Understanding the Involvement of Developers in Missing Link Community Smell: An Exploratory Study on Apache Projects

Toukir Ahammed, Moumita Asad and Kazi Sakib

Institute of Information Technology, University of Dhaka, Dhaka, Bangladesh

Abstract

Missing link smell occurs when developers collaborate in source code without communication. This can affect software maintenance by the means of lacking mutual awareness, mistrust and knowledge gap. Existing studies have investigated the relationship of missing link smell with code smell and different socio-technical factors like turnover. This study aims to understand how many developers are involved with missing link smell, by calculating the percentage of smelly developers for a project. The study also investigates the relationship between the number of contributions and the number of missing link involvements of a developer. The result shows that the percentage of smelly developers involved with missing link smell is 8.7% on average. The result also suggests a moderate positive correlation between the contribution of a developer to the project and the involvement in smell.

Keywords

missing link smell, community smell, software engineering, empirical analysis

1. Introduction

Community smells are the organizational and social anti-patterns in a development community [1]. Community smells may lead to the emergence of social debt which indicates unforeseen project costs connected to a sub-optimal software development community. Community smells may not be an immediate obstacle for software development but these can affect software maintenance negatively in the long run [2]. Missing link is one of the common community smells. It refers to the condition when two co-committing developers show uncooperative behavior by not communicating [3].

Missing link community smell decreases communication activities in the development community. The lack of communication and cooperation negatively affects mutual awareness and trust among developers [3]. A software product can be thought of as the combined effort of all developers. So, collaboration along with proper communication is necessary among developers. It is important to know how many developers are involved in missing link smell as they may affect the whole project. Identifying these developers and analyzing their characteristics is important. This will help the project managers to take steps such as task reassigning, team reformation, increasing awareness about communication etc. to keep communication issues lower among the developers in the community.

QuASoQ 2020: 8th International Workshop on Quantitative Approaches to Software Quality
EMAIL: bsse0806@iit.du.ac.bd (T. Ahammed); bsse0731@iit.du.ac.bd (M. Asad); sakib@iit.du.ac.bd (K. Sakib)

The detection of missing link smell and its impact on software artifacts have been analyzed in previous studies. S. Magnoni proposed the identification pattern of missing link community smell [3]. Tamburri et al. examined the relationship between community smells and different socio-technical factors, e.g., socio-technical congruence, turnover etc [4]. This study 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 [2]. Catolino et al. analyzed the role of four community smells including missing link smell on gender diversity and women participation in open-source community [5]. However, developer involvement in missing link smell and how developer contributions in the project relate to missing link smell have not been analyzed yet. In this context, the current study aims to focus on these factors by addressing the following Research Questions (RQs).

RQ1: How many developers are involved in missing link community smell?

In an open-source project, there can be many developers who contribute to the project. All the developers may not be involved in missing link community smell. This RQ aims to find how many developers are involved in missing link smells in a community. This is important to know the collective contribution of developers to the number of missing link smells in a project. This finding will help the project managers to understand the severity of communication issues among developers in the community. The action can be different to mitigate missing link smell based on the number of developers involved in smells.

RQ2: How does missing link smell relate with a developer contribution?

This RQ focuses on the involvement of individual developer in missing link smell. This RQ relates an important characteristic of a developer, i.e., contribution, to missing link smell. This finding will help project managers understanding which type of developers involve more in missing link smell. This information can be used to decide which developers can be monitored to control missing link smell in the community from the beginning of a project.

In this study, missing link smells are analyzed on seven open-source projects of *Apache* ecosystem. These projects are selected for being large enough to analyse and the availability of communication data, i.e., mailing list. First, the instances of missing link smell are detected in each project. The missing link smell is identified by finding cases where a collaboration does not have its communication counterpart. Then the developers associated with each smell are identified by extracting the instance of smell. The fraction of developers involved with missing link smell is calculated to check whether a subset of developers are involved with this type of smell. Then the correlation is investigated between the contribution of developers and their involvement in missing link smells.

The results of the study show that a small part of the total developers involved with missing link community smell. On average, 8.7% of the total developers of a project are involved with missing link smell. This study also finds a significant moderate positive correlation between the developer contribution and their involvement in missing link smell.

2. Background

This section provides some important terminologies to better understand the missing link community smell.

Developer Social Network (DSN): A network of a software development community where a node represents developer and relationships between developers, e.g., communication, coordination, are represented by an edge.

Collaboration Network: A specific type of DSN which indicates the collaboration in a development community. Here, a node represents a developer who contributes to the project in the version control system. Two developers are connected through an edge if they contribute to the same part of source code within a given time frame [3]. Figure 2 represents an example of a collaboration network.

