open-source projects (Magnoni, 2016; Tamburri et al., 2019). Besides, a few studies investigated the relationship and the impact of community smells on different software artifacts such as code smell, socio-technical congruence (Palomba et al., 2018c; Tamburri et al., 2019).

The concept of community smell is first introduced in an industrial case study (Tamburri et al., 2015). The authors defined nine different community smells and proposed a list of possible mitigations of these smells such as learning community, cultural conveyor, stand-up voting, etc. Magnoni proposed the identification pattern of four community smells and developed a tool named *Codeface4Smells* (Magnoni, 2016) extending an existing socio-technical network analysis tool *Codeface*[1] (Joblin et al., 2015). The tool is capable of detecting community smells in open-source projects.

Tamburri et al. explored the diffuseness of community smells and how developers perceived their presence and effect (Tamburri et al., 2019). The authors found that community smells are highly diffused in open-source projects and developers recognized them as relevant problems for software evolution. The authors also analysed the relationship between community smell and socio-technical factors such as socio-technical congruence, turnover, etc.

The relationship between community smells and code smells was investigated in (Giarola, 2018). A community-aware code smell intensity model was proposed in (Palomba et al., 2018c) to investigate the impact of community smells on predicting code smell intensity. The authors found that community smells influenced the intensity of code smells more than other community related factors.

Catolino et al. analysed the impact of gender diversity and women's participation on community smells (Catolino et al., 2019). The authors found that gender-diverse teams had fewer community smells compared to non-gender-diverse teams and the presence of women in teams can reduce the number of community smells. Catolino et al. suggested some refactoring strategies to deal with community smells in practice such as mentoring, creating communication plan and restructuring the development community (Catolino et al., 2020). The involvement of developers in missing link smell was analysed in (Ahammed et al., 2020). The authors found that a small part of community members was involved in missing link smell and the involvement in smell was correlated with developer contribution.

The existing studies focused on defining and detecting community smells as well as relating it with other technical aspects such as code smell. However, the relationship between missing link smell and bug introduction has not been investigated yet.

4 METHODOLOGY

This study aims at understanding the relationship between missing link smell and bug introduction in software projects. First, the smelly commits are identified by finding the missing link smell in the project. Then, the Fix-Inducing Changes (FIC) are identified from the project repository by finding erroneous code changes that induce a fix later. Finally, the relation between smelly commits and FIC commits is analysed.

4.1 Smelly Commits Detection

To identify smelly commits, missing link smells are detected according to the identification pattern introduced by (Magnoni, 2016). The projects are analysed using a six-month window. For each window, a collaboration DSN is generated analysing the project's *GitHub* repository. All commits are analyzed and developers who contribute to the same part of source code within that window are connected through an edge. Next, a communication DSN is constructed examining the mailing list of the project. All messages in the mailing list are analysed and developers who replied in the same email within a given window are connected. Finally, collaboration DSN and communication DSN are compared to find missing link smell. For each edge in the collaboration network, the corresponding communication part is searched in the communication DSN. Any edge that is present in collaboration DSN but absent in communication DSN is identified as missing link smell.

The above mentioned steps are performed using *Codeface4Smells* tool[2]. This tool returns the list of missing link smell per window along with the corresponding developers involved with these smells. The developers are identified as smelly developers and their all commits in that window are marked as smelly commits.

[1] http://siemens.github.io/codeface

[2] https://github.com/maelstromdat/CodeFace4Smells

4.2 Fix-Inducing Changes (FIC) Detection

FICs are the erroneous changes to the code that induce fixes in the future. The process of detecting FIC adopted in this study is similar to (Kim et al., 2008; Huq et al., 2019). To find FIC, the following steps are followed:

