Understanding Community Smells Variability: A Statistical Approach

Gemma Catolino[1,2], Fabio Palomba[3], Damian Andrew Tamburri[2,4], Alexander Serebrenik[4]
[1]Tilburg University, NL - [2]Jheronimus Academy of Data Science, NL
[3]University of Salerno, IT - [4]Eindhoven University of Technology, NL
g.catolino@tilburguniversity.edu, fpalomba@unisa.it, d.a.tamburri@tue.nl, a.serebrenik@tue.nl

***Abstract*—Social debt has been defined as the presence in a project of costly sub-optimal organizational conditions, *e.g.*, non-cohesive development communities whose members have communication or coordination issues. Community smells are indicators of such sub-optimal organizational structures and may well lead to social debt. Recently, several studies analyzed actors affecting presence of community smells and their harmfulness, or proposed refactoring strategies to mitigate them. However, to the best of our knowledge, there is still a limited understanding of the factors influencing the variability of community smells, namely how they increase/decrease in magnitude over time. In this paper, we aim at conducting the first statistical experimentation on the matter, by analyzing how a set of 40 socio-technical factors, *e.g.*, turnover or communicability, impact the variability of four community smells on a dataset composed of 60 open-source communities. The results of the study reveal that communicability is, in most cases, important to reduce the risk of an increase of community smell instances, while broadening the collaboration network does not always have a positive effect.**

***Index Terms*—Community Smells; Social Debt; Statistical Models; Empirical Study.**

I. Introduction

Developing software is by nature a social activity [1]: the way developers interact between each other not only has implications on social debt [2] but also on software code quality [3]. Communication and collaboration challenges can arise because of multiple reasons, such as different cultures and culture clashes [4], [5] and differences in expertise or power distance [6], [7]. In these circumstances, sub-optimal organizational and socio-technical situations that hamper or altogether impede the straightforward production, operation, and evolution of software [8] may occur: research has named them *community smells* [9].

The research on community smells has gained attention over the last years. Recent work has established the interaction between community smells and their technical counterparts [10] as well as, more generally, technical debt in its various forms [11], [12]. Moreover, researchers have been investigating how community smells appear [2], [13], what are the major factors that may influence community smells [13]–[15], and how developers refactor community smells in practice [16]. For instance, Tamburri *et al.* [13] showed that community smells are highly diffused and perceived harmful, and some existing socio-technical factors correlate with their presence. Moreover, in certain cases the emergence of community smells may be potentially reduced by increasing gender diversity and women participation [14], [15], even though there exist specific actions that can be performed to modify the structure of a community and get rid of these smells [16].

Despite the relevant body of knowledge built so far, there is still a limited understanding of whether and which are the socio-technical factors, such as turnover [17]–[19] or communicability [20], contributing to the *variability* of community smells: we define *variability* as the extent to which community smell instances increase or decrease over time. An improved understanding of this aspect is crucial for two main reasons: (1) it would be possible to keep known socio-technical factors affecting such a variability under control, possibly estimating how changes in these factors would impact the emergence of issues in the community; (2) it would be possible to derive additional aspects influencing the phenomenon, hence providing the research community with insights on the future steps to take to deal with community smells.

For the aforementioned reasons, in this paper we study how 40 previously identified socio-technical factors contribute to the variability of four well-known and harmful community smells, *i.e., Organizational Silo*, *Black Cloud*, *Lone Wolf*, and *Radio Silence*, considering the dynamics of 60 open-source communities. In this study we adopt a statistical approach: the idea is to shed lights on the factors that influence the phenomenon with enough statistical significance to support further research based on the findings of our study. The key results report that communicability metrics are those that in most cases increase the chances of community smells being stable over time, while increasing the collaboration network does not always lead to a reduction of community smells. Based on these results, we formulate a number of research avenues and challenges to tackle the risks associated to the variability of community smells.

To sum up, our paper offers three main contributions:

1) The first, large-scale empirical exploration of the variability of four community smells based on a set of 40 socio-technical metrics;
2) A research roadmap that other researchers can use to derive the next challenges in the field of community smell prevention and identification;
3) A publicly available replication package [21] supporting further studies of the characteristics of community smells as well as corroborating our findings.

II. Related Work

Effective communication and organization of a software development team might influence the quality of both development process and software created [22]. Costs of poor communication are estimated as $37 billion.[1] This motivated the research on "social debt", *i.e.,* the presence of non-cohesive development communities whose members have communication or coordination issues [23]. One of most harmful and recurring forms of social debt has the name of *community smells*, which represent suboptimal socio-technical patterns appearing within a software community and causing coordination and/or communication problems that not only increase social debt but may even lead to the emergence of technical debt [10].

Community smells started receiving particular interest since the release of CodeFace [24]. By mining code repositories and developer's interactions, the tool can first construct the so-called *developer networks*, namely graphs describing the relations of collaboration and communication actually in place among developers, and then build upon these networks to automatically identify software communities. The availability of this tool has naturally allowed further research in the field.

Tamburri *et al.* [13] extended CodeFace to enable the automatic identification of the four community smells selected for our study. The resulting tool, named CodeFace4Smells, has been empirically evaluated on a set of 60 projects: according to the findings, the tool does not output false positives/negatives and the community smells identified match in 100% of the cases the developer's expectations. As explained in Section III, the availability of this detection tool allowed us to identify community smells in the context of this work.

Later on, Palomba *et al.* [10] showed that community smells are among the top factors influencing the emergence of code smells in source code. Along the same line, other researchers focused on establishing the impact of community smells on other dimensions of software engineering, *e.g.,* architecture debt [11] and organization structure types [25].

Researchers have been also investigating how these smells appear, what are the major factors that may influence them, and the way developers refactor community smells in practice. Catolino *et al.* [14] showed how in certain situations the emergence of community smells may be potentially reduced by increasing gender diversity. A further survey study of the practitioners' opinions [15] reported that developers do not perceive gender diversity and presence of women in software teams as relevant factors to avoid community smells, while they believe that other aspects, like developer's experience or team size, may make a community more prone to be smelly [15]. Catolino *et al.* [16] also studied how developers remove community smells: by surveying 76 experts, the authors were able to elicit and distill a set of refactoring operations generally applied by practitioners to remove the four community smell types identifiable through CodeFace4Smells.