Communication Network: A specific type of DSN which indicates the communication within the defined communication channel of a development community. Here, a node represents developers who communicate in the defined communication channel, i.e., mailing list. Two developers are connected through an edge if they replied in the same e-mail within a given time frame [3]. A communication network is illustrated in Figure 3.

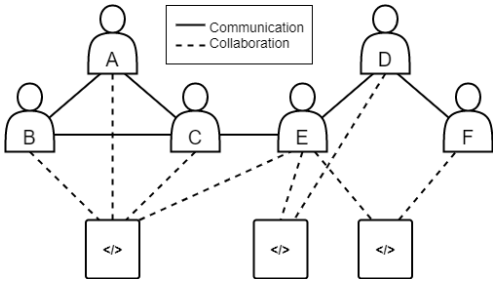

Figure 1: Developer Social Network

Missing Link Community Smell: A missing link community smell occurs when a couple of developers collaborate with each other but show uncooperative behaviors by not communicating. This smell can be identified by detecting collaboration between two developers that do not have the communication counterpart in defined communication channel, e.g., development mailing list [3].

An example of DSN is illustrated in Figure 1. The upper part of the graph represents communication and the lower part represents the collaboration among developers. The developers are connected with a solid line if they communicate with each other. The developers are connected to the file icon through a dashed line if they contribute to that source code file.

The collaboration and communication network can be generated separately from this DSN. Figure 2 and Figure 3 represent the collaboration and the communication network respectively. The missing link smell can be identified comparing the collaboration network with the communication network. There is a link between developer *E* and *F* in the collaboration network (Figure 2) but there is no corresponding link between these two developers in the communication network (Figure 3). Developer *E* and *F* are collaborating on the same part of source code but they are not connected through any communication link. Thus, this is considered as an instance of a missing link between developer *E* and *F*.

3. Related Work

In recent years, community smells are studied to incorporate the organizational and social aspects of developer community in software engineering research. Some studies focused on defining different community smells that can lead to unforeseen project costs [1], [6]. On the other hand, some studies investigated the impact of community

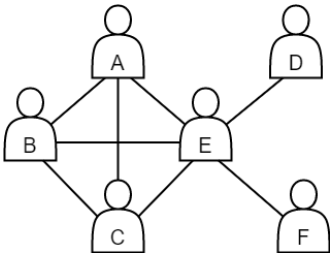

Figure 2: Collaboration Network

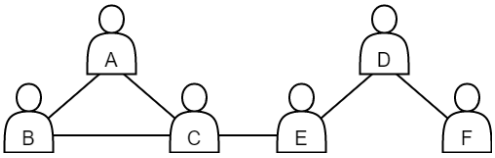

Figure 3: Communication Network

smells on different software artifacts [2], [4].

Tamburri et al. first introduced the concept of social debt in software engineering [6]. Later, in an industrial case study, they improved and elaborated the definition of social debt. In the same study, they defined nine different community smells which are connected to social debt [1]. They also suggested a list of possible mitigations of community smells such as learning community, cultural conveyors, stand-up voting etc., to avoid the negative effects.

Magnoni proposed the identification pattern of four out of nine community smells [3] defined in [1]. He developed an open-source tool *CODEFACE4SMELLS*[1] as an extension to *CODEFACE* [7]. This tool is capable of detecting community smells from the change history in the version control system and the communication history in development mailing list.

Tamburri et al. analysed the distribution of community smells in open-source projects [4]. They also assessed the relation between community smells and existing socio-technical quality factors, e.g., socio-technical congruence, communicability, turnover etc.

Palomba et. al examined the relationship between social and technical debt [2], [8]. They assessed the impact of community smells on code smells. They found community smells significantly influencing code smell intensity. They also proposed a community-aware code smell intensity model in which both technical and community related factors were considered.

Catolino et al. analysed the role of gender diversity and women participation in community smell [5]. They considered four types of community smell i.e., organizational Silo, Lone Wolf, Black Cloud and Radio Silence. They found that gender diverse team had a lower number of community smells than non-gender diverse team. They also showed that gender diversity and women participation were important factors for Black Cloud and Radio Silence whereas organizational Silo and Lone wolf were found partially related.

The existing studies have focused on community smells and the impact of these smells on software artifacts. The phenomenon of community smells is surrounded with developers in a development community. However, developer involvement in missing link smell and the relation between missing link smell and developer contributions have not been investigated yet. So, the developers involved with community smells and how their contribution relate to missing link smell need to be explored.