The first step of detecting FIC is finding changes that fix a bug, called the Fixing Changes (FC). To find the FCs, all commit messages are extracted from the project repository. Then, commit messages are searched for keywords - "Fix", "Bug", "Patch" including their past and gerund form. These commits indicate bug fixing activities and are labeled as FC commits. Next, changes made in each FC commit are extracted comparing with its immediate parent commit. *Diffj* tool[3] is used to obtain the location of changes, i.e, modified or deleted line numbers. *Diffj* tool ignores white space or other formatting changes and thus can mitigate the possibility of finding false FICs (Kim et al., 2006). Finally, the origin of these change locations is tracked using *git-blame*[4] command. The command is used to identify which commit is responsible for the latest changes made to a specific line of a file. This leads to the commit that introduces the bug that is FIC. Both FIC and the corresponding FC commits are stored for analysis.

4.3 Analysing Relationship and Bug Severity

To understand the direction of the relationship between the number of smelly commits and the number of FICs, correlation analysis is conducted. As a monotonic trend is observed between these two variables, Spearman's rank correlation (Spearman, 1987) method is chosen. A monotonic trend implies both variables tend to increase together and decrease together, or the opposite, but not exactly at a constant rate like a linear relationship. In Spearman's rank correlation coefficient, the relationship between two variables can be assessed using a monotonic function. The interpretation of the correlation coefficient is adapted from (Dancey and Reidy, 2007) and displayed in Table 1. The correlation coefficient, ρ, indicates the strength of the correlation. The value -1 or +1 means a perfect relationship and 0 means no relationship between two variables. As the value approaches -1 or +1, it indicates more strong correlation. A value closer to 0 indicates a weaker relationship. The positive value indicates a positive correlation whereas the negative value indicates a negative correlation. The correlation coefficient is considered significant in this study if the p-value is less than 0.01.

To understand the severity of bugs that are introduced while developers are involved in missing link smell, Smelly FIC commits are analysed. For every Smelly FIC commit, the corresponding FC commits are identified from the mapping stored in FIC detection step. Only those FC commits are considered that contain bug ID in their commit message. The corresponding bug report is retrieved from the bug repository using that bug ID. Thus Smelly FIC commits are linked to the bug repository and the severity of bugs introduced in these commits can be known.

[3]https://github.com/jpace/diffj

[4]https://git-scm.com/docs/git-blame

Table 1: Interpretation of the Spearman's rank correlation coefficient.

ρ (Negative)	ρ (Positive)	Interpretation
$\rho = 0$	$\rho = 0$	Zero
$-0.4<\rho<0.0$	$0.0<\rho\leq 0.4$	Weak
$-0.7<\rho\leq -0.4$	$0.4\leq\rho<0.7$	Moderate
$-1<\rho\leq -0.7$	$0.7\leq\rho<1$	Strong
$\rho = -1$	$\rho = 1$	Perfect

5 EXPERIMENTATION AND RESULT ANALYSIS

This section presents the dataset description and the result analysis of the study.

5.1 Dataset

This study aims at investigating the relationship between missing link smell and FIC. To perform the analysis, the study needs some specific software artifacts such as collaboration information, communication information and bug severity information. Thus the choice of the subject systems for this study is guided by the following factors:

1. Publicly available source code hosted in version control system
2. Publicly available archive of Developer mailing list
3. Bug repository maintaining the information of bug severity

Therefore, seven open-source projects from *Apache* ecosystem are selected for analysis considering the above criteria. The name of the selected projects is provided in Table 2 with their source code repository, mailing list and analysis period. These projects are

Table 2: List of Analysed Projects.