A more recent and promising trend is represented by the definition of modeling mechanisms able to describe the future structure of a community and alert project managers of the presence of social debt [26]–[28].

The closest work to the one presented in this paper is the paper by Tamburri *et al.* [13]. They performed a large-scale empirical study on 60 open-source ecosystems to evaluate the diffuseness of community smells, how they are perceived and how smells relate to existing socio-technical factors. They found that community smells are highly diffused in open-source and are perceived by developers as relevant problems for the evolution and sustainability of software communities. Finally, they performed a correlation analysis between some socio-technical metrics and the presence of community smells, identifying a number of positive correlations. With respect to this paper, ours must be considered as complementary: indeed, we do not aim at finding simple correlations between metrics and community smells, but providing a finer-grained view of how existing socio-technical metrics influence the *variability* of community smells. The variability perspective is, therefore, a novelty of this paper along with the analysis of how socio-technical metrics impact the increment/decrement of the number of different types of community smells.

III. Research Methodology

This section describes the methodology of our study, which followed the guidelines of Wohlin *et al.* [29].

A. Research Questions

The *goal* of the study is to analyze community smells with the *purpose* of investigating whether and which are the socio-technical metrics that affect their variability. The *quality focus* is on community smells and their variability within software projects. The *perspective* is of both researchers and practitioners: the former are interested in gathering a deeper understanding of the variability of community smells and their connected causes, while the latter in better monitoring the presence of community smells using socio-technical metrics.

Based on these objectives, we asked:

RQ$_1$. *How do socio-technical metrics affect the variability of community smells?*

With **RQ$_1$**, we analyzed whether and how existing socio-technical metrics may affect the variability of community smells, namely their increment or decrement over time. In so doing, we adopted a statistical approach: in other words, we sought to identify the relevant factors for the variability of community smells by defining a statistical model relating socio-technical metrics to the increment/decrement of community smells. It is important to point out that the research questions also allowed us to establish commonalities and differences among different community smells.

B. Context Selection

The *context* of the study consisted of software projects, socio-technical factors, and community smells.

[1]https://tinyurl.com/ydfjf4p6

TABLE I
SOCIO-TECHNICAL METRICS CONSIDERED IN OUR STUDY.

Category	Metric	Description
Developer Social Network metrics	devs	Number of developers present in the global Developers Social Network
	ml.only.devs	Number of developers present only in the communication Developers Social Network
	code.only.devs	Number of developers present only in the collaboration Developers Social Network
	ml.code.devs	Number of developers present both in the collaboration and in the communication DSNs
	perc.ml.only.devs	Percentage of developers present only in the communication Developers Social Network
	perc.code.only.devs	Percentage of developers present only in the collaboration Developers Social Network
	perc.ml.code.devs	Percentage of developers present both in the collaboration and in the communication DSNs
	sponsored.devs	Number of sponsored developers (95% of their commits are done in working hours)
	ratio.sponsored	Ratio of sponsored developers with respect to developers present in the collaboration DSN
Socio-Technical Metrics	st.congruence	Estimation of socio-technical congruence
	communicability	Estimation of information communicability (decisions diffusion)
	num.tz	Number of timezones involved in the software development
	ratio.smelly.devs	Ratio of developers involved in at least one Community Smell
Core community members metrics	core.global.devs	Number of core developers of the global Developers Social Network
	core.mail.devs	Number of core developers of the communication Developers Social Network
	core.code.devs	Number of core developers of the collaboration Developers Social Network
	sponsored.core.devs	Number of core sponsored developers
	ratio.sponsored.core	Ratio of core sponsored developers with respect to core developers of the collaboration DSN
	global.truck	Ratio of non-core developers of the global Developers Social Network
	mail.truck	Ratio of non-core developers of the communication Developers Social Network
	code.truck	Ratio of non-core developers of the collaboration Developers Social Network
	mail.only.core.devs	Number of core developers present only in the communication DSN
	code.only.core.devs	Number of core developers present only in the collaboration DSN
	ml.code.core.devs	Number of core developers present both in the communication and in the collaboration DSNs
	ratio.mail.only.core	Ratio of core developers present only in the communication DSN
	ratio.code.only.core	Ratio of core developers present only in the collaboration DSN
	ratio.ml.code.core	Ratio of core developers present both in the communication and in the collaboration DSNs
Turnover	global.turnover	Global developers turnover with respect to the previous temporal window
	code.turnover	Collaboration developers turnover with respect to the previous temporal window
	core.global.turnover	Core global developers turnover with respect to the previous temporal window
	core.mail.turnover	Core communication developers turnover with respect to the previous temporal window
	core.code.turnover	Core collaboration developers turnover with respect to the previous temporal window
	ratio.smelly.quitters	Ratio of developers previously involved in any Community Smell that left the community
Social Network Analysis metrics	closeness.centr	SNA degree metric of the global DSN computed using closeness
	betweenness.centr	SNA degree metric of the global DSN computed using betweenness
	degree.centr	SNA degree metric of the global DSN computed using degree
	global.mod	SNA modularity metric of the global DSN
	mail.mod	SNA modularity metric of the communication Developers Social Network
	code.mod	SNA modularity metric of the collaboration Developers Social Network
	density	SNA density metric of the global Developers Social Network

As for the former, our study targeted 60 active open-source software projects extracted from GITHUB. In particular, we relied on the publicly available dataset provided by Tamburri *et al.* [13]. The software systems in such a dataset have been originally selected based on various criteria: (1) *code-base size* - the dataset contains 20 medium-sized (200–500 KLOC), 20 large (500–850 KLOC) and 20 very large (> 850 KLOC) systems; (2) *programming language* - it contains projects developed using Java, C#, C, Python, YAML and other languages; (3) *community size* - the dataset contains equal numbers of projects with *i.e.,* medium (<50 members), large (50–150) and very-large (>150) communities; (4) *age* - the distribution in this case is evenly split among three ranges, *i.e.,* young projects (<24 months), established projects (24–32), and mature projects (>32). Due to the space limitation, the complete list of the projects is available in our online appendix [21]. More details on the project selection and data extraction are available in the paper by Tamburri *et al.* [13].