4. Methodology

This study aims to understand how many developers of a project are involved in missing link smell. This study also wants to assess the relationship between a developer's contribution and involvement in missing link smell. First, the missing link smell is detected for all the selected projects. Then the percentage of smelly developers is retrieved for each project. Later, the correlation analysis is performed between a developer's contribution and involvement in missing link smell.

4.1. Dataset

In this work, seven large open-source projects belonging to *APACHE* ecosystem are selected for analysis. These projects have been chosen because they are large and the mailing lists are publicly available. Table 1 provides the list of analysed projects with their name, source code link, development mailing list and analysis period. All projects are hosted in online version control system *GitHub* and the development mailing list archives are available on *Gmane*[2].

The selected projects are large enough in terms of community members and the number of commits. The projects have 668 community members on average. All the projects have a substantial number of commits, with an average of 10359. Thus the study has enough collaboration and communication data for analysis.

4.2. Missing Link Smell Detection

The selected projects are analysed using a six-month analysis window. The analysis period of a project starts from when both communication in mailing list and change history in repository are available. A few more months

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

[2] http://gmane.io

Table 1
List of Analysed Projects

#	Project Name	Source Code	Mailing List	Analysis Period
1	Apache Cassandra	github.com/apache/cassandra	gmane.comp.db.cassandra.devel	Oct-2009 - Sep-2020
2	Apache Cayenne	github.com/apache/cayenne	gmane.comp.java.cayenne.devel	Nov-2007 - Aug-2020
3	Apache CXF	github.com/apache/cxf	gmane.comp.apache.cxf.devel	Nov-2010 - Sep-2020
4	Apache Jackrabbit	github.com/apache/jackrabbit	gmane.comp.apache.jackrabbit.devel	Dec-2005 - Sep-2020
5	Apache Jena	github.com/apache/jena	gmane.comp.apache.jena.devel	Oct-2012 - Sep-2020
6	Apache Mahout	github.com/apache/mahout	gmane.comp.apache.mahout.devel	Oct-2008 - Aug-2020
7	Apache Pig	github.com/apache/pig	gmane.comp.java.hadoop.pig.devel	Oct-2010 - Aug-2020

are excluded to make the analysis period divisible by six months. The analysis period for each project is given in Table 1. For example, Apache Cassandra project has the analysis period of 11 years starting from October 2009 to September 2020.

For every analysis window of a project, a communication network and a collaboration network is built. The communication network is generated by extracting communication data from development mailing list and the collaboration network is generated by extracting collaboration data from the project repository. After having both communication and collaboration networks, the instances of missing link smell are identified by comparing every collaboration link with communication networks. If any collaboration link does not have its communication counterpart, this link is identified as a missing link instance.

An open-source tool, *CODEFACE4SMELLS* [4], is used to detect missing link community smell in this study. This tool is capable of detecting missing link smell in the aforementioned way from project repository and development mailing list. The tool requires the link of source code repository and mailing list archive as input. Then the tool returns a list of missing link instances for each window of the project. A missing link instance is represented by a pair of developers. For example, (a, b) represents a missing link instance between developer a and b.

4.3. Smelly Developers Identification

A developer involved with a missing link smell is considered as a smelly developer. An instance of missing link smell consists of two collaborating developers who do not communicate with each other. Thus for every missing link smell, there are two smelly developers. *CODEFACE4SMELLS* outputs a missing link instance as a pair of developers. So, the smelly developers can be obtained by extracting all missing link instances of a project. The smelly developers of a project x can be denoted by a set SD_x. The number of smelly developers of the project will be the number of elements in SD_x.

To calculate the percentage of smelly developers in a project, the total number of developers of that project is required. The total number of developers is defined as the sum of the number of developers who contribute to source code and the number of members who communicate on mailing list [3]. The total number of developers of a project is obtained by counting the number of members present in either collaboration or communication network generated by *CODEFACE4SMELLS*. The percentage of smelly developers of a project is calculated using the following formula (Equation 1),

$$percSD_x = \frac{numSD_x}{totalDev_x} \times 100\%, \tag{1}$$

where $numSD_x$ is the number of smelly developers in project x and $totalDev_x$ is the number of total developers in project x.

4.4. Correlation Analysis

RQ2 aims to understand the relationship between a developer's contribution and involvement in missing link smell. To address this RQ, the correlation between following two measures is analysed:

1. how many commits a developer has in the project repository
2. how many times a developer is involved in missing link smell

In open-source projects, commits are the most representative form of coding contribution [9]. So, the contribution of a developer in a project is measured by the number of commits of that developer in the project repository. The number of commits of every individual developer is retrieved from the source code repository.

