Exploring Community Smells in Open-Source: An Automated Approach

Damian A. Tamburri, *Member, IEEE,* Fabio Palomba, *Member, IEEE,* Rick Kazman, *Member, IEEE*

Abstract—Software engineering is now more than ever a community effort. Its success often weighs on balancing distance, culture, global engineering practices and more. In this scenario many unforeseen socio-technical events may result into additional project cost or "social" debt, e.g., sudden, collective employee turnover. With industrial research we discovered *community smells*, that is, sub-optimal patterns across the organisational and social structure in a software development community that are precursors of such nasty socio-technical events. To understand the impact of community smells at large, in this paper we first introduce CODEFACE4SMELLS, an automated approach able to identify four community smell types that reflect socio-technical issues that have been shown to be detrimental both the software engineering and organisational research fields. Then, we perform a large-scale empirical study involving over 100 years worth of releases and communication structures data of 60 open-source communities: we evaluate (i) their diffuseness, i.e., how much are they distributed in open-source, (ii) how developers perceive them, to understand whether practitioners recognize their presence and their negative effects in practice, and (iii) how community smells relate to existing socio-technical factors, with the aim of assessing the inter-relations between them. The key findings of our study highlight that community smells are highly diffused in open-source and are perceived by developers as relevant problems for the evolution of software communities. Moreover, a number of state-of-the-art socio-technical indicators (e.g., socio-technical congruence) can be used to monitor how healthy a community is and possibly avoid the emergence of social debt.

Index Terms—Software Organisational Structures; Software Community Smells; Human Aspects in Software Engineering; Social Software Engineering; Empirical Software Engineering;

1 INTRODUCTION

Software is increasingly being engineered by large globally-distributed communities with highly complex social networks of software development. Knowing more about the quality of these communities and their social networks as well as the factors that affect their quality is critical to software success [1], [2], [3]. Several notations have been used in the software engineering literature to elicit and study these social networks, e.g., Developer Social Networks (DSNs) for bug prediction or error-pronenness [4], [5]. Moreover, several quality factors have been proposed over the years to highlight the importance of social aspects in software engineering. Nevertheless, both research and practice discuss software development communities and their characteristics rather vaguely; none has yet precisely quantified and evaluated the cost of the potential flaws in community structures and their (mis-)alignment to software structures [6]. Fewer still have identified and quantified a meaningful set of software community characteristics and associated these with project thresholds for achieving good quality [7], [8].

Our objective is in line with the emerging DevOps trend of speeding up software lifecycles from a technical and organisational perspective; we aim to offer means to continuously analyse a live organisational structure and make it more "healthy" by finding, tracking, and possibly removing negative or detrimental community behaviour across the software community. We begin by formalising, operationalising, and evaluating the effects of software development community "smells" [9], that is, patterns of sub-optimal organisational and socio-technical characteristics that may lead to tangible problems in development communities [10]. In fact, much like code smells in source code [11], [12], community smells are **not** "show-stoppers" for software code or system builds, rather, they reflect circumstances that, on the long run, manifest in additional project cost — a phenomenon called *social debt* [13]. It is our intention to further our understanding also in the conditions wherefore community smells are actually detrimental or whether some of them can be accepted as de-facto organisational procedures, especially in an open-source context.

To conduct our analysis, we adopt a state of the art socio-technical analysis tool called CODEFACE [14] and use this to evaluate community smells in action. We focus on four community smells previously seen in organisations, social networks, and software engineering research [9], [15], [16], [17], namely: (1) the *Organisational Silo* effect—reflecting isolated sub-communities; (2) the *Black Cloud* effect—reflecting excessive recurrent communication; (3) the *Lone Wolf* effect—reflecting isolated individuals acting as knowledge brokers; (4) the *Bottleneck* effect—an instance of the "unique boundary spanner" phenomenon [18] in software engineering.

Stemming from the above community smells, we study: (a) their diffuseness, i.e., how many instances of such community smells are present in open-source; (b) their perception by developer communities; (c) their relation to

- *D. A. Tamburri is with the Eindhoven University of Technology, The Netherlands*
  E-mail: d.a.tamburri@tue.nl
- *F. Palomba is with the University of Zurich, Switzerland*
  E-mail: palomba@ifi.uzh.ch
- *R. Kazman is with the University of Hawaii & SEI/CMU, USA*
  E-mail: kazman@hawaii.edu

socio-technical factors from the state of the art, e.g., socio-technical congruence [7], [8]. While the first two objects of study serve as starting point for our analysis, the last point, serves the purpose of identifying which factors from the state of the art relate to community smells (if any) and can therefore be used as monitored quantities that mediate smells' occurrence.

In terms of diffuseness, we found an average of 25 community smells per community, per release. The only exception is with the *Black Cloud* effect, which is less present and tends to appear more frequently in communities composed of more than 50 participants. Moreover, in terms of smells perception, our survey in "smelly communities" shows that the negative effects connected to smells are indeed perceived by the respective community members. This is also confirmed by an additional study where we directly interviewed 35 developers of 11 communities, asking them about their perceptions of the harmfulness of community smells. Finally, we found that several socio-technical factors, such as socio-technical congruence, are correlated with a lower number of community smells. For example, we confirmed that socio-technical congruence *does* reflect a lower number of smells and hence a higher quality of organisational structures and consequent lower social debt. Conversely, other factors such as distance do not correlated with the emergence of any smell.

Our study led us to three conclusions: (a) further investigating the causes and effects connected to community smells is paramount for efficient running of distributed organisations, particularly those interested in high release velocity; (b) known factors from the state of the art offer a good starting point to instrument the study of community smells; (c) automated means for smell detection and evaluation need to be created and empirically evaluated.

In summary, in the scope of this manuscript we provide 4 novel contributions:

1) The operationalisation and quantitative detection mechanisms for 4 community smells from previous research;
2) The implementation of said mechanisms as an extension of the previously available CODEFACE tool. This extension is also open-source and was made available as the CODEFACE4SMELLS tool under the same license;
3) An extensive empirical evaluation of the presence, perception, consequences, and naturalness of community smells as a phenomenon occurring in 60 open-source communities.

The impact of the above contributions and conclusions on both industry and academia are manifold. First, academics can use our results, and the tools we devised to obtain them, to further study community smells and their associated effects. Second, practitioners could use said tools and conclusions to "refactor" their own communities in line with more efficient organisations. Third, open-source practitioners can use the results we provide for better community management and steering.

Paper Structure. Section 2 introduces the basic terminology needed to comprehend our work. Section 3 provides background and definitions. Section 4 outlines our study of community smells, while Section 5 offers an overview of study results and their discussion. Section 5.4 analyses the trade-offs of using community smells to assess how healthy a community is. In Section 6 we discuss possible threats to validity and verifiability. Section 7 discusses related work. Finally, Section 8 concludes the paper.

2 THEORETICAL FRAMEWORK

Our study aims at detecting and understanding the role of *community smells* in the scope of *software development communities* as reflected in their *organisational structure*. These concepts are defined as follows.

According to Pugh [19] an *organisational structure* is a complex multi-layered network of relations across people and artefacts outlining how certain activities are directed in order to achieve the goals of an organization. For the sake of operationalisation, organisational structures are formally-defined graphs, or *sociograms* [20], featuring (1) organisational nodes (people, artefacts, etc.) (2) social relations (across people and with collective intent [21]) as well as technical relations (across people and artefacts and relating to a specific craft [22]).

A *software development community* is a specific type of the afore-mentioned organisational social network upon which certain properties hold constantly (e.g., informal communication across a project's electronic channels) [10]. From a socio-metrical and social-networks analysis perspective, two structures need to be distinguished: (a) *macrostructure*, often simply called 'structure', corresponding to the overall organization of the community, its properties, definitions, norms [23]; (b) *microstructure*, corresponding to the micro-interactions between a subset of nodes part of the macrostructure.

From a research perspective, we seek to study the micro and macro *organisational structure* of *software development communities* with the purpose of identifying, possibly predicting, averting, and mitigating any recurring anti-patterns.

More specifically, just like any community structure, the aforementioned software development communities can develop sub-optimal conditions [24], [25] which we previously introduced as *community smells* [9] on analogy with code smells. The analogy signifies that, on one hand, community smells do identify sub-optimal circumstances (e.g., the lack of communication across different modules of a software system) but, on the other hand, these conditions do not necessarily destroy the organisational behaviour across the community. Rather, they prove detrimental and cause additional project costs, such as recurrent delays in communication, imperfect knowledge sharing, etc. Finally, with the term *project*, we identify the goal or shared practice that the community maintains as its central endeavour. For example, the Apache Spark community holds the delivery of the Apache Spark product as its key *project*.

It should be noted that community smells reflect sub-optimal recurrent *microstructures* that can be detected over time across a complex *macrostructure* such that, following Coleman et al. [23], software practitioners can influence and fine-tune the macrosocial behavior across their software community by uncovering and mitigating the microfoundations of such behavior. It is for this reason that each mi-

crostructural behavior reflecting known organisational anti-patterns (e.g., *Organisational Silo* effects, see Sec. 3) needs explicit addressing.

From the above theoretical foundations, it follows that, to positively influence and mitigate the *macrostructural* behaviors amounting to sub-optimal organisational circumstances (e.g., bad communication practices across the organisation or lack of vision of its members) a more fine-grained and *microstructural* lens of analysis is required. On one hand, previous research has attempted to formulate holistic indicators for the former macrostructural behaviors (e.g., [7]) but, on the other hand, never before investigated the microstructural circumstances which together amount to those macro-phenomena. Ours is the first such attempt and harnesses the notion of community smells as instances of transversal micro-/macro-structural phenomena that (1) indicate the emergence of nasty macrostructural circumstances (2) have structural features that can be detected using graph theory and social-network analysis and (3) are the most known to occur (e.g., consider the works of Alter [26] or Levina et al. [27]) and have an established body of knowledge all across organisational and social-networks literature, with empirically-established negative consequences [28], [29].

3 Community Smells and Their Detection

Community smells represent sub-optimal organisational and socio-technical characteristics within a software community that might lead to additional costs due to communication problems, rage-quitting (i.e., cases where a community member leaves the project before its end because of socio-technical issues with other community members), and so on [10]. In the context of this work, we considered four of the community smells defined by Tamburri et al. [6], namely:

1) *Organisational Silo Effect:* siloed areas of the developer community that do not communicate, except through one or two of their respective members;
2) *Black Cloud Effect:* information overload due to lack of structured communications or cooperation governance;
3) *Lone Wolf Effect:* unsanctioned or defiant contributors who carry out their work with little consideration of their peers, their decisions and communication;
4) *Bottleneck or "Radio-silence" Effect:* an instance of the "unique boundary spanner" [18] problem from social-networks analysis: one member interposes herself into every formal interaction across two or more sub-communities with little or no flexibility to introduce other parallel channels.

In the following sections, we describe (1) the reasons behind the selection of these community smells, especially in relation to the theoretical framework presented in Section 2 and (2) how we operationalised their detection.

3.1 A Theoretical Tale of the Selected Community Smells

The rationale behind the selection of *Organisational Silo*, *Black Cloud*, *Lone Wolf*, and *Bottleneck* as community smells on which to focus in this paper is twofold. First, they reflect sub-optimal macro and microstructures of the organisational structure that, according to previous industrial experiences [30], are commonly observable in practice. Second, all the community smells considered have been shown to have a negative impact in practice, as they may not only lead to the emergence of social debt [6], but also to poor source code quality [31]. As such, our study targets those smells whose impact is higher. In the following, we further describe the relation of these smells to the theoretical framework as well as provide an overview of their impacts in practice.

3.1.1 Relation with the Theoretical Framework

The selected community smells are related to recurrent communication and collaboration hindrances that occur together (in the context of *Organisational Siloes* and *Lone Wolf* effects) or across a larger and communication graph (as with *Black Clouds* and *Bottlenecks*). Altogether, the smells link macrostructural phenomena (e.g., lack of vision, lack of or nasty communication) with microstructural patterns. From a theoretical perspective, community smells are patterns of organisational, social, and technical circumstances that span both the macro- and microstructures in the organisational structure. For example, an *Organisational Silo* connects an organisational dyad (i.e., a pair of coordinated developers) [32] working over a single file in part of the project coordination microstructure to a hindrance of communication, even of multiple scales (e.g., in the *Lone Wolf* effect); the link between the communication and coordination microstructures may be used to diagnose higher-order macrostructural phenomena in a targeted community. Similarly, a microstructural pattern (e.g., a clique inside a larger graph) can be linked to a higher-order phenomenon in the macrostructure; such is the case of the *Black Cloud* and *Bottleneck* effects where separated communication cliques occur over time and manifest themselves into higher-order, recurring, and nasty circumstances (e.g., overall production delays, lower quality, code churn [31]).