Table I reports the socio-technical metrics considered in the scope of this paper. As shown, we took into account factors covering five different aspects characterizing a software community, *e.g.,* turnover metrics that measure, under different perspectives, the extent to which developers join and leave a project. Their selection was based on two key points. In the first place, all the metrics have been previously correlated to the presence of community smells [13]: as such, we selected these metrics to enlarge our knowledge on the matter and verify the role they can play when assessing the variability of community smells. In the second place, these metrics are broadly and widely connected to social debt, as established by previous work (*e.g.,* [20], [24], [41]: for the sake of comprehensiveness, Table II summarizes the insights provided by researchers in the past with respect to the relation between the selected socio-technical metrics and community smells. Hence, our selection procedure allowed us to analyze a wide range of relevant metrics and provide additional insights into their potential practical usefulness.

Finally, as for the community smells, we focused on:

1) ORGANIZATIONAL SILO EFFECT: This form of social debt refers to the presence of siloed areas of the developer community that do not communicate, except through one or two of their respective members;

TABLE II
METRICS SELECTION RATIONALE; METRICS AND RATIONALES FOLLOW THE SAME CLUSTERING SCHEMA FROM TABLE I.

Metric	Rationale
Developer Social Network Metrics	These metrics are largely derived from the works of Meenly *et al.* [20], [30] and Joblin *et al.* [24] who, respectively, (1) introduced the fundamental roles of Developer Social Networks in predicting failures and (2) prototyped technologies for the verifiable investigation of fully-formed developer communities. DSN metrics are not only relevant to specify the stability and predictive characteristics of organisational structures but also that they reflect relevant technical characteristics in code which reflect socio-technical issues.
Socio-Technical Metrics	These metrics are derived from the state-of-the-art in social software engineering. Previous research highlighted the relation between these metrics and sub-optimal conditions in the organisational structure,*e.g.,* Kwan *et al.* [31] who investigate build failures connected to sub-optimal coordination patterns or Tamburri [32] who studies sub-optimal architectural decision patterns connected to incommunicability.
Core-Community Members Metrics	Metrics in this cluster range between the truck-factor (whose relation to sub-optimal organizational conditions is highlighted in [33], [34]) to core-periphery numbering metrics, which are largely inspired by the work of Manteli *et al.* [35], [36] who investigate core-periphery organizational (anti-)patterns and their relation with community member descriptors as well as core-periphery mismatches.
Turnover Metrics	These metrics are derived directly from the original committers of the studies reported in [10], [13], [32] who desired to investigate their turnover rates across several distributed software development sites across the world.
Social-Network Analysis Metrics	These metrics are derived from the state of the art in organisations' research and reflect the most relevant centrality measures recurrently used in connection to sub-optimal organisational conditions such as prima-donna effects (e.g., see Dekker et al. [37]) or lack of boundary-spanning [38]–[40].

2) BLACK CLOUD EFFECT: This community smell reflects an information overload due to lack of structured communications or cooperation governance;
3) LONE WOLF EFFECT: This smell arises when the development community presents unsanctioned or defiant contributors who carry out their work with little consideration of their peers, their decisions and communication;
4) RADIO-SILENCE EFFECT: This smell appears when one member interposes herself into every formal interaction across two or more sub-communities with little or no flexibility to introduce other parallel channels.

These and additional community smell types have been originally introduced by Tamburri *et al.* [42]. We opted for these four smell types since they have been shown to have a strong negative impact on both social aspects of software communities [8], [9] and the resulting technical processes and products [10], [14], [15]. Moreover, the relevance of these community-related issues has been previously investigated in open-source [13], and in our previous studies [14]–[16], thus, their analysis could lead to confirm or refine the available findings. They have been extracted using the CODEFACE4SMELL tool [13]. It is important to point out that the dataset we relied on provides the set of community smells for each three-month time window from the conception of the systems along with the socio-technical metrics computed in the same time windows: this detail is key to enable our study, since we could compute how the instances of the four considered community smell types varied over time as well as how the socio-technical metrics impacted their variability.

C. Building the Statistical Model

To address our research question, we defined a statistical model relating the considered set of socio-technical metrics presented to community smells, defined as follows.

Response Variable. We were interested in explaining the variability of community smells. To this aim, we modeled the response variable similarly to the previous study of Palomba *et al.* [43]. For each time window pair (TW_i, TW_{i+1}) of a project, we first computed the difference between the number of community smells in TW_{i+1} and TW_i: if the resulting difference was higher than 0, this means that the number of community smells from TW_i to TW_{i+1} was increased, hence we labeled the event as an *"increase"*; if the difference was negative, then we labeled the case as a *"decrease"*; otherwise, the event was labeled as a *"stable"*. These three labels represented the response variable of our model. It is worth noting that the labeling process was repeated four times, one for each community smell considered, *i.e.,* we ended up with the construction of one model for each smell, so that we could assess the differences among them.

Independent Variables. These are represented by the 40 socio-technical metrics introduced in Section III-B.

Control Variable. The variability of community smells may depend on the considered socio-technical metrics, but also on some intrinsic peculiarities of the projects, *e.g.,* a community might have a code of conduct that promoted standards of ethical behavior [44], hence naturally influencing the variability of community smells. To account for these aspects, we considered the variable *project* as our control factor, *i.e.,* we left the model decide whether the specific project is relevant when explaining the variability of community smells.

D. Data Analysis

Given our dependent variable, which is categorical and can assume value in the set $\{decrease, stable, increase\}$, we fit a Multinomial Log-Linear model [45]. This is a classification method that generalizes logistic regression to multiclass problems and can handle either categorical and continuous independent variables, hence perfectly fitting our case - the control variable is categorical. From a technical perspective, multinomial models were built using `R`, and particularly exploiting the function `multinom` available in the package `nnet`,[2] *i.e.,* the model is fitted via neural networks.

A common step to perform when building a statistical model is to take into account the problem of multicollinearity, which appears when two or more independent variables are highly correlated and can be predicted one from the other, possibly biasing the way the model fits and, consequently, how the results are interpreted. In our work, we have applied the guidelines by Allison [46], that describe how to control a model for multicollinearity and when to ignore it: as a result, we did not remove any of the variables. This is because, as reported by Allison [46], the standard errors of the independent variables are narrow enough not to negatively influence the interpretability of the model: in our case, for all models they are lower than 0.9—note that standard errors must be ≤ 2.5 to produce a sufficiently narrow 95% prediction interval [47].

