How Do Community Smells Influence Code Smells?

Fabio Palomba[1], Damian A. Tamburri[2,5], Alexander Serebrenik[2],
Andy Zaidman[3], Francesca Arcelli Fontana[4], Rocco Oliveto[5]
[1]University of Zurich, Switzerland — [2]Eindhoven University of Technology, The Netherlands
[3]Delft University of Technology, The Netherlands — [4]University of Milano-Bicocca, Italy — [5]University of Molise, Italy

ABSTRACT

Code smells reflect sub-optimal patterns of code that often lead to critical software flaws or failure. In the same way, community smells reflect sub-optimal organisational and socio-technical patterns in the organisational structure of the software community.

To understand the relation between the community smells and code smells we start by surveying 162 developers of nine open-source systems. Then we look deeper into this connection by conducting an empirical study of 117 releases from these systems.

Our results indicate that community-related factors are intuitively perceived by most developers as causes of the persistence of code smells. Inspired by this observation we design a community-aware prediction model for code smells and show that it outperforms a model that does not consider community factors.

CCS CONCEPTS

• **Software and its engineering → Software organization and properties**;

KEYWORDS

Code Smells, Organisational Structure, Community Smells

ACM Reference Format:
Fabio Palomba[1], Damian A. Tamburri[2,5], Alexander Serebrenik[2], Andy Zaidman[3], Francesca Arcelli Fontana[4], Rocco Oliveto[5]. 2018. How Do Community Smells Influence Code Smells?. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194950

1 INTRODUCTION

Software engineering is, by nature, a "social" activity that involves organizations, developers, and stakeholders who are responsible for leading to the definition of a software product that meets the expected requirements. The social interactions among the involved actors may represent the key to success as well as a critical issue possibly causing the failure of the project [13].

Recently, the research community has investigated *social debt*, *i.e.,* unforeseen project cost connected to a "suboptimal" development community (*i.e.,* both in structure and behavior) [14]. One of the recent advances in this research field is represented by the definition of *community smells* [14], *i.e.,* a set of socio-technical characteristics (e.g., high formality) and patterns (*e.g.,* recurrent condescending behavior), which may lead to the emergence of social debt. Community smells are connected to code smells [12].

In this paper we aim at deeply and empirically exploring the relation between social and technical debts, in particular on how *community smells* influence the maintainability of *code smells* [3, 8]. We follow a convergence mixed-methods approach [4], in which quantitative and qualitative research are run in parallel over the same dataset aiming at converging towards a confirmed theory.

Our main results indicate that presence of community smells hinders refactoring of the code smells. In particular, we identify occurrence of two known and four previously unknown community smells; and show that community smells are relevant for all code smells and are often more relevant than traditional community-related measures such as socio-technical congruence.

2 SURVEYING DEVELOPERS

To understand the concerns affecting developers' decisions to eliminate or preserve code smells we consider 117 releases of 9 open-source projects from Apache (Mahout, Cassandra, Lucene, Cayenne, Pig, Jackrabbit and Jena) and Eclipse (CDT and CFX). Using DECOR [8] we identify the occurrence of five popular code smells (Long Method, Feature Envy, Blob Class, Spaghetti Code and Misplaced Class) and contact those developers that worked the most (in terms of commits) with one distinct smelly class instance in any of the releases. 162 developers responded out of the 472 we have contacted, for a response rate of 34,32%—that is almost twice than what has been achieved by previous papers (*e.g.,* [2, 9, 16, 17]).

We have shown the respondents the smelly code fragments and asked them for the reasons or risks that lead them to decide whether to refactor the smell. Two authors independently coded the answers.

Survey data highlights the relation between community smells and code smells. 80% of practitioners explicitly mention that avoiding community problems, is the reason why code smells are **not** addressed, meaning that it is more convenient to keep a technical smell than dealing with a community smell. We also confirm in open-source communities the presence and impact of 2 previously reported community smells in industry [5]. At the same time, we observe 4 new community smells that afflict open-source communities and strongly relate to code smells. The new community smells pertain to the inability to achieve consensus, fear that refactoring would be detrimental for understandability, highly complex classes that can be managed by 1-2 people at most and changes causing a previously existing group of modularised collaboration to fragment.

3 STATISTICAL MODELING

We perform statistical modeling of the relationship between community and code smells. We consider the same software systems

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194950

Table 1: Metrics contributing more than the threshold of 0.10 are highlighted in bold, community smells — in *italic*.