Overall, although the definition of such smells may seem similar, it is important to consider that they have different granularities and, as such, arise in a different ways and involve different socio-technical situations. As an example, the *Black Cloud* appears as a consequence of disagreeing repeated communications among community members that have the effect of obfuscating information, while the *Bottleneck* represents one single member that interposes herself into every formal interaction across two or more sub-communities with little or no flexibility to introduce alternative channels. Thus, while the former smell originates in the behavior of more community members and has the possible effect of obfuscating a wide amount of information, the second one refers to a single contributor that does not provide information on her own work. Thus, these phenomena represent different organisational/social situations that require specific attention and formulations which match their exact manifestations, emergence, and characteristics. Another example, as Leistner [33] pointed out, refers to the *Organisational Silo*, which is a smell that exists across several layers of the organisational structure, spanning both its macro- and micro-structure, namely the "local" (a communication/collaboration clique of the larger organizational network) and "global" graphs (i.e., the entire

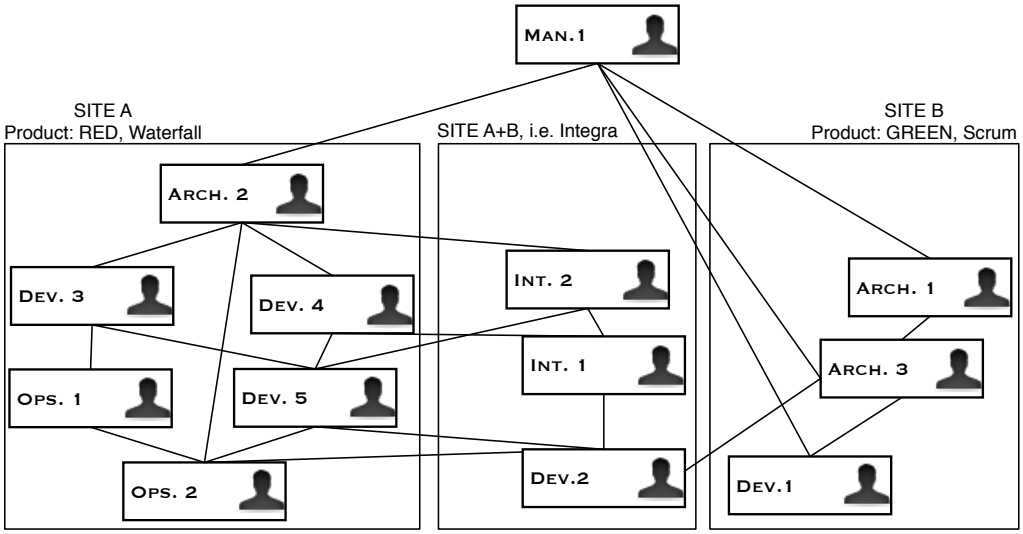


Fig. 1: The organisational scenario under investigation of the *Integra* project.

organizational network). Such a smell needs uncovering and explicit bridging [34] at all levels, but the organizational dynamic needs identification first. As a final note, it is worth mentioning that the selected community smells are those whose detection can be enabled using the data currently openly available in most open-source communities, i.e., communication and collaboration data extractable from issue trackers, mailing lists, and similar.

3.1.2 Impact of Community Smells

To intuitively grasp the notion of community smells and their impact in practice, one should consider that all community smells are *effects—Lone Wolf* effect, *Organisational Silo* effect, etc.—meaning that they produce a visible manifestation which is not bad per se, but, in the long run, may eventually manifest into a negative consequence. For example, consider the real-life industrial software organizational structure reported previously [6], [9] depicted in Fig. 1. The scenario features a software project (called *Integra* from now on) undergoing integration with two other software products, called RED and GREEN, respectively.

Nodes in the graph are people in the development network under study, while edges represent frequent recorded interactions between network members. The community of developers under analysis involves two geographically distributed production sites A (headquarters) and B (remote site) providing development as well as customer support and operations. A and B sites are both responsible for the implementation of incoming user requests (e.g., new requirements, revised requirements, bug-reports, etc.) but B also handles maintenance of both products being integrated.

A big organisational difference between sites A and B is that responsibilities in the remote site B are limited to what is decided by product managers in Site A. Product managers and architects in site A are responsible for management, software architecture, requirement elicitation and critical decision-making. In addition, while RED is a well-established product, active for well over 10 years, and managed following an established waterfall model, the GREEN product (and people) are relatively new and have adopted agile methods. The RED/GREEN integration team resides mainly in site A and acts as an intermediary between Sites A and B. The RED/GREEN team is meant to integrate decisions and their implementations on both products into one coherent whole, sometimes borrowing expertise from either RED or GREEN. Essentially, the scope of people working on RED is to maintain critical and un-replicable operations for the product while aiding its integration with GREEN. Conversely, the scope of people working on GREEN is to adapt some of its interfaces to interact with RED, while developing new functionality. Finally, the scope of people working on the RED/GREEN integration is to integrate RED and GREEN interfaces using middleware, wrappers, and similar integration technology.

According to our analysis, this scenario reflects the presence of at least 3 of the smells we focus on in this study. Specifically, practitioners at Site A (together with people at Sites A+B) constitute an *organizational silo* which is also blended within a *Black Cloud* effect, since they communicate with Site B only via a specific, restricted number of professionals (Dev 2, in this case). Similarly, Dev. 1 constitutes a *Lone Wolf*, who is focused on improving the microservices under her care, regardless of the integration efforts currently undergoing. From a practical point of view, all the considered community smells have been shown to have a negative impact for the evolution of software projects. This is, however, not only related to the emergence of social debt [6], which represents a natural consequence of having community smells. Indeed, in our previous work [31], we demonstrated that (i) software communities affected by these four community smells are more prone to the introduction of technical debt and (ii) different community smells lead to the emergence of different code smells in source code. On the basis of these results, we can claim that the definition of automatic mechanisms for the identification of those smells can provide important benefits for practitioners with respect to both management of the community and monitoring of the evolution of source code.

3.2 Operationalizing Community Smells

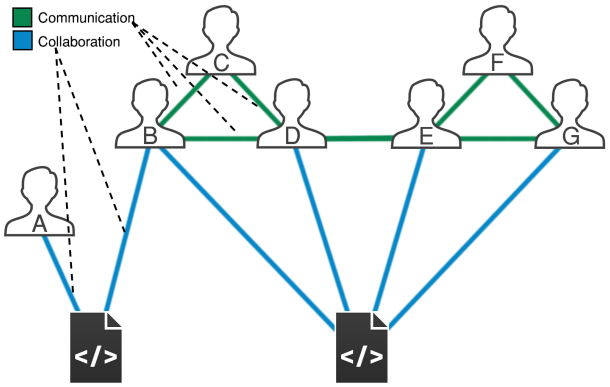


Fig. 2: High-level representation of the Developer Social Network (DSN) structure we used for community smell detection.

Starting from the developer networks built by CODEFACE, we detect instances of smells according to the formalisation below. For all of them, a premise is needed:

Premise. Let $G_m = (V_m, E_m)$ be the communication graph of a project and $G_c = (V_c, E_c)$ its the collaboration graph.

More precisely, for *communication* we mean the relation by which two or more developers communicate with each other through any channel: for example, a communication link between two developers is established in case they reply to the same discussion within a mailing list or they comment on the same issue in the issue tracker. As for *collaboration*, we mean the relation for which two or more developers have worked on the same source code elements. This is established by considering the change history of a project, looking for cases where two or more developers have modified the same code entities.

3.2.1 Organisational Silo Effect

The Silo effect reflects independent sub-communities—parts of a larger software development community—and these sub-communities often duplicate effort and waste resources due to their isolation [9], [35]. With the occurrence of Silo effects, social debt manifests as decaying communication across sub-communities and consequent negative effects on developers' situational awareness [36] as well as degradation of projects' socio-technical congruence [6], [31]. Also, according to recent findings [9], the Silo effect may lead to tunnel-vision, since participants may focus their cooperation and communication solely on other members of their narrow sub-community rather than on the broader community. Finally, community members belonging to an *Organisational Silo* may exhibit egotistical behaviour leading to unsanctioned architectural decisions [17] as well as defiance of the decisions of others [9].

Based on this definition, we define the set of *Organizational Silo* pairs S as follows:

$$\{(v_1, v_2) | v_1, v_2 \in V_c, (v_1, v_2) \in E_c, (v_1, v_2) \notin E_m^*\}$$

where E_m^* is the transitive closure of E_m. With *transitive closure* we indicate the transitive closure of a graph. More specifically, given a directed graph, the operation finds out if a vertex j is reachable from another vertex i for all vertex pairs (i, j) in the given graph. With *reachable* we mean that there is a path from vertex i to j. The reach-ability matrix is called *transitive closure* of a graph. For the sake of precision, we capture the Silo Effect at the finest grain possible, i.e., that of collaboration dyads: pairs of co-committing developers. An example is shown in Figure 3. Here the Silo effect is reflected on developer "1", who does not communicate with developer "2" even though "1" is collaborating with "2". Conversely, developer "2" is communicating with (at least) one other developer, "3", who belongs to a sub-community other than "1". Considering the example proposed in Figure 2, an operationalisation of the identification pattern for the Silo effect has two steps. In the first step, the identification mechanism compares the collaboration network (bottom half of Figure 2) with its communication counterpart (top half of Figure 2). Then it verifies that the developer identified by the letter A is present in the collaboration network, i.e., A commits to files co-committed by others, but is not present in the communication DSN reflecting those files.

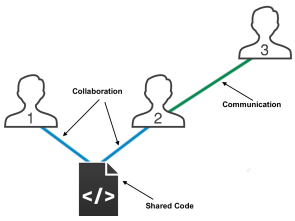


Fig. 3: *Organisational Silo* Effect Community Smell identification pattern.

3.2.2 Lone Wolf Effect

The *Lone Wolf* community smell reflects circumstances in which communication may indeed be present but insufficiently addressing project needs [9], [35]. The result is developer free-riding and unsanctioned architectural decisions that cause nasty ripple effects such as code duplication and churn [36]. Thus, we define the set of *Lone Wolf* pairs L as the set of collaborators that do not directly or indirectly communicate with each others. More formally:

$\{(v_1, v_2) | v_1, v_2 \in V_c, (v_1, v_2) \in E_c, (v_1, v_2) \notin E_m^*\}$. By definition $L \subseteq S$.

The identification pattern for the *Lone Wolf* smell is based on the detection of development collaborations between two community members that have intermittent communication counterparts or feature communication by means of an external "intruder", i.e., not involved in the collaboration. A simple example is given in Figure 4. In this example two developers, "1" and "2", are collaborating on some code, but they are not connected by any communication link other than developer "3", who is not co-committing on a shared file. In this case, either developer "1" or developer "2" (or both) can develop a *Lone Wolf* community smell.

This smell reflects the presence of possible side effects generated by the *Organisational Silo* such as communication decay or negative influence on developer awareness and heavy socio-technical congruence degradation. Our conjecture is that the occurrence of the *Organisational Silo* effect

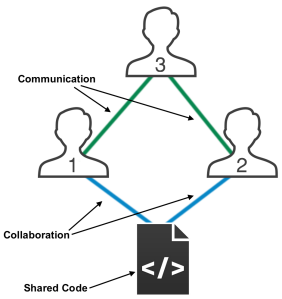


Fig. 4: Lone Wolf Community Smell identification pattern.

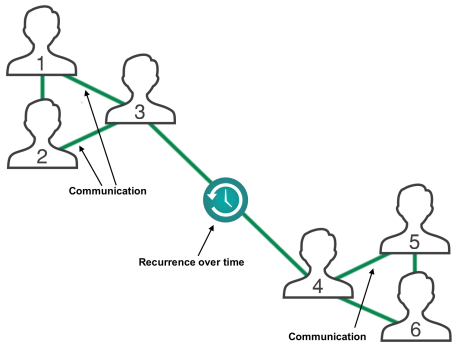


Fig. 5: Black-cloud Effect Community Smell, an identification pattern.

is not negative per se. But when that occurrence is compounded by the occurrence of *Lone Wolves*, extra attention must be paid to avoid negative consequences such as delays and unmanageable social debt. The *Lone Wolf* smell reflects dyads of co-committing (collaborating) software developers who exhibit uncooperative behaviour and mistrust by not appropriately communicating.

3.2.3 Black-cloud Effect

The Black Cloud Effect leads to negative social interactions within a software development community featuring: (a) community members' inability to cover knowledge and experience gaps between two different software products developed within the same software community [9]; (b) lack of periodic and official opportunities to share and exchange knowledge between all community members [9], [35]. Whenever these two circumstances occur together, they can generate a "black-cloud" of misinformation (e.g., confusing, delayed, or unnecessary communication that generates communication overload) within the community. The main consequence of the *Black Cloud* effect is to obfuscate project vision, compromising progress [9], [35]. The occurrence of this community smell can be generated or worsened by several socio-technical triggers:

- absence of ad-hoc protocols for knowledge sharing;
- lack of boundary spanners;
- presence of inefficient communication filtering protocols.