When it comes to the interpretation of the model, the logit coefficients that the model outputs are relative to a reference category and indicate how the independent variables change the chances of the dependent variable being affected with respect to the reference category. We set such a category to *"stable"*: in this way, we could understand how the different independent variables vary, in either positive or negative manner, the likelihood of community smells being stable over two time windows. It is worth remarking that a negative logit coefficient for a variable suggests that for one unit increase of that variable, the chances of variation of the response variable are increased of the amount indicated by the coefficient—in other words, the coefficients must be inversely interpreted. As an example, suppose that the variable *devs* has a logit coefficient of -0.02 in the statistical model built for *Black Cloud*: this would mean that a one-unit increase of *devs* would lead to an increase of the chances of *Black Cloud* being stable.

E. Replication Package

We release all scripts and data used ro tun the four models. These were made available in our online appendix [21].

IV. Analysis of the Results

This section reports the results to $\mathbf{RQ}_1$ for each community smell, We report only the statistically significant variables—full results are available in our online appendix [21].

A. Results for Organization Silo

Table III reports the significance of the considered socio-technical factors with respect to the variability of *Organization Silo* instances in our dataset. For the sake of space limitation, the table only reports the factors that appeared to be statistically significant - the complete results are available in our online appendix [21].

As shown in the table, the first two significant factors are *perc.code.only.devs* and *perc.ml.code.devs*, which respectively represent the percentage of developers only present in the collaboration network of the project's members and the percentage of developers present in both communication

[2]Link: https://cran.r-project.org/web/packages/nnet/nnet.pdf

TABLE III
Results achieved by the model built for Organizational Silo. * *means* $p < 0.1$; **—$p < 0.05$; ***—$p < 0.01$.

	Dependent Variable	
Factors	**Decrease**	**Increase**
perc.code.only.devs	-4.816***	-4.926***
perc.ml.code.devs	7.593***	3.753***
ratio.sponsored	1.872**	
ratio.sponsored.core	-4.157***	-1.633**
core.global.devs	0.486***	0.441***
core.mail.devs	-0.261***	-0.240***
st.congruence	2.708***	
communicability	-10.101***	-8.997***
code.turnover		-0.203**
core.code.turnover		0.297**
global.truck	12.894***	12.732***
mail.truck	-5.099***	-5.134***
code.truck	-1.274*	3.220***
closeness.centr	-2.187**	-1.664*
betweenness.centr	-2.823***	-1.793**
degree.centr	2.851***	
mail.mod	-1.818***	
code.mod	-1.415**	
density	-8.369***	-1.417**
mail.only.core.devs	-0.224***	-0.210***
ratio.mail.only.core	2.445***	1.155***
ratio.code.only.core	-2.093***	-3.967***
ratio.ml.code.core	3.285***	2.741***

and collaboration networks. As for *perc.code.only.devs*, we observe that the logit coefficients are negative either when considering the decrease and increase of community smells over different time windows. This indicates that a one-unit increase of the metric leads to an increase of the chances of *Organization Silo* being stable. In more practical terms, the more developers are involved in the collaboration the lower the risk of variability in the number of *Organization Silo* instances. This result was somehow expected, since the smell has to do with a lack of communication/collaboration between developers. As such, community shepherds may monitor this factor, possibly involving more developers into collaboration, to reduce the emergence of issues caused by the presence of siloed areas of the community. As for *perc.ml.code.devs*, the results report something different, namely that a one-unit increase of this factor decreases the chances of *Organization Silo* being stable. While this seems to contradict the finding of *perc.code.only.devs* other than being counter-intuitive, there is a clear motivation for this result. As shown by Singh *et al.* [48], an increase of the collaboration network does not only have positive effects on the structure of a community: indeed, there may be other specific external factors, *e.g.,* the technology diversity, that may worsen the ability of developers to work in a cohesive manner. Our findings not only confirm what discovered by Singh *et al.* [48], but also highlight the need for comprehensive tool-suites that can provide community shepherds with the possibility to consider both internal and external metrics when assessing the implications of enlarging the developer's collaboration network.

The *core.global.devs* positively influences the variability of *Organizational Silo*. This metric refers to the number of core developers in the developer's social network: as also reported by previous work [49], [50], the availability of a high number of core developers increases the overall awareness of the community as well as its ability to jointly work toward the success of the project. In our case, we discovered that this aspect also impacts the variability of *Organizational Silo* instances. Additionally, we can confirm the importance of sponsored developers (characterized by the variable *ratio.sponsored.core*) [51], who help increase the cohesion of the community.

According to the statistical results, other socio-technical metrics reduce the chances of emergence of *Organizational Silo*. Among them, it is worth mentioning the *st.congruence*, which measures the agreement between social and technical organization of the work [52], as well as the communicability, that is an indicator of the extent to which the information is shared among the team members. These metrics can be used, along with the others, as a means though which community shepherds can take informed decisions on how to (re-)structure the community to avoid the emergence of information losses typical of the *Organizational Silo*.

Main findings for *Organizational Silo*

Our findings report that there exist several factors whose monitoring may be helpful to reduce the likelihood of a community to be affected by the *Organizational Silo* smell. As expected, these mainly relate to communication-related metrics. However, our results also report that an increase of the collaboration network may be detrimental in terms of social debt. As such, we argue the need for (automated) instruments able to balance communication and collaboration aspects to optimize the community health.

B. Results for Lone Wolf

The results achieved when analyzing the *Lone Wolf* smell are in Table IV. Also in this case, we can observe that there are a number of aspects affecting the variability of the smell. Nevertheless, some metrics appear to be more relevant than others. The first interesting observation can be done by considering the coefficients for *perc.ml.code.devs*: these are negative, thus meaning that an increase of the developers in both communication and collaboration network leads to increase the chances of stability of this smell—conversely to what discovered in the case of *Organizational Silo*.

The explanation of this result relates to the peculiarities of the *Lone Wolf* smell. Since it highlights the presence of defiant contributors who carry out their work with little or no consideration of their peers, it seems clear that its effects or even emergence can be mitigated by means of a more inclusive community. In this sense, an increase of the *perc.ml.code.devs* metric may have benefits with respect to this smell.