The number of involvement in missing link smells can be obtained from the list of missing link instances of a project. First, the developers are extracted from all the missing link instances of the project. Then the number of involvement is calculated counting how many times a developer occurs in the list.

Both the number of commits and the number of involvement in smells of a developer are converted into

Table 2
Correlation coefficient interpretation

Correlation Coefficient (Negative)	Correlation Coefficient (Positive)	Interpretation
$-0.4 < \tau_b \leq 0.0$	$0.0 \leq \tau_b < 0.4$	Weak
$-0.7 < \tau_b \leq -0.4$	$0.4 \leq \tau_b < 0.7$	Moderate
$-0.9 < \tau_b \leq -0.7$	$0.7 \leq \tau_b < 0.9$	Strong
$-1.0 \leq \tau_b \leq -0.9$	$0.9 \leq \tau_b \leq 1.0$	Very Strong

percentage to achieve the relative measurement. The commit percentage of a developer is calculated using Equation 2.

$$percentCommit = \frac{numCommit_i}{\sum_{i=1}^{n} numCommit_i} \times 100\% \quad (2)$$

where $numCommit_i$ is the number of commits of developer i and n is the total number of smelly developers.

Equation 3 is used to calculate missing link smell involvement of a developer in percentage.

$$percentMissingLink = \frac{numMissingLink_i}{\sum_{i=1}^{n} numMissingLink_i} \times 100\% \quad (3)$$

where $numMissingLink_i$ is the number of involvement in missing link smells of developer i and n is the total number of smelly developers.

Finally, the correlation analysis is performed between *percentCommit* and *percentMissingLink* for each project individually. Kendall's tau-b [10] is used to assess the degree of association between these two variables. Both *percentCommit* and *percentMissingLink* have tied values in the dataset. As Kendall's tau-b can handle tied ranks, this is used for the correlation analysis. The correlation coefficient is considered significant if the p-value is less than 0.01. The correlation coefficient is interpreted according to Table 2. The correlation coefficient, τ_b, indicates the strength of the correlation. τ_b has a range of value from -1.0 to 1.0. As τ_b closes to 0, it indicates less correlation between two variables. As τ_b approaches to -1.0 or +1.0, the strength of correlation between two variables is increased. The positive value of τ_b indicates a positive correlation and the negative value of τ_b indicates a negative correlation between two variables.

5. Result Analysis

This section presents the result analysis and discussion of this study. All the missing link smells found in selected projects are analysed to answer the two research questions. Analysis and discussion for both research questions are provided as follows.

5.1. RQ1: How many developers are involved in missing link community smell?

To answer this RQ, all missing link smells of a project are considered. For every project, the number of total developers and the number of smelly developers are calculated. Then the percentage of smelly developers is obtained for each project.

Table 3 demonstrates the percentage of smelly developers for each project. For example, Apache Cassandra project has 1380 total developers and 205 smelly developers which is 14.9% of total developers. It is observed that on average 10.5% of total developers of a software community are involved in missing link smells. *Apache Cayenne* community has the highest percentage of smelly developers (21.1%). This is also the smallest community among 7 communities. Tamburri et. al. found that the number of community smell grows quadratically with the number of community members until the threshold of 200 community members [4]. The occurrences of community smell tend to stabilize after this threshold. As the number of total developers in *Apache Cayenne* community is less than 200, the number of missing link smell has not stabilized yet. So, this project has relatively more missing link smell and consequently more smelly developers. Excluding *Apache Cayenne* project, the rest six projects have 8.7% smelly developers on average.

These results suggest that only a small portion of developers in an open-source software community are involved with missing link smells. They do not communicate appropriately with their co-committing or collaborative developers. Thus, they contribute to the total number of community smells in a software community.

5.2. RQ2: How does missing link smell relate with a developer contribution?

To answer this RQ, the correlation between a developer's contribution and involvement in missing link smell is analyzed. Kendall's tau-b is used as a correlation technique since it can handle tied values.