Moreover, the *Black Cloud* Effect smell is associated with several other side effects such as: lowering of trust between developers, information obfuscation, as well as inception of the *Organisational Silo* Effect, due to the rise of egoistic behaviour. The identification pattern for the *Black Cloud* Effect smell reflects sub-communities that in subsequent release windows do not communicate, with the exception of two community members (i.e., boundary spanners in social-network jargon [37]), one belonging to each sub-community. The detection of the *Black Cloud* instances starts with the identification of vertex clusters as already implemented in CODEFACE. More specifically, let $P = \{p_1, \ldots, p_k\}$ be a mutually exclusive and completely exhaustive partition of V_m induced by the clustering algorithm. From the partition, *Black Cloud* is the set of pairs of developers C that connect otherwise isolated sub-communities, more formally:

$$\{(v_1, v_2) | v_1, v_2 \in V_m, (v_1, v_2) \in E_m, \forall i, j (((v_1 \in p_i \wedge v_2 \in p_j) \Rightarrow i \neq j) \wedge \forall v_x, v_y ((v_x \in p_i \wedge v_y \in p_j \wedge (v_x, v_y) \in E_m) \Rightarrow v_x = v_1 \wedge v_y = v_2))\}$$

The smell manifests if the above condition holds for at least two consecutive organisational time-windows (fixed to 3-month intervals, in the case of CODEFACE4SMELLS). An example is presented in Figure 5. Here the occurrence of *Black Clouds* reflects two developers, "3" and "4", who are the lone boundary spanners across two different sub-communities and over time—at least two subsequent analysis windows (3 months, in our case).

Detecting black clouds requires eliciting the communication network and applying known community detection algorithms [38] to identify sub-community structures and boundary spanners across them. For example see Figure 2 where two sub-communities (previously specified) can be detected by considering the density of communication links.

3.2.4 Bottleneck Effect

The *Bottleneck* community smell is characterised by the occurrence of the following sub-optimal characteristics within a software development community: (a) proposed changes within every software development phase require an extraordinary quantity of time to be implemented [9], [6]; (b) time waste [9], [35]; (c) hidden or counterintuitive information (and broker) locations [9]; (d) highly formal or complex organisational structure [9]; (e) highly regularized procedures [9], [35].

The fundamental side-effect generated by this community smell is a massive delay that characterises key organisational processes within the community such as decision-making, due to personnel unavailability or communication overload. The identification pattern of this smell is based on the detection of unique knowledge and information brokers in different sub-communities.

In our attempt to define an automatic identification pattern for this community smell we focused on the analysis of project communication networks. We considered the six key factors around *Bottleneck* as reflecting the presence, within a project organisational structure, of a unique boundary spanner across several different sub-communities (i.e., more than 2). The social-network analysis concept of unique boundary spanner [37] has, in fact, a remarkable similarity to *Bottleneck*. A unique boundary spanner interposes him/herself into every formal interaction across two or more sub-communities and if the organisational structure of the project is complex and characterised by highly formal procedures, it will not be possible to incept parallel

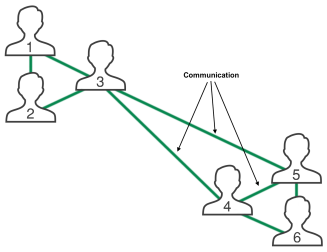


Fig. 6: Bottleneck Community Smell.

information channels between other members of the sub-communities. From a formal perspective, we define the set of *Bottlenecks* as:

$$\{v|v \in V_m, \exists i(v \in p_i \wedge \forall v_x(v_x \in p_i \Rightarrow v = v_x))\} \cup \{v|v \in V_m, \exists v_x, i, j(v \in p_i \wedge v_x \in p_j \wedge (v, v_x) \in E_m \wedge \forall v_y, v_z((v_y \in p_i \wedge v_z \in p_j \wedge (v_y, v_z) \in E_m) \Rightarrow v_y = v)\}$$

To further elaborate on the definition of this community smell, let consider the example proposed in Figure 6. As shown, detecting *Bottleneck* requires the identification of community members who are the *only* members of their sub-community that communicate with (at least) two other sub-communities. Therefore, assuming a communication link was present between developer "A" and "B", then developer "B" is the pivot of *Bottleneck*.

3.3 CodeFace4Smells Extension

From the perspective of the CODEFACE tool, a *community* is operationalised as a densely connected set of nodes within the community group (i.e., the members that make up a development community) which is sparsely connected to all other nodes in the network. To identify and properly characterise the community structure, the CODEFACE tool enacts two community detection strategies, defined as follows, paraphrasing from Joblin et al. [14]:

1) *Function* — To recover a community structure, CODEFACE uses a heuristic for identifying when two developers are engaged in a coordinated effort using a fine-grained heuristic based on code structure, where developers are considered to be coordinated when they actually contribute code to a common function block. Furthermore, CODEFACE uses the commits' timestamp for identifying the appropriate directions of the edges in the recovered community structure.
2) *Committer-Author* — In this method, the tool uses tags to identify relationships between all people that contributed to a common commit, including authors, reviewers, and testers. For example, sign-off tags are self-reported acknowledgments of participation on a commit, therefore the tag-based networks undoubtedly capture real-world collaboration [14].
3) *Community-Verification* — to verify the recovered community structure, CODEFACE uses a random null-model to compute the probability of observing the identified community in an equivalent class of null-model graphs that lack a community structure. The tool generates the null-model using a standard approach called the configuration model for random graphs, where nodes are joined uniformly at random under the constraint that the degree distribution is identical to the observed graph [39].

To the above heuristics, we add a systematic implementation of the operationalisation provided in the previous section, to allow for automated detection of community smells at the same time as CODEFACE operates community structure recovery. The output of the tool is represented by a `CSV` file containing the community smell instances identified over a social structure representation known as a Developer Social Network (DSN), a notation previously used for bug prediction and error-proneness [4], [5]. It is important to note that the detection tool has been previously evaluated [40]: in that context, we formally proved that it is able to correctly identify all the community smell instances affecting software communities using a formal interpretation of the DSN abstraction and the formalisations reported in this section. Thus, we claim that CODEFACE4SMELLS is an *accurate* tool on which to base the current study. Furthermore, the implementation and operationalisations currently available within CODEFACE4SMELLS inherit and extend the state of the art in social and organisational networks analysis. For example, we re-use and extend the concept of Simmelian ties [41] from Borgatti et al. and Krackhardt et al. [42] and use it to represent an organizational silo wherefore the ties in question (that is, triads of reciprocal and strong inter-personal relations in a social network of practitioners) among sub-groups across a development community constitute a stable set. Furthermore, we build upon the concept of similarity by Hinds et al. [43] to identify practitioners with overlapping extraction and identify *Lone Wolves*. Finally, we build upon solid foundations for the use of social networks data with regression modelling as shown by Krackhardt [44].

4 STUDY DESIGN

The *goal* of our study was to evaluate three aspects related to community smells: (i) their diffuseness in open source communities, (ii) the awareness of developers with respect to the symptoms indicating the presence of community smells, and (iii) the actual impact of community smells on open source communities. The *purpose* of the study was to improve the diagnosability of software community health. Specifically, we aim to answer the following research questions:

- **RQ1.** *How does the distribution of community smells in open-source software communities vary over time?*
- **RQ2.** *Do developers actually perceive the presence of smells in their community?*
- **RQ3.** *How do community smells relate to existing socio-technical factors?*

For the first research question, we studied the extent to which open-source communities suffer community smells and how the numbers of these smells vary over time. This is important to know so that we can give guidelines to architects and project managers regarding how to interpret trends and establish thresholds for "healthy" values. **RQ2** is aimed at understanding whether developers are conscious of the presence of such smells in their communities. If they

Community	# survey part.	# interviews part.
LibreOffice	12	4
Firefox	33	6
FFmpeg	14	2
VLC	14	3
Tomcat	9	5
Scala	18	1
Qt	22	3
Salt	13	5
Vagrant	21	2
Eclipse	5	3
Python	11	2

TABLE 1: Number of participants to survey and semi-structured interviews, respectively. Data is reported per community.

are not then it is even more important to detect such smells automatically. Finally, **RQ3** had the goal of assessing the relationship between community smells and other known socio-technical factors studied in the past. Smells are symptoms. For this research to affect the practice of software development we need to understand the root causes of smells so that we can provide advice on what project characteristics (measured by factors such as socio-technical congruence) to adjust.

4.1 Context of the study

The *context* of the study consisted of 60 active open-source software communities using GITHUB[1] as a tool for managing software versions. The selection of these systems was driven by two factors. On the one hand, to properly observe the phenomenon of community smells, we focused on communities having at least X contributors and Y commits performed in their history. On the other hand, we aimed at studying communities that are currently active, since this allows us to mitigate threats due to outdated phenomena. Thus, starting from the list of open source projects available on GITHUB, we randomly selected 60 systems having enough commits and contributors, verifying that at least ten commits were performed during the last month on each repository (our threshold for activeness). The complete list of communities studied, their repositories, and mailing list are reported in our online appendix [45].

Projects were selected according to the following criteria: (a) codebase size - 20 medium-sized (200-500 KLOC), 20 large (500-850 KLOC) and 20 very large ($>$ 850 KLOC); (b) main programming language - Java, C#, C, Python, YAML and other languages are included in our sample; (c) community size - our size distribution is evenly split among three ranges: medium ($<$50 members), large (50$>$150 members) and very-large ($>$150 members); (d) age - our age distribution is evenly split among three ranges: young projects ($<$24 months), established projects (24$>$32 months), and mature projects ($>$32 months).

1. http://github.com/

4.2 RQ1. The distribution of community smells

To answer **RQ1**, we ran CODEFACE4SMELLS over the set of software communities listed in Table **??**. It is important to note that the communication graph used to detect community smells was built considering the three months *before* the date where the analyzed project versions were released; this choice was guided by previous research [40] that showed how this time window allows us to correctly analyze the current organisational aspects of a software community, excluding outdated information.

Once we gathered the information about community smells, we verified their distribution in each time period for the subject projects. Furthermore, we verified whether there exists a correlation between system characteristics (#contributors, #commits, and KLOC) and the number of observed community smells. To properly verify this correlation, we repeated the analysis for each time period considered (e.g., we computed the correlation between the presence of community smells in a time period $T_{i \to j}$ and contributors present in the same period). To measure the relations, we exploited the Spearman rank correlation index [46] which measures the strength and direction of association between two ranked variables, and ranges between -1 and 1. A value of 1 represents a perfect positive linear relationship, -1 represents a perfect negative linear relationship, and values in between indicate the degree of linear dependence between the considered distributions. To interpret the results, we followed the guidelines provided by Cohen [47]: it is assumed that there is no correlation when $0 \leq \rho < 0.1$, small correlation when $0.1 \leq \rho < 0.3$, medium correlation when $0.3 \leq \rho < 0.5$, and strong correlation when $0.5 \leq \rho \leq 1$. Similar intervals also apply for negative correlations.

4.3 RQ2. The perception of community smells

To answer **RQ2**, we designed a survey involving the developers of the projects considered in the context of **RQ1**.

Recruitment. We adopted an *opt-in* strategy [48] when asking developers to participate in our study. We extracted from GITHUB the e-mail addresses of all the developers who have committed at least 10 changes during the year before the release dates taken into account: in this way, we focused only on developers having adequate experience with the projects and communities considered [49]. Then we sent them a first e-mail asking whether they would like to participate in our survey. In other words, we recruited only *volunteers* to avoid privacy issues or other developer concerns[2]. To mitigate the side effects of this generalisation, developers participation was stimulated offering by a prize of four Amazon gift cards with a total value of $100. The four prizes were awarded to four randomly chosen participants a few weeks after the survey ended.

As a result, we obtained a positive reply from 172 of the 5,169 developers approached (≈3.32%), who were later contacted with the actual survey. Table 1 reports the distribution of the participants across the investigated communities. We obtained answers from developers of 11 different projects in the dataset: thus, our survey covered 18% of all considered communities. We are aware that the opinions collected on

2. http://tinyurl.com/yd2usw5p

Question	Answer
Developer's background	
1. Country of Birth	Open answer
2. Year of Birth	Open answer
3. Current Occupation	Student Part-time Employee Full-time Employee Unemployed Retired
4. Community for which you contribute the most	Open answer
5. Role within the community where you contribute the most	Developer Maintainer Software Engineer Translator Graphic Other
6. Do you contribute to this project as an individual or because your company is involved in it?	Paid by a company Partially paid by a company Voluntary Developer
Community Smell Perception	
7. [BC] Delay of communications can stall some community activities.	strongly disagree|disagree|uncertain|agree|strongly agree
8. [BC] Lack of periodic opportunities to share and exchange knowledge between all community members is a threat to the success of the project.	strongly disagree|disagree|uncertain|agree|strongly agree
9. [LW] Community members having intermittent communications are a threat for the implementation of the project.	strongly disagree|disagree|uncertain|agree|strongly agree
9. [LW] Community members working on the same code and communicating by means of a third person cannot properly proceed the collaboration.	strongly disagree|disagree|uncertain|agree|strongly agree
11. [RS] High degree of formality is detrimental for project communications.	strongly disagree|disagree|uncertain|agree|strongly agree
12. [RS] There are highly regular procedures for communicating changes.	strongly disagree|disagree|uncertain|agree|strongly agree
13. [OS] There are different subgroups that rarely communicate with each other.	strongly disagree|disagree|uncertain|agree|strongly agree
14. [OS] Different subgroups are sometimes antagonists.	strongly disagree|disagree|uncertain|agree|strongly agree

TABLE 2: Questionnaire filled in by the study participants. [LW] indicates statements related to Lone Wolf; [RS] to Bottleneck; [BC] to Black Cloud; [OS] to Organisational Silo.

this sample might be not necessarily generalizable, however we argue that our analysis still provides confirmation with respect to the perception of community smells present in the community around the participants.