#	Project	Source Code	Mailing List	Analysis Period
1	ActiveMQ	github.com/apache/activemq	gmane.comp.java.activemq.devel	Apr-2006 - Dec-2020
2	Cassandra	github.com/apache/cassandra	gmane.comp.db.cassandra.devel	Oct-2009 - Sep-2020
3	Cayenne	github.com/apache/cayenne	gmane.comp.java.cayenne.devel	Nov-2007 - Aug-2020
4	CXF	github.com/apache/cxf	gmane.comp.apache.cxf.devel	Nov-2010 - Sep-2020
5	Jackrabbit	github.com/apache/jackrabbit	gmane.comp.apache.jackrabbit.devel	Dec-2005 - Sep-2020
6	Mahout	github.com/apache/mahout	gmane.comp.apache.mahout.devel	Oct-2008 - Aug-2020
7	Pig	github.com/apache/pig	gmane.comp.java.hadoop.pig.devel	Oct-2010 - Aug-2020

hosted in the online version control system *GitHub*. The development mailing list archive is available on *Gmane*[5]. All the selected projects use *Jira*[6] as the issue tracker. Projects of different ages and sizes are chosen for analysis. The age of the selected projects ranges from 10 to 15 years and the number of commits ranges from 2,451 to 17,098.

5.2 Correlation Analysis Result

To get an idea regarding the proportion of developers involved in missing link smell, the ratio of smelly committers to total committers is calculated. Table 3 shows the ratio of smelly committers per six-month analysis window for each evaluated project. The first column shows the name of the project, the second column shows the number of windows analysed in each project. The average number of committers and smelly committers are presented in the third and fourth column. Finally, the ratio of smelly committers to total committers is shown in the last column. The result suggests that on average 53% committers are involved in missing link smell per window.

In this study, the relationship between missing link smell and bug introduction is examined. To understand the relation, Spearman's rank correlation is performed between the number of smelly commits and the the number of FIC commits for all projects individually. The number of smelly commits and number of FIC commits are calculated for each window. The result of correlation analysis is shown in Table 4 with Spearman's correlation coefficient (ρ) and corresponding p-value.

The correlation coefficient is interpreted according to Table 1 and considered to be significant if the p-value is less than 0.01. The result suggests that there is a significant positive correlation between number of smelly commits and FIC commits. Among seven evaluated projects, *CXF*, *Pig*, *ActiveMQ*, *Cayenne*, *Jackrabbit* and *Mahout* show a strong positive correlation. A moderate positive correlation is found in *Cassandra*.

These results suggest that missing link smell and bugs are correlated in terms of number of smelly commits and number of FIC commits. It indicates that commits submitted by smelly developers, i.e., smelly commits, are very likely to introduce bugs in the system. This information can help the reviewing process in open-source projects. Smelly commits should be reviewed thoroughly to avoid possible bug introduction.

5.3 Bug Severity Analysis Result

Smelly FIC commits are analysed to understand the severity of bugs introduced by developers who are involved in missing link smell. The following five bug severity categories are found in *Jira* for these projects.

1. **Blocker.** These bugs block development and/or testing work. The production can not run.
2. **Critical.** These bugs cause crashes, loss of data, or severe memory leaks.
3. **Major.** These bugs result in major loss of function.
4. **Minor.** These bugs cause minor loss of function or other problems where an easy workaround is present.
5. **Trivial.** These bugs are about cosmetic problems, e.g., misspelled words or misaligned text.

All the evaluated projects except *Cassandra* use the above categorization for bug severity.

Table 5 reports the severity of bugs introduced in Smelly FIC commits. For example, Smelly FIC commits introduce 4.4% *Blocker* bugs, 7.7% *Critical* bugs, 74.4% *Major* bugs, 11.4% *Minor* bugs and 2.2% *Trivial* bugs in *CXF* project. Figure 4 shows that most of the bugs produced by smelly commits are major bugs. On average, 78.5% Smelly FIC introduce *Major* bugs, 15.6% introduce *Minor* bugs, 3.6% introduce *Critical*, 1.4% introduce *Blocker* bugs and 0.9% introduce *Trivial* bugs in the system.

[5]http://gmane.io

[6]https://issues.apache.org/jira/

Table 3: Percentage of Smelly Committers per window.