Long Method			Feature Envy			Blob			Spaghetti Code			Misplaced Class		
Metric	Gain	SK-ESD Likelihood	Metric	Gain	SK-ESD Likelihood	Metric	Gain	SK-ESD Likelihood	Metric	Gain	SK-ESD Likelihood	Metric	Gain	SK-ESD Likelihood
LOC	**0.83**	**94**	**CBO**	**0.87**	**87**	**Period Commits**	**0.74**	**92**	**LOC**	**0.77**	**85**	***Organizational Silo***	**0.85**	**91**
Churn	**0.68**	**85**	**Previous Intensity**	**0.79**	**83**	**Previous Intensity**	**0.72**	**81**	**Churn**	**0.73**	**83**	**ST-Congruence**	**0.76**	**89**
Previous Intensity	**0.68**	**84**	**Churn**	**0.76**	**78**	**ST-Congruence**	**0.72**	**79**	**Previous Intensity**	**0.71**	**77**	**Previous Intensity**	**0.61**	**77**
Period-Commits	**0.67**	**82**	***Lone Wolf***	**0.59**	**76**	**CBO**	**0.69**	**72**	**Period Commits**	**0.65**	**71**	***Black-Cloud***	**0.55**	**75**
CS-Persistence	**0.55**	**78**	**ST-Congruence**	**0.56**	**71**	**Churn**	**0.65**	**75**	***Lone Wolf***	**0.56**	**66**	**CBO**	**0.45**	**61**
Black-Cloud	**0.55**	**75**	**LOC**	**0.55**	**70**	**Project Tenure**	**0.65**	**65**	**CS-Persistence**	**0.45**	**63**	**Committers**	**0.35**	**59**
Clones	**0.46**	**67**	**Clones**	**0.53**	**64**	**LOC**	**0.57**	**59**	**Ratio Core-Periphery**	**0.38**	**61**	**Period Commits**	**0.33**	**55**
Organizational Silo	**0.27**	**66**	**Project Tenure**	**0.39**	**53**	***Organizational Silo***	**0.45**	**57**	***Organizational Silo***	**0.24**	**56**	**Ratio Core-Periphery**	**0.32**	**51**
ST-Congruence	**0.18**	**59**	**Truck Factor**	**0.23**	**41**	**Truck Factor**	**0.38**	**51**	**Project Tenure**	**0.23**	**45**	**Truck Factor**	**0.24**	**45**
Turnover	**0.17**	**43**	**Period Commits**	**0.15**	**16**	***Bottleneck***	**0.34**	**50**	**Smelly Quitters**	**0.22**	**41**	**Smelly Quitters**	**0.17**	**42**
Commit Tenure	**0.15**	**35**	**Smelly Quitters**	**0.12**	**13**	**Clones**	**0.27**	**44**	**CBO**	**0.13**	**41**	**CS-Persistence**	**0.14**	**38**
Ratio Core-Periphery	**0.12**	**27**	**Committers**	**0.12**	**11**	**Committers**	**0.24**	**41**	**Total Commits**	**0.12**	**34**	**Churn**	**0.12**	**31**
CBO	**0.12**	**24**	*Bottleneck*	0.09	8	**Turnover**	**0.23**	**33**	***Bottleneck***	**0.12**	**31**	**Project Tenure**	**0.12**	**24**
Fault-proneness	**0.12**	**21**	*Organizational Silo*	0.08	7	**Smelly Quitters**	**0.16**	**27**	Committers	0.09	5	*Bottleneck*	0.09	11
Lone Wolf	0.09	8	Fault-proneness	0.05	4	**Fault-proneness**	**0.14**	**19**	Truck Factor	0.08	4	Total Commits	0.08	8
Total Commits	0.08	8	Turnover	0.05	4	Ratio Core-Periphery	0.09	9	Commit Tenure	0.08	4	Clones	0.08	8
Bottleneck	0.07	6	Ratio Core-Periphery	0.04	2	*Black Cloud*	0.04	5	Fault-proneness	0.07	3	*Lone Wolf*	0.07	5
Committers	0.07	4	Total Commits	0.03	2	*Lone Wolf*	0.03	3	Clones	0.06	2	Turnover	0.07	3
Truck Factor	0.05	2	CS-Persistence	0.03	2	CS-Persistence	0.03	2	Turnover	0.05	1	LOC	0.02	1
Smelly Quitters	0.04	2	*Black Cloud*	0.02	1	Total Commits	0.01	1	*Black Cloud*	0.04	1	Commit Tenure	0.02	0
Project Tenure	0.02	1	Commit Tenure	0.01	0	Commit Tenure	0.01	1	ST-Congruence	0.03	1	Fault-proneness	0.01	0

and the code smells as above. Using the CodeFace4smells tool, a fork of CodeFace [6], we detect the following community smells: Organisational Silo Effect, Black-cloud Effect, Lone-wolf Effect, and Bottleneck Effect [14]. We also include in the model traditional metrics such as LOC, Churn or CBO and community-related variables such as socio-technical congruence. As the dependent variable we consider the code smell intensity, i.e., how much the value of a chosen metric exceeds a given threshold [7]. We use *Gain Ratio Feature Evaluation* [10] to rank the independent variables and the Scott-Knott Effect Size Difference (SK-ESD) [15] to verify the ranks. We report the likelihood that a feature was ranked at the top.

Looking at the results in Table 1, we observe that for all the code smells but Misplaced Class traditional metrics such as LOC, CBO, and Code Churn influence the code smell intensity most. These results were quite expected given the importance of LOC for evolution of code smells [11]. Also the previous intensity of a smell is important: likely, this is caused by developers rarely refactoring code smells [1], thus increasing the intensity of smells over time.