The importance of having sponsored developers within the community—as indicated by the *ratio.sponsored.core* metric—is confirmed. This finding possibly sheds lights on a new perspective of the involvement of commercial developers into open-source projects: besides significantly contributing to the

TABLE IV
RESULTS ACHIEVED BY THE MODEL BUILT FOR LONE WOLF. NOTE THAT * *means* $p < 0.1$; **—$p < 0.05$; ***—$p < 0.01$.

	Dependent Variable	
Factors	**Decrease**	**Increase**
perc.ml.only.devs	2.270***	1.687**
perc.code.only.devs	-3.221***	-3.092***
perc.ml.code.devs	5.017***	3.716***
ratio.sponsored	6.739***	5.721***
sponsored.core.devs	0.488*	0.463*
ratio.sponsored.core	-13.324***	-11.435***
core.global.devs	0.191*	
st.congruence	-5.096***	-2.970***
communicability	-3.194***	-3.385***
code.turnover	-0.229**	-0.312***
core.code.turnover		0.372***
ratio.smelly.quitters		-1.590**
global.truck	7.797***	5.798***
mail.truck	-6.697***	-4.448***
code.truck	-2.567***	1.519**
closeness.centr	-2.738***	-3.319***
betweenness.centr	-1.581**	
degree.centr	3.673***	
global.mod	-1.516**	-2.445***
density	-4.739***	
mail.only.core.devs	-0.187***	-0.186***
code.only.core.devs	-0.234***	-0.209***
ml.code.core.devs	0.170***	0.185***
ratio.mail.only.core	-0.940*	-0.748**
ratio.code.only.core	2.038***	1.149*
ratio.ml.code.core	3.325***	2.187***

creation and evolution of a social capital [53], it seems that they may lead the community to be more cohesive, hence damping the emergence of situations where individual contributors start working independently.

It is also worth commenting the results of *global.truck*, *mail.truck*, and *code.truck*: the first refers to the ratio of non-core developers of the global developer's social network, the second to the ratio of non-core developers of the communication network, the latter to the ratio of non-core developers of the collaboration network. In all cases, the coefficients are negative and indicate that their increase may lead to a stability in terms of *Lone Wolf* emergence. These results somehow reinforce the conclusions drawn so far: indeed, involving more developers, including non-core ones, into the community reduces the risk of having lone wolfs. Finally, our findings also suggest that a keeping a good amount of core developers (see the coefficients of *ratio.code.only.core*) provide an additional mitigation strategy for the emergence of *Lone Wolf* instances.

Main findings for *Lone Wolf*

We argue that the stability of *Lone Wolf* strictly depends on the community structure implemented and, particularly, on its cohesiveness. Some preliminary studies into these aspects have been conducted [2], [14], [25], yet further analyses aiming at understanding and/or devising the mechanisms that community shepherds should employ to keep a community cohesive and inclusive may be worthy.

TABLE V
RESULTS ACHIEVED BY THE MODEL BUILT FOR BLACK CLOUD. NOTE THAT * *means* $p < 0.1$; **—$p < 0.05$; ***—$p < 0.01$.

	Dependent Variable	
Factors	**Decrease**	**Increase**
code.only.devs	0.012*	
perc.ml.only.devs	-5.745***	-8.191***
perc.code.only.devs		3.438**
perc.ml.code.devs		-3.924**
sponsored.devs		0.042**
ratio.sponsored	-3.866***	-8.627***
ratio.sponsored.core		3.505***
num.tz	-0.064***	-0.034*
core.global.devs		-0.092*
core.code.devs		0.040*
st.congruence		-1.965*
ratio.smelly.quitters	3.199***	
ratio.smelly.devs	3.201**	4.524***
global.truck	-11.587***	-29.948***
mail.truck	19.148***	44.307***
code.truck	-2.472*	
degree.centr		5.424**
global.mod		-2.512*
mail.mod		4.168**
density		-4.104***
ratio.mail.only.core	2.850***	2.216**
ratio.code.only.core	-6.049***	-12.678***
ratio.ml.code.core	-2.396**	

C. Results for Black Cloud

The results achieved when considering the *Black Cloud* community smell are depicted in Table V. They are very much in line with the discussion points raised so far and, indeed, most of the relevant variables are the same as the previous community smells. On the one hand, the results confirm the fact that certain metrics, like the involvement of non-core developers into the community, represent valid measures to use to monitor the health status of a community and take informed decisions on how to evolve it. On the other hand, there are different reasons making the same metrics relevant for a variety of smells; in the following we elaborate on what makes *perc.ml.only.devs*, *global.truck*, and *mail.truck* the most relevant metrics influencing the emergence of *Black Cloud*.

By definition, a community suffers from a *Black Cloud* smell in cases where there is an information overload due to a lack of structured communications. As such, an increase of one-unit of *perc.ml.only.devs* can improve the community with respect to this smell simply because the more developers are involved in the communications, the easier the information is received by all developers hence reducing the risk associated to the emergence of *Black Could* instances. Much in the same way, the results achieved when considering *global.truck* and *mail.truck* indicate that a good mechanism to deal with the smell is to improve the cohesion of the community through the involvement of the developers within communications.

Two new, different points of discussion are given by the *ratio.smelly.devs* and *ratio.smelly.quitters* metrics. The former measures the ratio of developers involved in at least one community smell: our findings show that having developers previously involved in a social debt may lead to a reduction of the chances of *Black Cloud* being stable. In other words, it seems that developers that reiterate behaviors causing communication and/or coordination issues increases the likelihood of introducing information overload that, in turn, can not only produce social debt but also lead to code quality issues [10]. As for the *ratio.smelly.quitters* metric, it measures the ratio of developers previously involved in any community smell that left the community. Also in this case, our analysis reveals that this factor possibly reduces the likelihood of stability of the *Black Cloud* smell, even if with a lower extent (the coefficient stability/increase is slightly negative, *i.e.,*-0.469, but not statistically significant).

Main findings for *Black Cloud*

We can argue, once again, that the involvement of community members into the communication network seems to be a good practice to reduce the risk of incurring in *Black Cloud* instances. Nevertheless, community shepherds should particularly take care of involving members that have been involved in community smells in the past, so that they could avoid reiterating behaviors that may cause new sources of social and technical debt.

TABLE VI
RESULTS ACHIEVED BY THE MODEL BUILT FOR RADIO SILENCE. NOTE THAT * *means* $p < 0.1$; **—$p < 0.05$; ***—$p < 0.01$.