#	project	#analysedWindows	Avg. #committers	Avg. #SmellyCommitters	Ratio
1	ActiveMQ	30	13.27	7.17	0.54
2	Cassandra	26	6.85	4.00	0.58
3	Cayenne	20	23.05	12.55	0.54
4	CXF	30	9.83	4.77	0.48
5	Jackrabbit	16	12.75	4.63	0.36
6	Mahout	24	8.21	4.25	0.52
7	Pig	20	6.10	4.70	0.77
	Overall	*166*	*11.17*	*5.92*	*0.53*

Table 4: Correlation Analysis.

#	project	rho (ρ)	p-value
1	ActiveMQ	0.858	<0.01
2	Cassandra	0.648	<0.01
3	Cayenne	0.797	<0.01
4	CXF	0.944	<0.01
5	Jackrabbit	0.768	<0.01
6	Mahout	0.769	<0.01
7	Pig	0.941	<0.01

Table 5: Bug Severity of Smelly FIC Commits.

#	project	Blocker (%)	Critical (%)	Major (%)	Minor (%)	Trivial (%)
1	ActiveMQ	4.4	7.7	74.4	11.4	2.2
2	Cayenne	0.0	0.6	88.5	10.8	0.0
3	CXF	0.3	4.3	82.1	13.0	0.3
4	Jackrabbit	3.2	5.3	68.4	22.1	1.1
5	Mahout	0.0	1.5	67.0	31.0	0.4
6	Pig	0.7	2.1	90.7	5.0	1.4
	Average	*1.4*	*3.6*	*78.5*	*15.6*	*0.9*

These results suggest that developers introduce mostly *Major* level bugs in their FIC commits while involved in missing link smell. *Major* bugs are found to have longer fixing time in the literature (Panjer, 2007). Hence, extra maintenance effort and cost may be needed to fix these bugs introduced by the developers who are involved in missing link smell.

6 THREATS TO VALIDITY

This section presents several potential threats that may affect the validity of this study.

- **Threats to External Validity.** Threats to external validity deal with the generalization of the results of the study. Seven open-source projects from

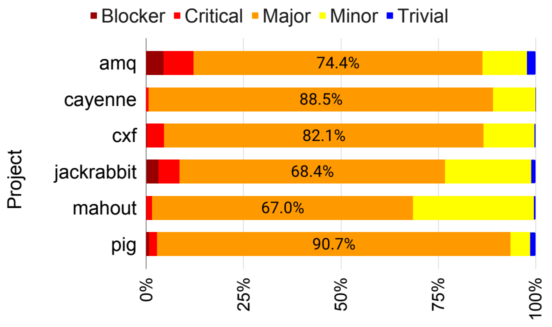

Figure 4: Bug Severity of Smelly FIC Commits.

Apache are analysed in this study. The choice of these projects is guided by several factors such as the availability of source code repository, mailing list archive and bug repository. However, projects of different sizes and ages are selected for analysis to mitigate this threat. The age of the evaluated projects varies from 10 to 15 years and the size of projects ranges between 2,451 to 17,098 in terms of number of commits.

- **Threats to Internal Validity.** Threats to internal validity deal with the factors that may threaten the validity of the result but are not accounted for. An open-source tool, *Codeface4Smells*, is used to detect missing link smell in this study. The identified smells are directly included in the analysis of this study without further verification. However, this tool is commonly used to detect community smell in related studies (Palomba et al., 2018c; Catolino et al., 2019). Moreover, this tool uses mailing list as the source of communication data to generate communication network. The result can be different if other communication channels, e.g., Skype, Slack, etc. are considered. However, according to contribution guidelines of evaluated projects, mailing list is the primary communication channel in these communities. Besides, mailing list is used as the communication source in other related studies (Joblin et al., 2015; Tamburri et al., 2019).

7 CONCLUSION AND FUTURE WORK

This study investigates the relationship between missing link smells and FIC. Furthermore, it examines the severity of bugs that are introduced in the system by the developers who are involved in missing link smell.