Survey Dissemination. The questionnaire was created and distributed to participants using GOOGLE FORMS[3]. We opened it two times with the aim of collecting as many replies as possible: the survey was first available from February 15 to March 5, 2016; then, it was open from May 7 to July 26, 2018[4]. The link to the questionnaire and a short introduction were sent to every recruited developer via e-mail. We estimated a completion time of 25 minutes.

Survey Design. The complete list of questions in our survey is presented in Table 2. It contained three parts. First, we gathered information about occupation, reference community, and role of the developers involved (Q1-Q6). Second, we asked participants to rate the validity of 8 statements using a 5 point Likert scale [50] ranging between *"Strongly Disagree"* to *"Strongly Agree"* (Q7-Q14). Such statements reported typical situations in which community smells occur. By collecting developers' answers to these questions we could match their perception of community smells with the community smells actually detected in their communities. It is important to point out that each statement referred to only one type of community smell. For instance, the statement *"There are different subgroups that rarely communicate with each other"* (Q13) was aimed at understanding whether developers actually recognise the presence of an *"Organisational Silo Effect"* in their community. In Table 2 we map each statement with the community smell we were interested in. It is important to note that, to avoid any possible bias in the responses, we took two precautions. First, we never mentioned the terms social debt or community smells in the questionnaire. Second, we collected participants' opinions on the typical situations leading to the emergence of community smells rather that directly ask them to rate how healthy was their own community. This was a conscious design decision taken on the basis of well-established findings in psychology research: as pointed out by Fisher [51], people might be reluctant to express negative or controversial opinions on topics they are too much involved in, and for this reason *indirect questioning* [52] should be preferred to increase the validity of the collected observations. In our context, this means that the surveyed developers might have been reluctant to reveal the presence of problems in the community they belong to, leading to provide invalid answers and bias our results. For this reason, we proceeded with the selected research approach.

3. https://www.google.us/intl/en/forms/about/

4. In the second round, we (re-)invited only those developers who had not already participated in the questionnaire in the first round.

Data Analysis. Once we collected the developers' questionnaires, we answered **RQ2** by means of statistical analyses. We computed the percentage of times developers perceived a community smell. We considered a smell as *perceived* in case a developer marked a statement indicating the symptoms of a smell with an *"Agree"* or a *"Strongly Agree"*.

Confirmatory semi-structured interviews. While a survey provides useful insights to understand the developers' perception of community smells, it might not be sufficient to provide definitive conclusions, as developers did not directly comment on such smells. For this reason, we performed a post-survey additional confirmatory study aimed at corroborating the findings of the survey study. We invited the participants who performed the survey to conduct a further semi-structured interview, to discuss with us the results obtained on their perceptions of community smells. We were able to perform 35 semi-structured interviews with developers coming from all 11 communities. Table 1 reports the number of interviewees per community. The interviews were conducted via Skype, and required ≈30 minutes per interview. We started the discussion by summarising the results obtained from the surveys—reporting both the overall findings and those related to the project the developer was involved to—and presenting the definitions of the community smells analysed in the study. This step was needed to provide developers with a complete overview of our research and to prepare them for the discussion. Afterwards, for each community smell, we focused on the three main questions:

1) *Did you happen to observe a [community smell] in your organisation?*
2) *How frequent is [community smell] in your organisation?*
3) *Do you think that [community smell] can be harmful for any socio-technical aspect of your organisation? If so, which ones?*

All interviews were recorded and transcribed for analysis. Then, two of the authors of this paper manually went through the developers' answers to extract relevant information and observations that confirmed/rejected the findings obtained from the surveys. In a first step, the two authors independently analyzed the developers' discussion and came up with a summary for each of the considered community smells. In a second step, they opened a discussion on the basis of their summaries with the aim of reaching an agreement on the insights given by the participants. Such a two-step process was needed to avoid subjective interpretations of the developers' answers.

4.4 RQ3. The relationship between community smells and existing socio-technical factors

In the context of this research question, we focused on identifying which socio-technical development aspects are related to or responsible for the presence of community smells. Such an analysis has a vital importance since it can reveal what are the factors that can be used as monitored quantities that mediate the occurrence of community smells. It is important to note that since social debt is considered a ubiquitous phenomenon within the software development lifecycle [9], it is not possible to clearly identify specific socio-technical aspects or phases responsible for increases or decreases of social debt. Thus, our analysis aims to shed light on the phenomenon as a whole: specifically, we defined a *Socio-technical Quality Framework* [53] that includes 40 socio-technical quality factors, which were extracted by analysing the existing literature.

Specifically, we conducted a systematic literature review (SLR) that investigated previous work concerning social factors for software engineering (e.g., [54], [13], [55], [56]), identifying the metrics proposed in related work [53]. The quality factors that emerged from our study are recapped in Table 3. To better scope the work of the present study, we selected a subset of the afore-mentioned 40 factors (see Table 3) by first isolating three independent lists of factors identified by the authors of this paper, who took the role of *inspectors* and extracted a set of initial metrics each. Subsequently, the inspectors refined the three sets with the goal of finding a single set containing all the social metrics: this has been done by merging the factors identified by all the inspectors and discussing whether to include or not the initial metrics discovered by one or two of them. To measure the level of agreement among the inspectors we computed the Krippendorff's alpha Kr_α [57]. Agreement measured 0.87, considerably higher than the 0.80 standard reference score [58] for Kr_α. A complete report of the papers analysed as well as of the coding activities performed is available in our on-line appendix [45].

5 Study Results

In this section we describe and analyse the results achieved when answering our research questions.

5.1 The Distribution of Community Smells

Figure 7 depicts the box plots reporting the distribution of the community smells analysed over the 60 projects in our dataset. For sake of readability, we removed outliers from the plots; furthermore, the plots report the overall distributions, i.e., the ones resulting by considering all the community smells arising over the different 3-month periods. To have a fine-grained view of the distribution of community smells over time, Table 4 reports minimum, mean, maximum, and standard deviation observed in each of the considered 3-month periods. Finally, Table 5 reports instead the number of projects in which we found at least one instance of such smells. To more clearly understand their distribution, in the table we divided the subject communities into three categories—*small*, *medium*, and *large*—based on the number of contributors.

As shown in Figure 7, three community smells—*Organisational Silo*, *Lone Wolf*, and *Bottleneck*—appear consistently: the median number for *Bottleneck* is 13, for *Lone Wolf* is 12 and for *Organisational Silo* is 9. Surprisingly, the median number of *Black Cloud* instances is 1. This result seems to contradict what has been previously observed in industrial communities [6], where practitioners frequently pointed out the existence of this smell. From a practical perspective, the absence of such a smell highlights that open-source

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

TABLE 3: Socio-technical Quality Factors Investigated in Total, reported from literature [53].

Community smell	Min.	Mean	Max.	St. Dev.
Black Cloud	0	0.18	7	0.65
Organisational Silo	0	13.65	91	19
Lone Wolf	0	25.62	151	36.17
Bottleneck	0	21.52	96	23.59

TABLE 4: Statistics of the number of community smells occurring in each of the considered 3-months time windows.

communities **do not** lack structured communication or governance, despite their contributors being largely part-time volunteers who are often involved in other projects. In other words, FLOSS developers try to organise communications in a structured manner so that they can avoid information overload. At the same time, the high numbers of the other community smells tell us a different story: although communications among developers are structured, there are often members who try to interpose themselves into every formal interaction across sub-communities (creating a *Bottleneck*) or that perform their tasks independently from community decisions (thus being a *Lone Wolf*). Furthermore, our findings highlight that, in several cases, there exist sub-communities that essentially do not communicate, each creating *Organisational Silos*. When considering the distribution over time (Table 4), we could provide similar observations. Indeed, *Organisational Silo*, *Lone Wolf*, and *Bottleneck* are the smells that appear more frequently in each time window, while the mean of *Black Cloud* instances is close to 0 and has a standard deviation of 0.65. Interestingly, there seems to be a pretty high standard deviation for all the three most diffused community smells (e.g., *Organisational Silo* has a standard deviation of 19): this indicates that there are time windows in which the number of community smells tend to decrease significantly with respect to other periods. Further analyses aimed at understanding the root causes behind this phenomenon as well as how community smell instances evolve over time are part of our future research agenda.

Looking at Table 5 we observe that the three most frequent community smells are present in almost all of the projects analysed. **All** the communities experienced at least one *Bottleneck* during their evolution, while 59 of them had at least one *Organisational Silo* and *Lone Wolf*. When correlating the number of community smells with project characteristics (Table 6), we observed a weak correlation between the number of contributors within the community and *Organisational Silo*, *Lone Wolf*, and *Bottleneck*. This confirms that the amount of instances of these smells is not solely dependent on the size of the community.

Community Dimension	Black Cloud	Organisational Silo	Lone Wolf	Bottleneck
< 50	8	19	19	20
$50 - 150$	11	20	20	20
> 150	12	20	20	20
Overall	**31**	**59**	**59**	**60**

TABLE 5: Number of projects/communities affected by community smells.

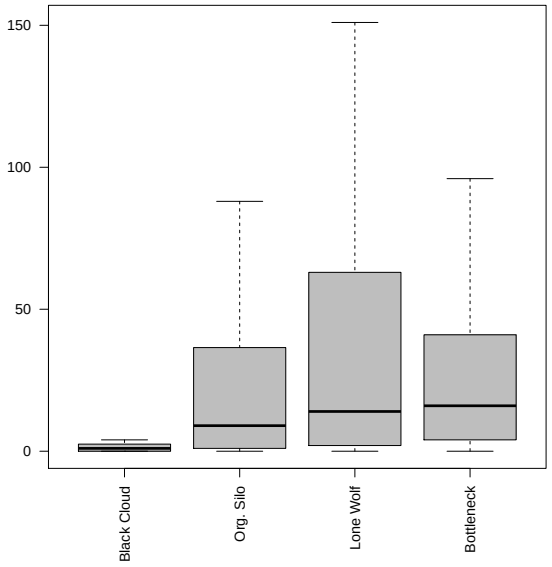


Fig. 7: Distribution of community smells on the 60 projects considered in our study.

	# Contributors	# Commits	KLOC
Black Cloud	**0.77***	0.23	0.11
Organisational Silo	0.48*	0.32	0.39*
Lone Wolf	0.47	0.36*	0.32
Bottleneck	0.42	0.44*	0.26

TABLE 6: Correlations between Project Characteristics and Community Smells. Statistically significant correlations ($\rho - value < 0.05$) are marked with '*'.

As for the *Black Cloud* smell, the discussion is different. Only eight projects with fewer than 50 contributors suffered this smell while the number of smelly communities slightly increases as the number of contributors increases: this may suggest that the number of *Black Cloud* instances is somehow dependent on the size of the community. The Spearman's value between *Black Cloud* and number of contributors seems to confirm that this smell is dependent on the size of the community, however further verification would be needed.

At the same time, we discovered that the simplistic technical aspects we controlled for, e.g., the number of commits or the total number of lines of code in a project, do not strongly influence the diffuseness of community smells. This result does not mean that we can exclude a relation between social and technical aspects in general. Our data suggests that a more structured research approach—digging deeper in potential confounding factors—is in order.

Summary for RQ1. In general, community smells are highly diffused in open source communities. Among the studied ones, *Organisational Silo*, *Lone Wolf*, and *Bottleneck* are found regularly, while the presence of *Black Cloud* instances may depend on the number of contributors within a community. We have not found a simple correlations between such smells and "technical" project characteristics.

5.2 The Perception of Community Smells

As explained in Section 4.3, we obtained 172 survey responses from 11 open source projects. Before discussing the perceptions of community smells, we report on the demographic and FLOSS-related data provided by our study participants.

5.2.1 Demographics

Figure 8 illustrates age, occupation, and role of the survey participants, along with the information on the number of developers falling into each of the reported categories. In the first place, we can observe that the participants were almost uniformly distributed across the different age ranges. The majority of them were between 26 and 35 years old (49), while developers younger than 25 years old and between 36 and 50 years old were 42 and 41, respectively. Finally, 40 participants were over 50 years old.

Looking at the working status of the respondents, we observe that 104 of them have a full-time job, while a minority had a part-time job (45). 22 participants identified themselves as a student and 1 was neither employed nor a student. This indicates that our data was collected primarily from people constantly working in open-source projects.

The respondents were mainly developers (89). This was expected, given the modalities through which we retrieved developer information. However, an important fraction of the respondents specified their status as software engineer (67), while a few stated that they were maintainers or translators (5 and 2, respectively). Finally, 9 were not able to identify themselves in any of these categories.