	Dependent Variable	
Factors	**Decrease**	**Increase**
perc.ml.only.devs	-1.762**	-1.346*
perc.code.only.devs	4.448***	4.700***
perc.ml.code.devs	-2.057***	-1.395**
ratio.sponsored	1.802*	
ratio.sponsored.core	3.846***	2.290***
core.global.devs	0.486***	0.441***
core.mail.devs	-0.261***	-0.240***
communicability	5.096***	3.581***
core.global.turnover	-0.443*	
core.mail.turnover	0.738*	
ratio.smelly.quitters	-2.046**	
ratio.smelly.devs	9.852***	1.772**
global.truck	-12.677***	-12.286***
mail.truck	10.084***	13.121***
code.truck	-6.747***	-6.095***
betweenness.centr	-3.289***	-1.535*
degree.centr	1.935**	
global.mod	-3.890***	-3.961***
mail.mod	2.518***	
density	-4.850***	-1.716***
ratio.mail.only.core	4.904***	3.695***
ratio.code.only.core	-4.317***	-2.674***

D. Results for Radio Silence

The results for the *Radio Silence* community smell are reported in Table VI. Most of the significant metrics are in line with those discussed so far. For instance, we confirm that an improvement of key communication aspects, as measured by *perc.code.only.devs*, *perc.ml.code.devs*, *ratio.sponsored.core*, *communicability*, and *code.truck*, should be carefully taken into account when analyzing and addressing the health status of an open-source community. However, some of these metrics have a positive sign, meaning that they negatively affects the stability of the smell. As an example, let us consider the case of *ratio.sponsored.core*: so far we have discussed this factor as relevant to reduce the risk of community smell emergence, while it seems to be detrimental for the stability of *Radio Silence*. One of the possible reasons behind this result falls into the fact that, despite increasing the overall cohesiveness of a community, the presence of sponsored and possibly more authoritative developers may favor the spillage of an authoritarian leadership by some of the community members who may feel the need to express their strong personality within the community [54]. In other words, we can conjecture that the *Radio Silence* smell is different in nature with respect to the others and may be more related to the individual personalities of the developers of a community. As such, our findings seem to suggest the need for additional tools or instruments able to profile developers so that community shepherds may understand how changes in the community can impact the community itself.

In such a context, however, it is worth discussing the two new factors emerging as relevant for this smell, namely *global.mod*, *density*, both contributing in a positive manner to the stability of *Radio Silence*. The first measures the strength of division of the developer's social network into modules: its statistical significance possibly indicates that the creation of cohesive subgroups may reduce the risk of the emergence of *Radio Silence* instances, confirming previous findings in the field [55]. The second measures the proportion of direct ties in the developer's social network. In this case, the positive effect is likely due to the close relationships that are installed between community members: in other words, the results seem to confirm that cohesion exercises could actually lead to the mitigation of this smell [16].

Main findings for *Radio Silence*

We can argue that the *Radio Silence* smell is all but easy to analyze and mitigate since it revolves around aspects that go beyond what can be currently minable, like the personalities of developers. Nevertheless, some preventive refactoring strategies, like a re-structuring of the community in more sub-communities can reduce the risk of appearance of *Radio Silence* instances.

V. Threats to Validity

In this section,we discuss possible threats that could have affected our results and how we mitigated them.

Construct Validity. Threats in this category refer to the correctness of the dataset exploited in the study. We relied on a publicly available source built in the context of previous research [25]; both the independent and response variables used in our models were already available and computed for all the time windows considered in the study. Furthermore, it is worth noting that the response variable was computed by means of the CodeFace4Smells tool: this has been previously validated through a qualitative investigation with developers [25]. According to the results, the community smells output by the tool are all true positives and additional smell instances were not highlighted by developers, hence reducing the risk of having false negatives. This validation makes us confident of the reliability of the response variable.

Conclusion Validity. A major threat to the conclusions drawn is related to the statistical methods employed. The selection of the Multinomial Logistic Linear statistical approach [45] was driven by the fact that our response variable was categorical and composed of three possible values. The statistical approach is able to handle multiclass problems with categorical and continuous independent variables, therefore fitting the problem of interest. In addition, we also take into account the multicollinearity among independent variables: by relying on the guidelines by Allison [46], we verified that the standard errors of the coefficients were narrow enough not to influence the interpretability of the model.

External validity. Threats in this category mainly concern the generalization of results. We analyzed a total of 60 software systems coming from different application domains and having different characteristics (size, programming languages, number of classes, etc.). Of course, we cannot claim the generalizability of the findings to other systems; our future research agenda includes the extension of the study with more different set of systems.

VI. Discussion and Implications

Our results highlighted a number of points to be further discussed and several implications for the research community.

Communication as a key factor for reducing social debt. Our results clearly pointed out the relevance of communicability and other factors related to the way developers communicate among each other to avoid the proliferation of community smells. Despite being expected, we believe that our work provides additional motivations to the research around the identification and definition of novel communication methods to improve developer's productivity. In particular, further research may want to consider the inclusion of predictive mechanisms [26], [28] within communication tools in order to establish how the likelihood of the emergence of community-related issues varies based on the current types/strength of communications among developers.

Increasing collaboration network is not a golden hammer. We noticed a different trend when investigating the role of collaboration on the stability of community smells. While in some cases the increase of the collaboration network seems to provide some benefits (*e.g.,* in the case of *Lone Wolf*), our results report that in other cases it can be detrimental and even lead to increasing the chances of new social debt. On the one hand, it is clear that further investigations, for instance in the form of ethnographic studies, into this matter would be beneficial to more precisely understand this phenomenon and how the increase of collaboration may impact the emergence of community smells. On the other hand, our results also highlighted that external factors may have an impact on collaboration itself: hence, we argue the definition of novel, comprehensive tool boxes that may support community shepherds providing them with both internal and external factors to establish the health status of a community as well as the implications of enlarging it.

Community evolution as a multi-objective problem. The dichotomy between communication and collaboration as well as the influence of other socio-technical factors on the stability of community smells allowed us to indicate the need for brand new automated assistants able to combine together pieces of information coming from different sources in order to recommend the actions to be performed on the community to see it successfully evolved. As an example, we envision the adoption of search-based software engineering methods for the creation of intelligent algorithms that find a compromise between contrasting desirable objectives to achieve within the community.