Turning the attention to the community smells, we observed that they represent relevant features *for all* the considered code smells. Note that the community-related control factors (*e.g.,* socio-technical congruence) are often not able to explain the dependent variable better than community smells, meaning that the community smells are more powerful "predictors" than such control factors. For instance, the Organisational Silo provides a higher entropy reduction (0.27) than socio-technical congruence (0.18) when considering the Long Method smell. In the second place, we note that different community smells are related to different code smells, meaning that circumstances occurring within a community influence the maintainability of code smells. For example, in the case of Long Method, such circumstances are related to non-structured or missing communication between developers, as highlighted by Black Cloud and Organisational Silo being highly influential.

4 SUMMARY

The organisational and technical structures of software are deeply interconnected. Similarly, we conjectured that the social and technical debt may be connected just as deeply. In this paper, we start exploring this relation from the manifestations of both social and technical debt, i.e., code and community smells. While previous work offered evidence that these two phenomena occurring in software engineering *may* be correlated, in this paper using a mixed-method empirical evaluation we provide evidence of this relation.

REFERENCES

[1] Bavota, G., De Lucia, A., Di Penta, M., Oliveto, R., and Palomba, F. An experimental investigation on the innate relationship between quality and refactoring. *J. Syst. Softw. 107*, C (Sept. 2015), 1–14.

[2] Bavota, G., Linares Vásquez, M., Bernal-Cárdenas, C. E., Di Penta, M., Oliveto, R., and Poshyvanyk, D. The impact of API change- and fault-proneness on the user ratings of android apps. *TSE 41*, 4 (April 2015), 384–407.

[3] Di Nucci, D., Palomba, F., Tamburri, D. A., Serebrenik, A., and De Lucia, A. Detecting code smells using machine learning techniques: Are we there yet? In *SANER* (March 2018), IEEE.

[4] Di Penta, M., and Tamburri, D. A. Combining quantitative and qualitative studies in empirical software engineering research. In *ICSE (Companion Volume)* (2017), S. Uchitel, A. Orso, and M. P. Robillard, Eds., ACM, pp. 499–500.

[5] Jiménez, M., and Piattini, M. Problems and solutions in distributed software development: A systematic review. In *SEAFOOD* (2009), pp. 107–125.

[6] Joblin, M., Mauerer, W., Apel, S., Siegmund, J., and Riehle, D. From developer networks to verified communities: A fine-grained approach. In *ICSE* (2015), A. Bertolino, G. Canfora, and S. G. Elbaum, Eds., IEEE, pp. 563–573.

[7] Marinescu, R. Assessing technical debt by identifying design flaws in software systems. *IBM Journal of Research and Development 56*, 5 (Sept 2012), 9:1–9:13.

[8] Moha, N., and Guéhéeuc, Y.-G. Decor: a tool for the detection of design defects. In *ASE* (2007), R. E. K. Stirewalt, A. Egyed, and B. Fischer, Eds., ACM, pp. 527–528.

[9] Palomba, F., Bavota, G., Di Penta, M., Oliveto, R., Poshyvanyk, D., and De Lucia, A. Mining version histories for detecting code smells. *TSE 41*, 5 (May 2015), 462–489.

[10] Quinlan, J. R. Induction of decision trees. *Mach. Learn. 1*, 1 (Mar. 1986), 81–106.

[11] Tahmid, A., Nahar, N., and Sakib, K. Understanding the evolution of code smells by observing code smell clusters. In *SANER* (March 2016), vol. 4, pp. 8–11.

[12] Tamburri, D. A., and Di Nitto, E. When software architecture leads to social debt. In *WICSA* (2015), L. Bass, P. Lago, and P. Kruchten, Eds., IEEE, pp. 61–64.

[13] Tamburri, D. A., Kazman, R., and Fahimi, H. The architect's role in community shepherding. *IEEE Software 33*, 6 (2016), 70–79.

[14] Tamburri, D. A., Kruchten, P., Lago, P., and van Vliet, H. Social debt in software engineering: insights from industry. *J. Internet Services and Applications 6*, 1 (2015), 10:1–10:17.

[15] Tantithamthavorn, C., McIntosh, S., Hassan, A. E., and Matsumoto, K. An empirical comparison of model validation techniques for defect prediction models. *TSE 43*, 1 (Jan. 2017), 1–18.

[16] Vasilescu, B., Filkov, V., and Serebrenik, A. Perceptions of diversity on github: A user survey. In *CHASE* (2015), IEEE, pp. 50–56.

[17] Vasilescu, B., Posnett, D., Ray, B., van den Brand, M. G. J., Serebrenik, A., Devanbu, P. T., and Filkov, V. Gender and tenure diversity in github teams. In *CHI* (2015), B. Begole, J. Kim, K. Inkpen, and W. Woo, Eds., ACM, pp. 3789–3798.