Finally, we asked respondents to specify if their contribution to the project is on voluntary or if their involvement within the project community is sponsored by a commercial company. We found that 144 of them participate in FLOSS development without any monetary interest; considering the 11 reference communities independently, the number of sponsored developers varied from a minimum of 3 (Tomcat) to a maximum of 27 (Firefox). The percentage of respondents

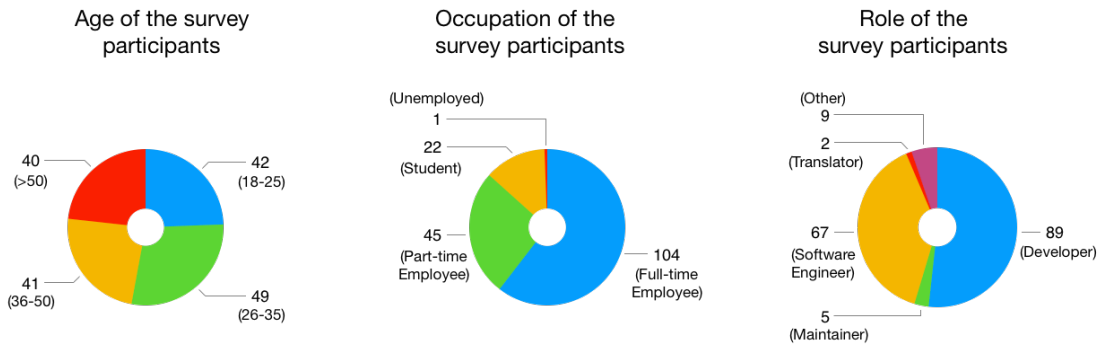


Fig. 8: Demographical information of the survey participants.

whose involvement was completely supported by a company was 17. Furthermore, 11 participants said that they were paid only partially by a commercial company. All in all, we conclude that the vast majority of our respondents work in open-source development as a full-time job. This made our population appropriate for the aspects that we wanted to investigate, i.e., community smells in open-source projects.

Besides the questions posed to participants, we also took into account the role of experience. Indeed, it is likely that this factor impacts the way developers perceive community smells. To account for this aspect, we mined the repositories of their 11 projects with the aim of computing two well-established metrics [59] capturing the experience of the contributors who participated in the survey: (i) commit tenure, which measures developer experience as the number of months since the developer's first event on any Github repository; (ii) project tenure, which measures developer experience on the project of interest as the number of months since their first event on the project repository. Once we extracted those metrics for all survey participants we correlated them—using the Pearson correlation coefficient—with the Likert scale values assigned by the participants to each of the survey statements; we mapped nominal values to numeric ones, e.g., *"Strongly Disagree"* was assigned to 1 while *"Strongly Agree"* to 5. As a result, we could examine the relationship between commit tenure and the answers provided in the survey. Indeed, the correlations between this experience metric and the values assigned to all the statements are higher than 0.6, meaning that there is a strong correlation between the two distributions. At the same time, we did not find any strong correlations when considering the project tenure metric. In other words, the overall level of experience in open source projects, but not the experience on a specific project, strongly correlates with the perception of community smells.

5.2.2 Community Smells Perception

Table 7 reports the results obtained when analysing the developers' opinions on the symptoms behind community smells. For each statement, we report the number of developers who rated its validity from *"Strongly Disagree"* to *"Strongly Agree"*. Our survey of open-source practitioners strongly confirmed their perception of the nasty effects associated with community smells: the vast majority of developers reported *all* the situations behind such smells as serious threats to the health of the community and the quality of the communication among developers. This was also confirmed by the participants of the confirmatory semi-structured interviews: all of them reported to have observed at least one community smell in their organisation and, at the same time, that the frequency of appearance is generally high and that they may cause socio-technical problems. It is important to note that we did not notice discrepant answers, meaning that all the interviewees agreed on the harmfulness of community smells. Thus, the first key result of **RQ2** is represented by the fact that developers perceive community smells as actually pervasive and harmful for the health of software communities. In the following subsections, we describe the results analyzing each community smell independently.

Black Cloud. The presence of this smell strongly impacts the way developers communicate with each other and share knowledge about the status of the project. As a consequence, communication delays might cause problems such as a complete stall of the community. 101 and 44 respondents rated statement 7 as *"Strongly Agree"* and *"Agree"*, respectively, thus confirming that effective communication among community members is a must to carry out the project. Interestingly, our respondents did not confirm that having periodic meetings with all community members is an important aspect to improve project activities. 117 participants were *"Uncertain"* with respect to statement 8. To further investigate the answers related to the presence of a *Black Cloud* smell we discussed these findings during semi-structured interviews. All 35 interviewees reported that *Black Cloud* instances occurred in their communities, even though their frequency depend on the size and maturity of the community. Moreover, they also reported that constant knowledge sharing is vital for the survival of the community. However, it is not relevant for them to have meetings with all community members: they just need to effectively communicate with other developers involved in the development of shared portions of code. For instance, one Tomcat developer reported:

"It happened to us in the past, and in general is pretty frequent to see people that do not want to discuss what they do or simply delay in communicating what they have implemented. Code review can somehow mitigate the problem, but still, programmers sometimes do not properly or timely communicate, causing other

Question	strongly disagree | disagree | uncertain | agree | strongly agree
Community Smell Perception	
7. [BC] Delay of communications can stall some community activities.	5 | 8 | 14 | 44 | 101
8. [BC] Lack of periodic opportunities to share and exchange knowledge between all community members is a threat to the success of the project.	5 | 5 | 117 | 32 | 13
9. [LW] Community members having intermittent communications are a threat for the implementation of the project.	6 | 17 | 25 | 36 | 88
10. [LW] Community members working on the same code and communicating by means of a third person cannot properly proceed the collaboration.	0 | 0 | 5 | 22 | 145
11. [RS] High degree of formality is detrimental for project communications.	4 | 8 | 11 | 46 | 103
12. [RS] There are highly regular procedures for communicating changes.	0 | 94 | 55 | 21 | 2
13. [OS] There are different subgroups that rarely communicate with each other.	0 | 2 | 4 | 100 | 66
14. [OS] Different subgroups are sometimes antagonists.	14 | 11 | 10 | 66 | 71

TABLE 7: Results of the questionnaire. Number of answers for each level on the Likert scale are reported: [LW] indicates statements related to Lone Wolf, [RS] to Bottleneck, [BC] to Black Cloud, [OS] to Organisational Silo.

people to lose time understanding source code implemented by others."

Thus, we conclude that the *Black Cloud* effect can cause the introduction of social debt in form of additional work spent to re-construct the knowledge around the activities done by people involved in the smell. At the same time, the semi-structured interviews explained the results achieved in **RQ1** with respect to the low diffusion of this smell: developers can mitigate communication delays by "forcing" members to take part in code reviews, which are one of the main means of knowledge sharing [60].

Lone Wolf. The emergence of this smell is caused by intermittent or indirect communications between members who are supposed to collaborate on the development of a piece of code. The majority of the participants agreed or strongly agreed with both statements (9 and 10) related to *Lone Wolf*. This result indicates that developers perceive the smell as harmful for the success of a project. This was also confirmed when interviewing developers: all of them reported that the presence of a *Lone Wolf* is frequent is practice and can lead to serious problems for the entire community. For example, a developer from Qt said:

"This is terribly frequent. The main problem here is that, if these people leave, we are stuck. If another guy needs to work on the same code, we are stuck. Much more attention should be given to control this kind of problem, as it can cause losing entire weeks of productivity to only understand what a former member has done or having information from other people."

Thus we can claim that the *Lone Wolf* smell is perceived by developers as an important source leading to social debt.

Bottleneck. Our survey participants confirmed that a high degree of formality in the community can be detrimental for communications (149 participants rated this statement as *"Agree"* or *"Strongly Agree"*). However, at the same time they did not believe that regular procedures to communicate and propose changes represent a threat to the success of the project. In other words, while they did not support having an extremely formal community, they agreed that constant communications help in understanding and sharing project knowledge. This became even clearer during the semi-structured interviews: while all participants recognized the frequent presence of this community smell within their communities, they also highlighted that dealing with it can be hard because of the compromise between formality and informality of the development structure. For instance, a participant from VLC declared:

"Project development should be always lean and easy to perform. Developers need freedom and complex organisations make the job hard. However, communicating constantly with other developers is important to correctly carrying out the activities."

With respect to this smell, we claim that extremely formal organisations—typical in cases where the *Bottleneck* smell appears—should be avoided. Thus, we confirm that developers perceive the symptoms behind this smell as a source of social debt; moreover, an efficient way to deal with *Bottleneck* instances seems to be that of having constant (informal) communications about the changes to be performed.

Organisational Silo. The last two statements of Table 7 referred to this smell, and reported situations where there are parallel sub-communities within the same organisation. Our participants widely recognised this as an important problem, since these sub-communities might rarely communicate with each other and, even more importantly, they might become antagonist. This would lead to serious consequences in terms of social debt and additional project costs. This was also reported by all the interviewees. All the 35 interviewees explain that this smell is frequent and can cause serious issues to the community. An interesting comment is reported below, and refers to a participant from Eclipse:

"Unfortunately, in open source this problem is extremely frequent because of the distributed nature of such projects. This is beyond doubts a problem, as having sub-communities naturally lead to missing communications that can potentially cause serious technical problems."

We therefore confirm that the presence of *Organisational Silo* is perceived by developers as a negative pattern in the structure of a software community.

Summary for RQ2. We can conclude that developers perceive the presence of all community smells considered in the study, identifying them as important sources for the emergence of social debt. Sometimes, as in the case of *Black Cloud*, developers learn to cope with these smells in an orderly and governed fashion, reflecting an organised and coherent whole. Further study of the emergence of community smells and any connected (causal) socio-technical factors may foster the discovery of best-fit governance patterns for open- and closed-source projects.

5.3 Relations Among Community Smells and State of the Art Quality Factors

This section reports on our findings in correlating the occurrence of community smells across our dataset with socio-technical quality factors and theories from the state of the art. More than verifying theories, we aimed at finding empirical thresholds that indicate **stability** of smells, that is, levels at which the number of smells no longer increases. We now discuss the quality factors and the stability thresholds associated with each. Specifically, following guidelines from the research literature [61] the thresholds were elicited as the level above which the frequency of datapoints for a certain correlation became less than 20%; that is, the cutpoint where frequency of datapoints becomes more and more statistically irrelevant.

5.3.1 Community Smells vs. Sponsored Developers

The research literature suggests that a higher number of paid developers (sponsored by commercial companies or self-employed) is correlated to a higher attractiveness and health of an Open-Source community [62]. Our data, however, indicates that a higher number of sponsored developers is associated with a linear increase of the number of occurrences of *Organisational Silo* Effect and *Lone Wolf* community smells in 25% of our dataset. It is worth noting that, in the context of **RQ1**, we also reported a weak correlation between the number of contributors and the presence of all smells, with the exception of the *Black Cloud* effect. This result does not contradict the findings of **RQ3**; rather, it highlights the dependence between the role of *sponsored* vs. non-sponsored developers and the levels of different smells.

Community Smells Stability Threshold. We observe a threshold of **10** trimestral sponsored developers above which the number of detected Community Smells grows super-linearly above median values.

Moreover, the role of sponsored developers in the generation of additional Community Smells has different effects depending on the size of the software project:

- In projects with fewer than 50 trimestral community members, socio-technical quality factors related to the identification of sponsored developers were not correlated at all to the generation of additional Community Smells;
- In projects with more than 50 but fewer than 150 trimestral community members, the occurrences of *Organisational Silo* Effect and *Lone Wolf* community smells was not only positively correlated to the number of sponsored developers in a community, but also to the number of sponsored developers who were core developers. Furthermore, a higher ratio of sponsored core developers was associated with an increase in the occurrences of the *Organisational Silo* Effect. This finding highlights that in software communities with 50-150 trimestral members, *core sponsored developers tend to isolate themselves and not participate in project communication channels*.

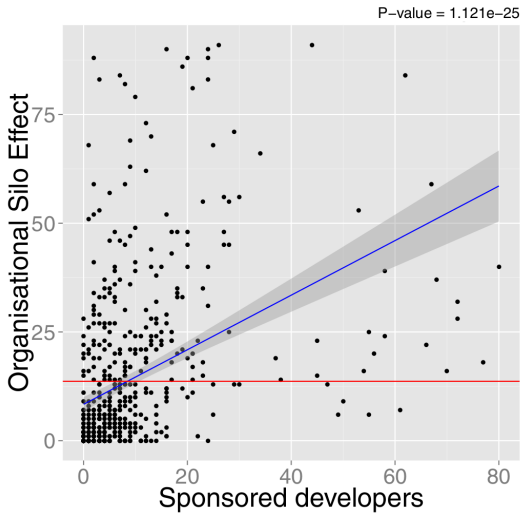


Fig. 9: Community Smells vs. Sponsored Developers.