Learning not to reiterate community smells. Some community smells (*e.g., Black Cloud*) may manifest themselves again as a consequence of the engagement of community members previously involved in a smell. This has clearly to do with the governance and management policies applied in a software community: on the one hand, it is not convenient to isolate developers only because they have been involved in community smells; on the other hand, it is not convenient to create new social debt too. As such, this is again a compromise to consider when taking decisions on the evolution of the community. In this respect, our findings seem to suggest the need for tools or methods that community shepherds can employ to assess the impact of involving individual developers into communication and collaboration networks. At the same time, our findings can inform refactoring recommendation approaches [16] so that they can be more aligned to the needs of community shepherds.

Personalities matter. Last but not least, we discovered that the developer's personality can have an impact on the emergence and variability of certain community smells (this is the case of *Radio Silence*, for instance) and, therefore, having information on this matter may be useful to properly integrate a member in the community as well as to successfully evolve the entire community. This is, however, not easy to achieve. First, in most cases personality data cannot be automatically minable from open repositories, hence limiting their actual extraction. Second, there are evident privacy concerns that may even preclude the extraction/usage of personality data. For these reasons, a possible new research avenue is represented by the definition of methodologies able to estimate developer's personality through the extraction of data that do not interfere with the private sphere of developers. In addition, we also recommend community shepherds as well as automated tools to consider personality when splitting a community in sub-teams.

VII. Conclusion

In this paper, we sought to understand the variability of community smells, considering 60 software projects and statistically analyzing how 40 socio-technical metrics relate to the increase/decrease of four community smell types. Our findings revealed the factors that community shepherds should monitor to avoid the proliferation of specific community-related issues.

Our future research agenda reflects the main findings of our work. We aim at corroborating the results by considering different projects as well as systems developed in other settings (*e.g.,* industrial projects). Secondly, we envision further ethnographic studies with companies to verify on the ground the effect of socio-technical factors on community smells. Finally, we aim at focusing on the creation of usable tools and methods that can visualize and report to community shepherds the key data to enable appropriate governance mechanisms.

Acknowledgment

Gemma and Damian are partially supported by the European Commission grant no. 825040 (RADON). Fabio acknowledges the support of the Swiss National Science Foundation through the SNF Project No. PZ00P2_186090 (TED).

References

[1] P. Ralph, M. Chiasson, and H. Kelley, “Social theory for software engineering research.” in *EASE*, S. Beecham, B. A. Kitchenham, and S. G. MacDonell, Eds. ACM, 2016, pp. 44:1–44:11.

[2] M. De Stefano, F. Pecorelli, D. A. Tamburri, F. Palomba, and A. De Lucia, “Splicing community patterns and smells: A preliminary study,” in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 703–710.

[3] N. Bettenburg and A. E. Hassan, “Studying the impact of social structures on software quality,” in *2010 IEEE 18th International Conference on Program Comprehension*. IEEE, 2010, pp. 124–133.

[4] H. Sharp, H. Robinson, and M. Woodman, “Software engineering: Community and culture.” *IEEE Software*, vol. 17, pp. 40–47, 2000.

[5] H. Jaakkola, “Culture sensitive aspects in software engineering.” in *Conceptual Modelling and Its Theoretical Foundations*, ser. Lecture Notes in Computer Science, A. Dusterhoft, M. Klettke, and K.-D. Schewe, Eds., vol. 7260. Springer, 2012, pp. 291–315.

[6] G. J. Hofstede, C. M. Jonker, and T. Verwaart, “Modeling power distance in trade.” in *MABS*, ser. Lecture Notes in Computer Science, N. David and J. S. Sichman, Eds., vol. 5269. Springer, 2008, pp. 1–16.

[7] D. J. Greenhoe, “Properties of distance spaces with power triangle inequalities.” *PeerJ PrePrints*, vol. 4, p. e2055, 2016.

[8] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, “Social debt in software engineering: Insights from industry,” *Journal of Internet Services and Applications*, pp. 1–17, Oct. 2014.

[9] ——, “What is social debt in software engineering?” in *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, May 2013, pp. 93–96.

[10] F. Palomba, D. A. A. Tamburri, F. A. Fontana, R. Oliveto, A. Zaidman, and A. Serebrenik, “Beyond technical aspects: How do community smells influence the intensity of code smells?” *IEEE Transactions on Software Engineering*, 2018.

[11] A. Martini and J. Bosch, “Revealing social debt with the CAFFEA framework: An antidote to architectural debt,” in *2017 IEEE International Conference on Software Architecture Workshops*. IEEE Computer Society, 2017, pp. 179–181.

[12] A. Martini, T. Besker, and J. Bosch, “Technical debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations,” *Science of Computer Programming*, vol. 163, pp. 42–61, 2018.

[13] D. A. A. Tamburri, F. Palomba, and R. Kazman, “Exploring community smells in open-source: An automated approach,” *IEEE Transactions on Software Engineering*, 2019.

[14] G. Catolino, F. Palomba, D. A. Tamburri, A. Serebrenik, and F. Ferrucci, "Gender diversity and women in software teams: How do they affect community smells?" in *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Society*. IEEE Press, 2019, pp. 11–20.

[15] ——, "Gender diversity and community smells: insights from the trenches," *IEEE Software*, vol. 37, no. 1, pp. 10–16, 2019.

[16] ——, "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*. Association for Computing Machinery, 2020, p. 25–34.

[17] A. Mockus, "Organizational volatility and its effects on software defects," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 2010.

[18] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial open source projects: Insights from applying survival analysis," in *12th IEEE International Conference on Global Software Engineering, ICGSE 2017, Buenos Aires, Argentina, May 22-23, 2017*. IEEE Computer Society, 2017, pp. 66–75.

[19] B. Vasilescu, D. Posnett, B. Ray, M. G. J. van den Brand, A. Serebrenik, P. T. Devanbu, and V. Filkov, "Gender and tenure diversity in github teams," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*. ACM, 2015, pp. 3789–3798.

[20] A. Meneely, L. Williams, W. Snipes, and J. A. Osborne, "Predicting failures with developer networks and social network analysis." in *SIGSOFT FSE*. ACM, 2008, pp. 13–23.

[21] G. Catolino, F. Palomba, D. A. Tamburri, and A. Serebrenik, "Toward an understanding of the community smell variability: A statistical perspective - online appendix - https://tinyurl.com/y6rp7kul," 2021.