First, the correlation analysis is performed individually for each development community. The Kendall's tau-b coefficients and p-values are provided in Table 4. For

Table 3
Percentage of Smelly Developers

#	Project Name	Total Developers	Smelly Developers	Smelly Developers(%)	Average
1	Apache Cassandra	1380	205	14.9%	**8.7%**
2	Apache CXF	972	94	9.7%	
3	Apache Jena	244	34	13.9%	
4	Apache Mahout	615	28	4.6%	
5	Apache Pig	668	22	6.0%	
6	Apache Jackrabbit	927	28	3.0%	
7	Apache Cayenne	175	37	21.1%	
	Average	**668**	**64**	**10.5%**	

Table 4
Correlation Analysis

#	Project Name	Tau-b	p-value
1	Apache Cassandra	0.508	< 0.01
2	Apache Cayenne	0.543	< 0.01
3	Apache CXF	0.528	< 0.01
4	Apache Jackrabbit	0.589	< 0.01
5	Apache Jena	0.452	< 0.01
6	Apache Mahout	0.409	< 0.01
7	Apache Pig	0.513	< 0.01
	Overall	**0.612**	**< 0.01**

example, the correlation coefficient for Apache Cassandra project is 0.508 and it represents a moderate positive correlation. The value of correlation coefficient is significant with a p-value less than 0.01. All seven projects of this study show a moderate positive correlation between number of commits and number of smells which is statistically significant with p<0.01.

Another correlation analysis is performed after combining the data from all the projects. The value of the correlation coefficient is slightly increased to 0.612 but still falls under the range of moderate positive correlation. This result is also statistically significant with a p-value less than 0.01.

These results suggest that a developer who contributes more in a project tends to have more missing link smells. This can happen because a developer, who contributes more, have to communicate more with other developers. The overload of communication may be the reason for involving in more missing link smells than others. From another point of view, a developer having more contribution to a project is likely to be more familiar and experienced with that project. As he knows most of the aspects of that project, he may take the communication with co-committers lightly while contributing. However further analysis is required to find out the causes of involving in more smells.

6. Threats to Validity

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

Threats to external validity: Threats to external validity concern the generalization of the obtained results. In this study, seven projects from *Apache* are analysed. Thus the generalisation requires more projects belonging to different systems. However, to mitigate this threat large and diverse projects are selected that have a long change history - 11 years on average.

Threats to internal validity: Threats to internal validity concern the factors that can influence the result but are not accounted for. In this study, *CODEFACE4SMELLS* tool is used for the detection of missing link smell. The outputs of *CODEFACE4SMELLS* are directly incorporated in this study without checking whether there is any defect in the tool. However, the capability of this tool of identifying missing link smell was evaluated in [3]. This tool is also used in other studies in detecting community smells [2], [5], [11].

Moreover, this tool relies on mailing list to detect communication among developers. But there may exist other communication channels, e.g., Skype, Facebook etc., where developers communicate with each other. The result can be changed if these communication source are considered. However, mailing list represents the main communication channel for the projects analysed in this study according to the contribution guidelines of these projects. Besides, mailing list is used as the communication source in other related studies [4], [7].

7. Conclusion

This study explores the percentage of developers in a software development community involved in missing link smells. Furthermore, the relationship between developer contribution and involvement in missing link smell is examined. At first, missing link smells are detected for all the projects. Next, the smelly developers are identified

by extracting missing link instances. The percentage of smelly developers are calculated for every project. The number of appearances of a developer in missing link smell is counted. The contribution of a developer to a project is measured by the number of commits. Finally, correlation analysis is done between contribution and involvement in smell.

This study analyses seven open-source projects of *Apache*. The result shows that the number of developers involved in missing link smells is 8.7% on average. This study also founds that there is a moderate positive correlation between the number of commits of a developer and the number of involvement in missing link smells. The developers who contribute more tend to involve in more missing link smell.

In future, projects from other systems can be analysed to assess the generalization of the result. Besides, other types of community smell, e.g., organizational silo, radio silence, can be examined to find their association with developers contribution.

Acknowledgments

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

References

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

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

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

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

[5] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, F. Ferrucci, 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, IEEE, 2019, pp. 11–20.

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

[7] M. Joblin, W. Mauerer, S. Apel, J. Siegmund, D. Riehle, From developer networks to verified communities: a fine-grained approach, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, volume 1, IEEE, 2015, pp. 563–573.

[8] F. Palomba, D. A. Tamburri, A. Serebrenik, A. Zaidman, F. A. Fontana, R. Oliveto, Poster: How do community smells influence code smells?, in: 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion, IEEE, 2018, pp. 240–241.

[9] S. Daniel, R. Agarwal, K. J. Stewart, The effects of diversity in global, distributed collectives: A study of open source project success, Information Systems Research 24 (2013) 312–333.

[10] M. G. Kendall, Rank correlation methods, 1948.

[11] F. GIAROLA, Detecting code and community smells in open-source: an automated approach (2018).