5.3.2 Community Smells vs. Temporal and Geographic Dispersion

The research literature suggests that temporal and geographic dispersion generates socio-technical issues across software development communities affecting product outcomes and qualities [63]. Therefore, it was expected that the number of time-zones involved in a software development community—a proxy for geographic and temporal dispersion—would be positively correlated with an increase in Community Smells. But this correlation was detected in just 5% of analysed projects. A possible explanation is that, nowadays, distributed development is common in software development and only implies delays in communications among developers. Moreover, FLOSS is founded on the concept of distributed software development and so coordination mechanisms are built. These mechanisms, we postulate, help to mitigate the development of Community Smells due to temporal and geographic dispersion.

5.3.3 Community Smells vs. Socio-Technical Congruence

Cataldo et al. [64] remark that higher socio-technical congruence is correlated to higher software development performance and thus, considering the concept of socio-technical congruence as an indicator of the alignment between an organisational structure and its technical requirements, it was reasonable to suppose that a higher level of socio-technical congruence would be associated with fewer occurrences of community smells. Our results support this hypothesis

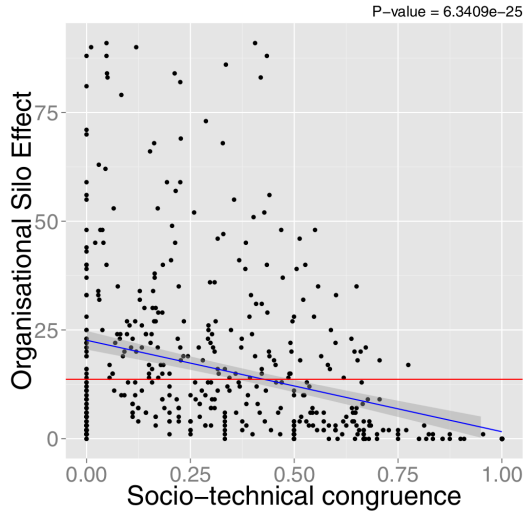


Fig. 10: Community Smells vs. Socio-Technical Congruence.

since, in over 50% of analysed projects, an increase in socio-technical congruence strongly correlates to a decrease of two smells—*Organisational Silo* and *Lone Wolf*. Also, considering the scatter plot in Figure 10 and the regression therein, it was possible to identify a *quality threshold of 0.5 for socio-technical congruence*. Over this threshold the number of detected community smells were over median values.

5.3.4 Community Smells vs. Community Structure

A number of community structure factors were also investigated as possible predictors for community smells. For example, the number of core community members correlates to additional occurrences of *Organisational Silo* (27% of analysed projects), *Lone Wolf* (28% of analysed projects) and *Bottleneck* (30% of analysed projects) smells. Specifically, our data shows that an increment in the number of core developers belonging to the collaboration DSN was associated with higher numbers of *Organisational Silo* in 80% of analysed projects and of *Lone Wolf* in 93% of analysed projects. It is important to highlight that this positive correlation was found both with respect to the number of developers who were considered core members in just the collaboration DSN, and with respect to the number of developers who were considered core members of the collaboration and the communication DSNs. Therefore, an increment of any type of core developer in the collaboration DSN was correlated to an increment of the number of occurrences of *Organisational Silo* and *Lone Wolf*. The incidence of core developers in the generation of additional Community Smells is confirmed by the relation between a higher ratio of peripheral developers and a decrease of the number of occurrences of *Organisational Silo* and *Lone Wolf*, in 42% and in 52% of analysed projects respectively. Furthermore, since a higher truck number [65] is strongly related to the density of the collaboration DSN, it follows that the truck-number itself relates strongly to the emergence of more *Organisational Silo* and *Lone Wolf* smells.

Additionally, in 27% of analysed projects, an increase in the number of core community members belonging to the communication DSN correlates to a higher number of occurrences of the *Bottleneck* smell. In projects with more than 150 trimestral community members, the correlation between the number of core community members belonging to the global or communication DSNs and the number of occurrences of *Bottleneck* is found to be irrelevant. A possible explanation can be that within big communities the influence of core community members tends to decrease; core community members tend to lose their role of unique knowledge and information brokers in larger development communities [66]. This explanation is also motivated by further analyses which revealed that in 43% of projects, a higher truck number related to the communication DSN, thus a higher ratio of peripheral community members was correlated to an increment of the number of occurrences of *Bottleneck*. Therefore, the *Bottleneck* smell is generated by both core and peripheral community members belonging to the communication DSN but additional occurrences of *Bottleneck* generated by core members tend to be irrelevant in big communities. Considering the scatter plots represented in Figure 11 and the linear regressions therein, it was possible to identify the following *quality thresholds* for which the amount of detected Community Smells were above the average values:

Community Quality Thresholds.

- *25 trimestral core community members in the global DSN*. The quality threshold with respect to *Lone Wolf* and *Bottleneck* smells was a bit higher (30 in both cases), but since the threshold with respect to *Organisational Silo* was 25, it was selected as the global threshold for the number of core community members within the global DSN;
- *9 trimestral core developers in the collaboration DSN;*
- *30 trimestral core community members in the communication DSN;*
- *0.8 for the truck number of the collaboration DSN;*
- *0.55 for the truck number of the communication DSN.*

5.3.5 Community Smells vs. Number of Quitters

Within small communities, the quitting of members previously involved in Community Smells generates additional smells. In projects with fewer than 50 trimestral community members a positive correlation was found between the number of members who left the community and who were implicated in at least one Community Smell in the previous 3 months, and the number of occurrences of *Organisational Silo* and *Lone Wolf* smells.

Even if the *centrality* of the global Developer Social Network was not associated with an increase or decrease of the number of occurrences of community smells at a global level, several strong correlations were found considering different categories of FLOSS development communities:

- in projects with fewer than 50 trimestral community members, the increase of closeness centrality was associated with additional occurrences of the *Organisational Silo* smell;
- in projects with more than 150 trimestral community members, the increase of betweenness centrality was

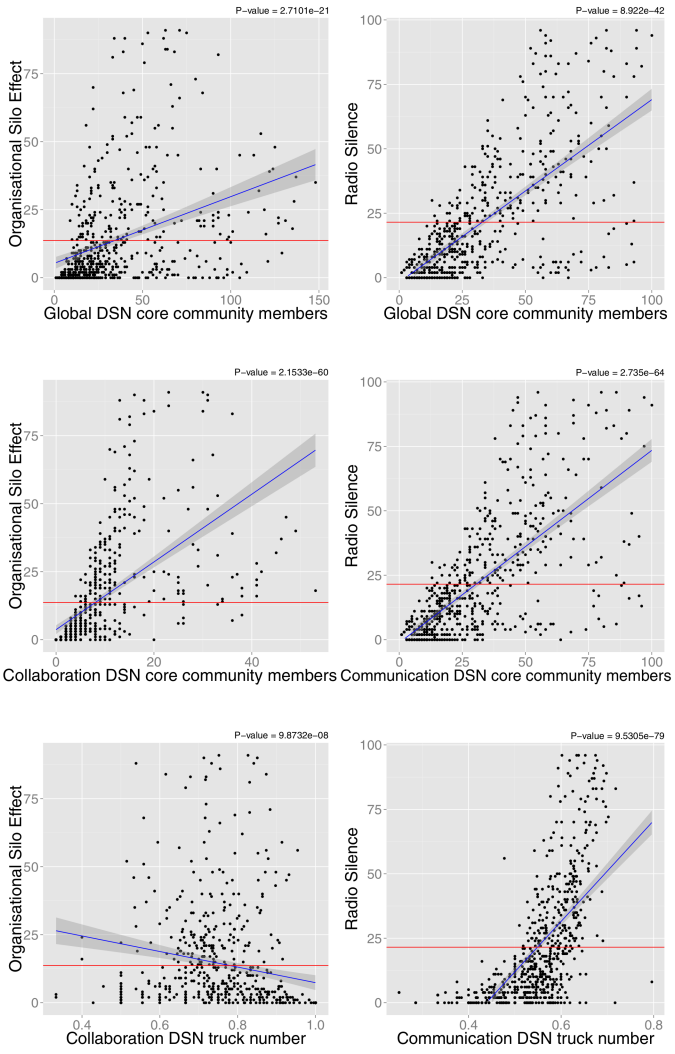


Fig. 11: Community Smells vs. Community Structure Social Network Features.

associated with additional occurrences of *Black Cloud* smell;

Table 8 shows an overview of all the relevant correlations found between quality factors from the state of the art (see Tab. 3) and Community Smells; positive correlations are identified by the "+" sign while the negative ones with a "-".

5.3.6 Community Smells vs. Turnover

Cataldo et al. [67] show that if a software developer community is stable, then its socio-technical congruence increases linearly over time. Thus, it was reasonable to hypothesise that a lower turnover is associated with a lower number of Community Smells. Our results reveal that this is not true in general. Rather, in the case of *Lone Wolf* effects, a higher turnover of *core* developers belonging to the collaboration DSN correlates strongly to a decrease of the number of occurrences of the Smell. In projects with fewer than 50 trimestral community members this correlation becomes irrelevant; none of the turnover metrics were found to have a correlation with any community smell in such projects.

Quality factor (ID)	Organisational Silo Effect	Bottleneck	Lone Wolf
devs	+	+	+
ml.only.devs		+	
code.only.devs	+		+
ml.code.devs		+	
perc.ml.only.devs	-		-
perc.code.only.devs	+		+
sponsored.devs	+		+
st.congruence	-		-
communicability	-		-
ratio.smelly.devs	+	+	+
core.global.devs	+	+	+
core.mail.devs		+	
core.code.devs	+		+
mail.truck		+	
code.truck	-		-
mail.only.core.devs		+	
code.only.core.devs	+		+
ml.code.core.devs	+		+
ratio.mail.only.core	-		-
ratio.code.only.core	+		+
core.code.turnover			-
mail.mod		-	

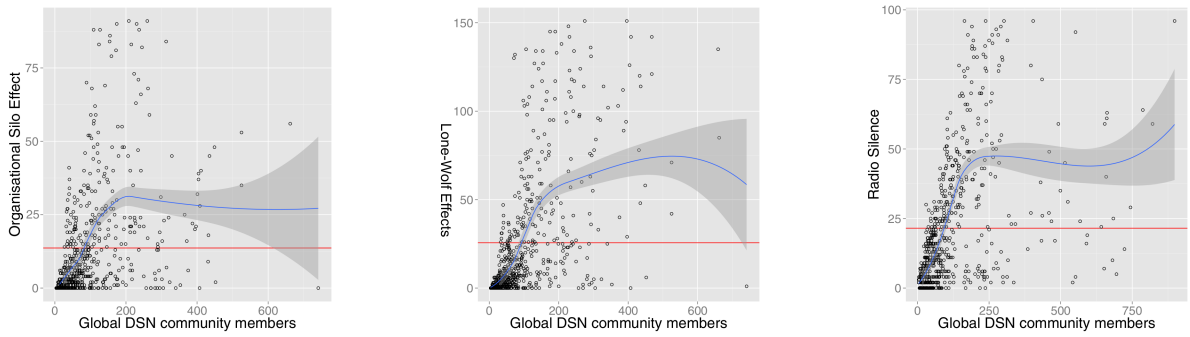
TABLE 8: Summary of quality factors correlated to Community Smells. The *Black Cloud* smell is missing and largely absent across our dataset.

Conversely, in projects with more than 150 trimestral community members the turnover of core developers negatively correlates with the emergence of additional *Organisational Silo* smells and the turnover of global core community members was found to be negatively correlated with both *Organisational Silo* and *Lone Wolf* community smells. In summary, even if it was evident that turnover of core developers does in fact influence the presence of community smells, it was not possible to identify a quality threshold.

5.3.7 Discussion: Relations with State of the Art Factors

In our investigation of factors from the state of the art, we noticed several thresholds playing a role in community smell emergence. However, we noticed that the only factor which strongly correlates with all smells reported across our dataset was the number of community members. It is not surprising that community smells increase as the number of community members increases but, perhaps more importantly, the number of occurrences of all community smells grows **quadratically** until the threshold of **200** community members is reached. After this threshold the occurrences tend to stabilise (see Fig. 12).

In combination with the above number of 200 developers recall that several factors fluctuated around a threshold of 50 members. This also means that major fluctuations of community smells fluctuate in the range $50<150<200$, a range very close to Dunbar's Number [68], [69], [70] of 150. This number theoretically dictates the number of people in a community required to reach organisational saturation and stability. Further investigation of this and similar organisational theories from SNA may reveal even more stability thresholds and characteristics to be used for improved governance, participation, engagement, and sustainability of open-source communities at large.



(a) Organisational silo effect (b) Lone Wolf effect (c) Bottleneck

Fig. 12: Scatter plots for global DSN community members and three community smells. The threshold around 200 members is evident.

Summary for RQ3. We conclude that several factors from the state of the art relate strongly to community smells. Not surprisingly, factors that are health indicators for community structure (e.g., socio-technical congruence) are correlated with smells: the better the congruence the fewer smells we find. Conversely, we report several relations with theories and conjectures from organisations and social-networks research which are still unproven and are not investigated in the scope of existing software engineering research.