[22] G. Catolino, F. Palomba, and D. A. Tamburri, "The secret life of software communities: What we know and what we don't know." in *BENEVOL*, 2019.

[23] I. Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 307–324, May 2011.

[24] M. Joblin, W. Mauerer, S. Apel, J. Siegmund, and D. Riehle, "From developer networks to verified communities: A fine-grained approach," in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*. IEEE Press, 2015, pp. 563–573.

[25] D. A. Tamburri, F. Palomba, A. Serebrenik, and A. Zaidman, "Discovering community patterns in open-source: A systematic approach and its evaluation," *Empirical Software Engineering*, vol. 24, no. 3, pp. 1369–1417, 2019.

[26] N. Almarimi, A. Ouni, and M. W. Mkaouer, "Learning to detect community smells in open source software projects," *Knowledge-Based Systems*, vol. 204, p. 106201, 2020.

[27] N. Almarimi, A. Ouni, M. Chouchen, I. Saidani, and M. W. Mkaouer, "On the detection of community smells using genetic programming-based ensemble classifier chain," in *Proceedings of the 15th International Conference on Global Software Engineering*, 2020, pp. 43–54.

[28] F. Palomba and D. A. Tamburri, "Predicting the emergence of community smells using socio-technical metrics: a machine-learning approach," *Journal of Systems and Software*, p. to appear, 2020.

[29] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *EASE'14*, 2014.

[30] A. Meneely and L. Williams, "Socio-technical developer networks: should we trust our measurements?" *2011 33rd International Conference on Software Engineering (ICSE)*, pp. 281–290, 2011.

[31] I. Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success? a study of coordination in a software project," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 307–324, 2011.

[32] D. A. Tamburri, "Software architecture social debt: Managing the incommunicability factor," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, pp. 20–37, Feb 2019.

[33] G. Avelino, M. T. Valente, and A. C. Hora, "What is the truck factor of popular github applications? a first assessment." *PeerJ PrePrints*, vol. 5, p. e1233, 2017.

[34] F. Ricca, A. Marchetto, and M. Torchiano, "On the difficulty of computing the truck factor." in *PROFES*, ser. Lecture Notes in Business Information Processing, D. Caivano, M. Oivo, M. T. Baldassarre, and G. Visaggio, Eds., vol. 6759. Springer, 2011, pp. 337–351.

[35] C. Manteli, B. van den Hooff, H. van Vliet, and W. van Duinkerken, "Overcoming challenges in global software development: The role of brokers." in *RCIS*, M. Bajec, M. Collard, and R. Deneckere, Eds. IEEE, 2014, pp. 1–9.

[36] C. Manteli, B. van den Hooff, and H. van Vliet, "The effect of governance on global software development: An empirical research in transactive memory systems." *Information & Software Technology*, vol. 56, no. 10, pp. 1309–1321, 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/infsof/infsof56.html#ManteliHV14

[37] S. W. A. Dekker, "Deferring to expertise versus the prima donna syndrome: a manager's dilemma." *Cognitive Technoogy Workshop*, vol. 16, no. 4, pp. 541–548, 2014.

[38] R. Leifer and A. Delbecq, "Organizational/environmental interchange: A model of boundary spanning activity," *The Academy of Mgmt. Rev.*, vol. 3, no. 1, pp. 40–50, Jan. 1978.

[39] N. Levina and E. Vaast, "The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems." *MIS Quarterly*, vol. 29, no. 2, pp. 335–363, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/misq/misq29.html#LevinaV05

[40] W. Du and S. L. Pan, "Boundary spanning by design: Toward aligning boundary-spanning capacity and strategy in it outsourcing." *IEEE Trans. Engineering Management*, vol. 60, no. 1, pp. 59–76, 2013.

[41] D. Tamburri, F. Palomba, and R. Kazman, "Success and failure in software engineering: a followup systematic literature review," *ArXiv*, vol. abs/2006.12086, 2020.

[42] D. A. Tamburri, R. Kazman, and H. Fahimi, "The architect's role in community shepherding." *IEEE Software*, no. 6, pp. 70–79, 2016.

[43] F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, and A. De Lucia, "The scent of a smell: An extensive comparison between textual and structural smells," *IEEE Transactions on Software Engineering*, 2018.

[44] P. Tourani, B. Adams, and A. Serebrenik, "Code of conduct in open source projects," in *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2017, pp. 24–33.

[45] H. Theil, "A multinomial extension of the linear logit model," *International economic review*, vol. 10, no. 3, pp. 251–259, 1969.

[46] P. Allison, "When can you safely ignore multicollinearity," *Statistical horizons*, vol. 5, no. 1, pp. 1–2, 2012.

[47] D. N. McCloskey and S. T. Ziliak, "The standard error of regressions," *Journal of economic literature*, vol. 34, no. 1, pp. 97–114, 1996.

[48] P. V. Singh, Y. Tan, and V. Mookerjee, "Network effects: The influence of structural capital on open source project success," *Mis Quarterly*, pp. 813–829, 2011.

[49] S. A. Licorish, "Collaboration patterns of successful globally distributed agile software teams: the role of core developers," Ph.D. dissertation, Auckland University of Technology, 2013.

[50] J. Long, "Understanding the role of core developers in open source software development." *Journal of Information, Information Technology & Organizations*, vol. 1, 2006.

[51] J. West and S. O'mahony, "The role of participation architecture in growing sponsored open source communities," *Industry and innovation*, vol. 15, no. 2, pp. 145–168, 2008.

[52] M. Cataldo, J. D. Herbsleb, and K. M. Carley, "Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity," in *Second ACM-IEEE international symposium on Empirical software engineering and measurement*. New York, NY, USA: ACM, 2008.

[53] D. Homscheid, M. Schaarschmidt, and S. Staab, "Firm-sponsored developers in open source software projects: a social capital perspective," 2016.

[54] B. Bazelli, A. Hindle, and E. Stroulia, "On the personality traits of stackoverflow users," in *2013 IEEE international conference on software maintenance*. IEEE, 2013, pp. 460–463.

[55] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, "An experimental investigation of personality types impact on pair effectiveness in pair programming," *Empirical Software Engineering*, vol. 14, no. 2, p. 187, 2009.