For this purpose, seven diverse and open-source projects from *Apache* are analysed. The correlation analysis shows there is a significant positive correlation between the number of smelly commits and FIC commits. In addition, results reveal that developers mostly introduce major bugs in the system while involved in missing link smell.

In the future, more open-source projects will be analyzed to generalize the observed result. Furthermore, other types of community smell, e.g., organizational silo, radio silence, can be included to find how they relate to the introduction of bugs.

ACKNOWLEDGEMENTS

The virtual machine facility used in this research is provided by Bangladesh Research and Education Network (BdREN).

REFERENCES

Ahammed, T., Asad, M., and Sakib, K. (2020). Understanding the involvement of developers in missing link community smell: An exploratory study on apache projects. In *QuASoQ@APSEC*, pages 64–70.

Catolino, G., Palomba, F., Tamburri, D. A., Serebrenik, A., and Ferrucci, F. (2019). Gender diversity and women in software teams: How do they affect community smells? In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society*, pages 11–20. IEEE.

Catolino, G., Palomba, F., Tamburri, D. A., Serebrenik, A., and Ferrucci, F. (2020). Refactoring community smells in the wild: the practitioner's field manual. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*, pages 25–34.

Dancey, C. P. and Reidy, J. (2007). *Statistics without maths for psychology*. Pearson education.

Giarola, F. (2018). Detecting code and community smells in open-source: an automated approach.

Huq, S. F., Sadiq, A. Z., and Sakib, K. (2019). Understanding the effect of developer sentiment on fix-inducing changes: an exploratory study on github pull requests. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 514–521. IEEE.

Joblin, M., Mauerer, W., Apel, S., Siegmund, J., and Riehle, D. (2015). From developer networks to verified communities: a fine-grained approach. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 563–573. IEEE.

Khomh, F., Di Penta, M., Guéhéneuc, Y.-G., and Antoniol, G. (2012). An exploratory study of the impact of antipatterns on class change-and fault-proneness. *Empirical Software Engineering*, 17(3):243–275.

Kim, S., Whitehead, E. J., and Zhang, Y. (2008). Classifying software changes: Clean or buggy? *IEEE Transactions on Software Engineering*, 34(2):181–196.

Kim, S., Zimmermann, T., Pan, K., James Jr, E., et al. (2006). Automatic identification of bug-introducing changes. In *21st IEEE/ACM international conference on automated software engineering (ASE'06)*, pages 81–90. IEEE.

Magnoni, S. (2016). An approach to measure community smells in software development communities.

Palomba, F., Bavota, G., Di Penta, M., Fasano, F., Oliveto, R., and De Lucia, A. (2018a). On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation. *Empirical Software Engineering*, 23(3):1188–1221.

Palomba, F., Tamburri, D. A., Serebrenik, A., Zaidman, A., Fontana, F. A., and Oliveto, R. (2018b). Poster: How do community smells influence code smells? In *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion*, pages 240–241. IEEE.

Palomba, F., Tamburri, D. A. A., Fontana, F. A., Oliveto, R., Zaidman, A., and Serebrenik, A. (2018c). Beyond technical aspects: How do community smells influence the intensity of code smells? *IEEE transactions on software engineering*.

Panjer, L. D. (2007). Predicting eclipse bug lifetimes. In *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)*, pages 29–29. IEEE.

Śliwerski, J., Zimmermann, T., and Zeller, A. (2005). When do changes induce fixes? *ACM sigsoft software engineering notes*, 30(4):1–5.

Spearman, C. (1987). The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471.

Tamburri, D. A., Kruchten, P., Lago, P., and van Vliet, H. (2013). What is social debt in software engineering? In *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 93–96. IEEE.

Tamburri, D. A., Kruchten, P., Lago, P., and Van Vliet, H. (2015). Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*, 6(1):10.

Tamburri, D. A., Palomba, F., and Kazman, R. (2019). Exploring community smells in open-source: An automated approach. *IEEE Transactions on software Engineering*.