5.4 Refactoring Software Communities: Smells and Factors Trade-offs

Our data indicates that community smells and socio-technical factors may jointly be used to refactor software communities for improved organizational quality. In that respect, however, several observations can be made.

First, there are underlying forces in action which are yet to be discovered and elaborated fully and that relate to the counterbalance between factors and their mediating role with respect to community smells. For example, plots in Fig. 11 outline the interactions between several smells and a specific quality factor but the cross-relations between factors and their counterbalancing effect needs to be weighted against the single factor's role as a mediator for more or less community smells of a specific type. In the specific case highlighted by Fig. 11 the bottom-left plot points out to the clear relation between radio-silence effects and the truck factor; in that specific instance, however, there exist several outliers wherefore a high truck factor corresponds to few smells and the opposite conditions are true as well. The aforementioned conditions reflect a trade-off that needs to be evaluated when fine-tuning with community structures, e.g., with specific communication protocols. For example, reducing the truck factor by increasing the modularization of the community positively influences the ratio of radio silence effects, but, at the same time, the action weighs negatively on *Organisational Siloes* which are connected to overly modularised community structures.

Similarly, organisational structures are fluidly evolving networks [71] where sub-optimal effects often occur in stages following a fuzzy logic at best [72]. Consider for example the plot in Fig. 10: the plot distinctively identifies at least three stages of considerable socio-technical congruence value density, namely, STC = 0, STC $\simeq$ 0,30 and STC $>$ 0,50; given these three stages, practitioners and community leaders would need to experiment with each along with different communication and/or collaboration protocols in an effort to strike a balance between their own desired/optimal values of STC with respect to other desired parameter levels. For example, focusing on Fig. 10, the stage in which STC = 0 offers evidence of considerable variance between the numbers of *Organisational Siloes*, which itself reflects the existence of other possible mediators to be acted upon.

In summary, these aforementioned observations clearly indicate the need for further quantitative and qualitative empirical experimentation on the route to figuring out appropriate community refactoring mechanisms and patterns. Said patterns are likely to consist of interactions among smells and sets of socio-technical factors. This work acts a first keystone towards this exploration.

5.5 Observations and Impact

The findings captured in the previous sections have several impacts in the state of research and practice in software engineering.

First, the thresholds we identified could be used as stability metrics to be combined with data and inputs from sites such as OpenHub and Bitergia, to evaluate the participation or adoption of specific open-source products. For example, community activity and stability around a specific software component may need to be evaluated in order to reduce the risks connected to adopting that component in the first place. Our results can offer a lens of analysis and a series of thresholds to evaluate whether the quality of a certain community is sufficient enough for adoption. At the same time, community smells and their detection can be used as a means to mitigate the connected risks — for example, practitioners can re-modularise their products to work around smelly components and communities.

Furthermore, from a research perspective, academicians can further the study and understanding over the relations of more complex patterns of software engineering organisational structures with respect to more complex patterns of software (e.g., architectural or design patterns, etc.) as well

as known code quality issues and metrics in the scope of software maintenance and evolution activities.

In addition, from a closed-source perspective, the tool we provide can offer a valuable basis to enact continuous community improvement in the scope of DevOps pipelines wherefore the quality of the organisation is as much important as the quality of the product underneath.

6 Verifiability and Threats to Validity

6.1 Verifiability

As pointed out throughout this article, we encourage replication of this study to more deeply study community smells and to characterise the forces around social debt. To support this goal, first we have created a complete report [73] with the full details of the study design, from the literature analysis that drove its inception to the entire span of the evaluation (partially) reported in this article. Second, we prepared a detailed explanation of the set-up and execution for the data-mining and analysis tools used throughout our work[5]. Finally, we prepared an online appendix [45] reporting fine-grained details on the analyses conducted. We hope this material can support both replication of this study and the further study of community smells from other perspectives.

6.2 Threats to validity

In the following, we discuss the threats that might have affected our findings, and how we have mitigated these threats.

Construct Validity. Threats to construct validity are related to the relationships between theory and observation. Generally, this threat is constituted by imprecision in performed measurements. The part of the study related to community smell identification is affected by construct validity since the metrics of our quality model may have been implemented in our tool in a “biased" way. For example, the operationalisation that we exploit stems from previous work of Joblin et al., who refined and operationalised the concept of verified community [14]. Community structures are detected through a series of methodological triangulations (code function level, commit level and communication level approaches) validated based on comparisons with null-models, hence the term ‘verified’. This notwithstanding, the operationalisation they provide could still suffer from the consequence of unknown organisational circumstances. Furthermore, the operationalisation we adopt looks for instances of smells following the definitions in the strictest sense. That is, for example, in the case of the *Black Cloud* and *Lone Wolf* effects, no communication needs to be observed but look for this effect over a period of two consecutive time-windows (i.e., 6 months). This is because the smell has a meaningful observable effect when manifests over such a time window; at the same time, the smell may persist even if after such a time window there is in fact communication which is intermittent or by the interposition of an external “intruder".

Likewise, the survey was phrased to identify such instances, since they were less common and more difficult to identify based on our quantitative empirical data. These approaches are themselves empirically-derived and may be compromising the validity of our theoretical constructs. Also, part of our study is based on survey results, so construct validity may be compromised by developer perceptions. To mitigate possible biases, we first inquired participants indirectly, as recommended by Fisher [51]: this was due to the fact that participants of survey studies might be reluctant to express negative or controversial opinions on topics they are too much involved in. Then, we perform confirmatory semi-structured interviews to ask developers direct experiences with community smells. Besides employing such established research techniques, we were not able to mitigate these threats further. Therefore the study remains potentially affected.

Internal Validity. Threats to internal validity are related to factors that could have influenced our results but which were not accounted for. Concerning community smell identification, one factor that might have impacted on our ability to correctly detect smells concerned how the communication network was built. Specifically, we relied on mailing lists to mine the communications among developers, but there might be additional channels where community members communicate with each other. While this is a limitation of our approach, it is worth discussing this point further by clarifying three important points:

- 95% of the open-source communities investigated in this work (and, in fact, the vast majority of existing communities) explicitly state in the contribution guidelines that mailing lists represent the main channel to give and get updates about the status of the project. For instance, the first requirement to get involved in the Apache Mahout project[6] is to join the `'user'`, `'development'`, and `'commit'` lists, to help others, join discussions of changes, and be informed of new commits, respectively. Moreover, the project explicitly states that *“discussions at Apache happen on the mailing list”*. In other words, mailing lists represent the main communication channel in open-source projects: we believe that this aspect supports our choice of methodology to identify community smells.
- Despite the prominent role of mailing lists, we cannot exclude that some communications are conducted by developers using other channels (e.g., Skype). However, the mining of such channels would (i) not be practical, e.g., there is no way to access the Skype data of a developer, and (ii) more importantly, have important privacy implications.
- We included confirmatory survey questions as part of the questionnaire used to infer our original quality model. Survey results [73] strongly support that project mailing lists were indeed the key communication channel used across the community for almost 87% of survey respondents.

For these reasons, we believe that mailing lists represent a sufficiently accurate information source to study communication and collaboration among developers. As further proof, it is worth mentioning that the CodeFace approach was empirically validated by the original authors [74]: one

5. https://github.com/smnmgn/codeface

6. https://mahout.apache.org/developers/how-to-contribute

of their main findings was that the verified community structure detected by CODEFACE was in line with the developer networks built using the manual certificate-of-origin reporting systems that documented code changes. Thus, the methodology appears suitable to properly capture real-world collaborations.

In the context of **RQ2** we evaluated developer perception of community smells by conducting a questionnaire involving 172 contributors coming from 11 of the 60 projects considered. While this sample might not reflect the perceptions of developers in other communities, we argue that our analysis provides important insights on the perceived impact of community smells. Furthermore, the additional confirmatory analysis made by means of semi-structured interviews allows us to be more confident about the reported findings. Notwithstanding, additional experiments and replications of our work would be desirable. A factor that might have influenced the respondents' perception of community smells could be their level of experience in open-source projects. To control for this aspect we computed two well-established metrics—commit- and project-tenure [59]—by mining Github repositories, and we computed statistical tests to verify the correlation between developers' experience and perception of community smells. However, further analyses of how experience impacts community smells should be carried out in future studies.

Furthermore, we must acknowledge that the exploration of our study subject (namely community structure patterns) is subject to many biases from the members of the communities. On one hand, it was our research design decision to confirm the validity of the tool in detecting community smells, reflecting actual sub-optimal conditions, thus avoiding, in the scope of **RQ2**, explicit questions phrased to directly assess the smells themselves. This decision was made explicitly to avoid any bias connected to our confirmatory evaluation connotation [75], [76]. On the other hand, this still constitutes a threat to validity—we were not able to directly and consistently confirm all questions with their single nasty effects, which would require an additional confirmatory study (which is currently being planned).

In the context of **RQ3**, when looking at the relationship between the community smells and socio-technical metrics we did not normalise the number of smells but relied on the absolute values. This might have potentially affected our interpretations: to understand whether and how much the results would have been affected, we re-run our analyses while normalising the number of community smells based on the number of community members: in other words, we considered the relation between the density of community smells and the other socio-technical factors considered. As a result, the findings achieved do not change when compared to those reported in Section 5.3. This indicates that, independently from the normalisation, the relations between community smells and socio-technical factors highlighted hold. A complete report of these additional analyses is available in our online appendix [45].

External Validity. Threats to external validity are related to the generalisation of obtained results. Our tool support CODEFACE4SMELLS currently considers only four indicators of social debt but there are doubtless many community smells yet to uncover. Also, in our evaluation, our analysis was made over a total of 60 FLOSS projects, strengthening the generality of our findings. However, these results might be influenced by the temporal window we selected for our analysis (3 months) as well as myriad other factors that we could not control for.

7 RELATED WORK

This section describes the related work that is the foundation for our contributions, ranging from research on Conway's Law [77] to efforts that capture and empirically evaluate the impact of organisational structure quality to code and general product quality. We also review research on the dimensions that affect community quality in a global software engineering context.

Since the original inception of Conway's Law, positing the relationship between the structure of a system and the structure of the organisation that designed it, several studies have tried to understand more about this intriguing relation. For example, the works by Cataldo et al. and Herbsleb et al. around socio-technical congruence [7], [8], [78] study software development as a social-technical activity, in which the technical and the social components need to be aligned to succeed. These and similar works introduce valuable socio-technical factors to be addressed and tracked for software communities to succeed. The fundamental component, Herbsleb [79], Damian [80] and others say, is to achieve an effective coordination among teams, whose organisational structure is a key dimension that should be considered as much as project plans, processes and coordination mechanisms [78].

These insights are also reinforced by evidence regarding the opposite influence — for example, in 2010 Colfer and Baldwin [81] complemented Conway's law verifying the validity of their "mirroring hypothesis", which assumed that the organisational patterns of a development community (e.g. team co-membership and geographic distribution, communication links) mirror the technical dependency patterns of the software under development. With respect to these works, the results outlined in this paper take a non-trivial further step. Building upon the state of the art, our intent was to gather metrics to evaluate the health status of software engineering communities, formulating the thresholds needed to automatically detect and "diagnose" community problems we observed in industry that lead to social debt [13]. Although mainly validated in open-source projects our model and appraisal approach and tools can be used effectively on any DSN [4], given that the foundations of the theory behind the model (i.e., social debt and community smells) were elaborated solely on large-scale distributed industrial case-studies [9], [17].

Moreover, there is much research related to our intensions, e.g., in terms of providing effective ways to study the success (and failure) factors of global software engineering. For example, In 2002 Herbsleb and Mockus carried out an initial social-network analysis featuring mailing lists, code repositories and issue tracking systems of two important FLOSS projects and analysed developer participation and community metrics [82], [83]. Herbsleb and Mockus concluded that higher levels of organisational modularity indi-

cate a lower coordination need and that a "communication-only" approach in a distributed software development environment, such as a FLOSS ecosystem, does not scale because communication channels will be overwhelmed as the size and the complexity of the project and community will grow [83]. Similar results were also confirmed by Bird et al. a number of years later both in open- [84] and closed-source [85] projects. Also, Nagappan et al. [1] analysed the relation between organisational structure and software quality. They proposed eight measures to quantify organisational complexity from the code viewpoint and empirically evaluated their efficacy to identify failure-prone binaries in a commercial project. The failure-proneness prediction model based on the organisational metrics outperformed traditional technical metrics (e.g. code churn, code complexity, LOC). Even more recently, the work of Lavallee et al. [86] analysed the relationships between several organisational factors and their impact on developers' working conditions and performance. Lavallee et al. identified several socio-technical organisational issues (i.e., community smells [9]) that compromise software quality and its success, e.g., the truck number across the community structure.

In the same way, we argue that a key number of organisational and socio-technical dynamics, inherited from social-network research [10], need to be studied and tracked to ensure the performance and organisational health for software development communities. In this paper we present a tool to aid in this endeavour, constructed from prior theories and from the social debt and community smells we observed through industry case-study research. Our tool was evaluated on over 60 open-source projects and did indeed reveal key insights on communities health. To the best of our knowledge the tool we propose is the first of its kind: its key benefit is that of offering confirmed community quality dimensions and thresholds for diagnosing potentially expensive organisational and socio-technical community smells that may be introducing friction in DevOps lifecycles.

8 Conclusions

8.1 Contributions

This paper introduces, elaborates, and evaluates the diffuseness, developer perception, and impact of community smells—detrimental socio-technical circumstances that increase social debt [9], [17]—across large, distributed open-source communities. We contribute to the state of the art by quantitatively illustrating the occurrences of community smells across over 100 years worth of historical commit and communication data for 60 open-source projects. Also, we confirm the value of our contributions showing, via a survey, the impacts that open-source developers perceive regarding smells in their own communities. Finally, we contribute to the state of the art using known and previously studied socio-technical quality factors to discover several quality thresholds that strongly correlate with community smell stability over time.

We have made all of our tools and data available for other researchers to use, and for replication purposes.

8.2 Conclusion

We conclude that community smells are indeed a force to be reckoned with. But, just as important, we have discovered several thresholds over measurable socio-technical quality factors that aid in the prediction and mitigation of community smells along with any connected nasty effects.

8.3 Future Work

In the future work we plan to strengthen the usability and evaluation of our tool, experimenting further with its dimensions based on additional evaluation. Also, we plan to evaluate the tool from a technical perspective, i.e., answering the research question "what factors from the model reflect better technical quality?". In addition, we plan to merge our tool support with the main CodeFace distribution, aiding its software community improvement potential.

Furthermore, in recognising the limitations of the current implementation of CodeFace4Smells tool, we plan to further provide additional operationalisations that cover the remaining community smells which were not further explored in the scope of this study. For example, stemming from the remaining community smells identified in previous work [9], we aim at further investigating what other community smells can be operationalised using open-data available from software repositories other than GitHub, as well as the feasibility of using non-conventional techniques (e.g., Neuro-linguistic programming, natural language processing, sentiment analysis, etc.) to operationalise the remaining community smells from the state of the art. Finally, the patterns and community smells we analyse were operationalised in an rather 'assertive' fashion, meaning that the patterns themselves have no weights associated to them (e.g., if a *Lone Wolf* is a more experienced developer or whether a *Bottleneck* effect separates very large communities) — disregarding these aspects constitutes a required simplification for the nature of this study but should be addressed with future work.

Acknowledgements

The authors would like to thank Dr. Simone Magnoni to the definition and refinement of the contents of this study and to Prof. Dr. Elisabetta Di Nitto for the feedback she gave during the inception of this study. Dr. Palomba gratefully acknowledges the support of the SNSF Project named "Data-driven Contemporary Code Review" (No. PP00P2_170529).

References

[1] N. Nagappan, B. Murphy, and V. Basili, "The influence of organizational structure on software quality: an empirical case study," in *International conference on Software engineering*. Leipzig, Germany: IEEE, May 2008, pp. 521–530. [Online]. Available: http://doi.acm.org/10.1145/1368088.1368160

[2] M. Saeki, "Communication, collaboration, and cooperation in software development-how should we support group work in software development?" in *APSEC*. IEEE Computer Society, 1995, pp. 12–21. [Online]. Available: http://dblp.uni-trier.de/db/conf/apsec/apsec1995.html#Saeki95

[3] Y. Dittrich, J. Norbjerg, P. Tell, and L. Bendix, "Researching cooperation and communication in continuous software engineering." in *CHASE@ICSE*, H. Sharp, C. R. B. de Souza, D. Graziotin, M. Levy, and D. Socha, Eds. ACM, 2018, pp. 87–90. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/chase2018.html#DittrichNTB18

[4] M. Pinzger, N. Nagappan, and B. Murphy, "Can developer social networks predict failures?" in *FSE '08: Proceedings of the 16th ACM SIGSOFT international symposium on Foundations of software engineering*, 2008.

[5] C. Bird, N. Nagappan, H. C. Gall, B. Murphy, and P. T. Devanbu, "Putting it all together: Using socio-technical networks to predict failures." in *ISSRE*. IEEE Computer Society, 2009, pp. 109–119. [Online]. Available: http://dblp.uni-trier.de/db/conf/issre/issre2009.html#BirdNGMD09

[6] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "Social debt in software engineering: insights from industry." *J. Internet Services and Applications*, vol. 6, no. 1, pp. 10:1–10:17, 2015.

[7] 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 *ESEM*, H. D. Rombach, S. G. Elbaum, and J. Munch, Eds. ACM, 2008, pp. 2–11. [Online]. Available: http://dblp.uni-trier.de/db/conf/esem/esem2008.html#CataldoHC08

[8] J. D. Herbsleb, M. Cataldo, D. Damian, P. T. Devanbu, S. M. Easterbrook, and A. Mockus, "Socio-technical congruence (stc 2008)." in *ICSE Companion*. ACM, 2008, pp. 1027–1028, 978-1-60558-079-1. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/icsec2008.html#HerbslebCDDEM08

[9] D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "Social debt in software engineering: insights from industry." *J. Internet Services and Applications*, vol. 6, no. 1, pp. 10:1–10:17, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/jisa/jisa6.html#TamburriKLV15

[10] D. A. Tamburri, P. Lago, and H. v. Vliet, "Organizational social structures for software engineering," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 3:1–3:35, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2522968.2522971

[11] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[12] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, and A. De Lucia, "On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1188–1221, 2018.

[13] D. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "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.

[14] M. Joblin, W. Mauerer, S. Apel, J. Siegmund, and D. Riehle, "From developer networks to verified communities: A fine-grained approach." in *ICSE (1)*, A. Bertolino, G. Canfora, and S. G. Elbaum, Eds. IEEE Computer Society, 2015, pp. 563–573. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/icse2015-1.html#JoblinMASR15

[15] E. Allman, "Managing technical debt." *Commun. ACM*, vol. 55, no. 5, pp. 50–55, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/cacm/cacm55.html#Allman12

[16] C. B. Seaman, R. L. Nord, P. Kruchten, and I. Ozkaya, "Technical debt: Beyond definition to understanding report on the sixth international workshop on managing technical debt." *ACM SIGSOFT Software Engineering Notes*, vol. 40, no. 2, pp. 32–34, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/sigsoft/sigsoft40.html#SeamanNKO15

[17] D. A. Tamburri and E. D. Nitto, "When software architecture leads to social debt." in *WICSA*, L. Bass, P. Lago, and P. Kruchten, Eds. IEEE Computer Society, 2015, pp. 61–64. [Online]. Available: http://dblp.uni-trier.de/db/conf/wicsa/wicsa2015.html#TamburriN15

[18] S. Wasserman and K. Faust, *Social Network Analysis. Methods and Applications*. Cambridge University Press, 1994.

[19] D. Pugh and M. Weber, Eds., *Organization theory*, 2nd ed., ser. Penguin education. Harmondsworth, Middlesex [u.a.]: Penguin Books, 1984. [Online]. Available: http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+022044973&sourceid=fbw_bibsonomy

[20] A. Rapoport, "Contribution to the theory of random and biased nets," *Bulletin of Mathematical Biology*, vol. 19, pp. 257–277, 1957. [Online]. Available: http://dx.doi.org/10.1007/BF02478417

[21] R. K. Merton, *Social Theory and Social Structure*. New York, London: The Free Press, 1968.

[22] M. Prilla, "Supporting collaborative reflection at work: A socio-technical analysis," *AIS Transactions of Human-Computer Interaction*, 2015 (upcoming).

[23] J. S. Coleman, *Microfoundations and Macrosocial Behavior*. Berkley, CA: University of California Press, 1987, pp. 153–176.

[24] M. M. Burnett and B. A. Myers, "Future of end-user software engineering: beyond the silos." in *FOSE*, J. D. Herbsleb and M. B. Dwyer, Eds. ACM, 2014, pp. 201–211. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/fose2014.html#BurnettM14

[25] M.-A. D. Storey, L. Singer, B. Cleary, F. M. F. Filho, and A. Zagalsky, "The (r) evolution of social media in software engineering." in *FOSE*, J. D. Herbsleb and M. B. Dwyer, Eds. ACM, 2014, pp. 100–116. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/fose2014.html#StoreySCFZ14

[26] S. Alter, "Sidestepping the it artifact, scrapping the is silo, and laying claim to "systems in organizations"." *CAIS*, vol. 12, p. 30, 2003. [Online]. Available: http://dblp.uni-trier.de/db/journals/cais/cais12.html#Alter03b

[27] 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

[28] D. Gotterbarn, "Professional trust and privacy: the dangers of silo thinking." *Inroads*, vol. 3, no. 2, pp. 4–5, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/inroads/inroads3.html#Gotterbarn12

[29] W. van Osch, C. Steinfield, and Y. Zhao, "Spanning the boundary: Measuring the realized and lifecycle impact of distinct boundary spanning activities on project success and completion." in *HICSS*. AIS Electronic Library (AISeL), 2017. [Online]. Available: http://dblp.uni-trier.de/db/conf/hicss/hicss2017.html#OschSZ17

[30] D. A. Tamburri, R. Kazman, and H. Fahimi, "The architect's role in community shepherding." *IEEE Software*, vol. 33, no. 6, pp. 70–79, 2016.

[31] 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.

[32] K. Stein, "Moving from dyad to triad: a triangular interdependent perspective on brand relationships," Ph.D. dissertation, Uni Mainz, 2013.

[33] F. Leistner, *Connecting organizational silos: Taking knowledge flow management to the next level with social media*. John Wiley & Sons, 2012.

[34] O. Serrat, "Bridging organizational silos," in *Knowledge Solutions*. Springer, 2017, pp. 711–716.

[35] D. A. Tamburri, P. Lago, and H. van Vliet, "Uncovering latent social communities in software development." *IEEE Software*, vol. 30, no. 1, pp. 29–36, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/software/software30.html#TamburriLV13

[36] Q. Huang, H. Liu, and X. Zhong, "The impact of transactive memory systems on team performance." *IT & People*, vol. 26, no. 2, pp. 191–212, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/itp/itp26.html#HuangLZ13

[37] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, 1st ed., ser. Structural analysis in the social sciences. Cambridge University Press, 1994, no. 8.

[38] M. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, September 2003. [Online]. Available: http://arxiv.org/abs/cond-mat/0309508

[39] C. Gkantsidis, M. Mihail, and E. W. Zegura, "The markov chain simulation method for generating connected power law random graphs." in *ALENEX*, R. E. Ladner, Ed. SIAM, 2003, pp. 16–25. [Online]. Available: http://dblp.uni-trier.de/db/conf/alenex/alenex2003.html#GkantsidisMMZ03

[40] M. M. Bersani, F. Marconi, D. A. Tamburri, and F. Palomba, "Formally-correct organizational anti-pattern detection for open-source community health," *Journal of Systems and Software*, Under Review.

[41] S. P. Borgatti, K. M. Carley, and D. Krackhardt, "On the robustness of centrality measures under conditions of

imperfect data," *Social networks*, vol. 28, no. 2, pp. 124–136, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378873305000353

[42] D. Krackhardt and M. Kilduff, "Structure, culture and simmelian ties in entrepreneurial firms." *Social Networks*, vol. 24, no. 3, pp. 279–290, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/socnet/socnet24.html#KrackhardtK02

[43] P. J. Hinds, K. M. Carley, D. Krackhardt, and D. Wholey, "Choosing work group members: Balancing similarity, competence, and familiarity," *Organizational Behavior and Human Decision Processes*, vol. 81, no. 2, pp. 226–251, 2000.

[44] D. Krackhardt, "Predicting with networks: Nonparametric multiple regression analysis of dyadic data," *Social networks*, vol. 10, no. 4, pp. 359–381, 1988.

[45] D. Tamburri, F. Palomba, and R. Kazman, "Exploring community smells in open-source: An automated approach - online appendix - https://www.mediafire.com/folder/d429grc6jah32/," 2018.

[46] Student, "An experimental determination of the probable error of dr spearman's correlation coefficients," *Biometrika*, pp. 263–282, 1921.

[47] J. Cohen, *Statistical power analysis for the behavioral sciences*, 2nd ed. Lawrence Earlbaum Associates, 1988.

[48] K. J. Hunt, N. Shlomo, and J. Addington-Hall, "Participant recruitment in sensitive surveys: a comparative trial of ?opt in?versus ?opt out?approaches," *BMC medical research methodology*, vol. 13, no. 1, p. 3, 2013.

[49] W. Sugar, *Studies of ID practices: A review and synthesis of research on ID current practices*. Springer, 2014.

[50] R. Likert, "A technique for the measurement of attitudes." *Archives of psychology*, 1932.

[51] R. J. Fisher, "Social desirability bias and the validity of indirect questioning," *Journal of consumer research*, vol. 20, no. 2, pp. 303–315, 1993.

[52] G. Hoinville and R. Jowell, "Survey research practice." 1978.

[53] S. Magnoni, D. A. Tamburri, E. Di Nitto, and R. Kazman, "Analyzing quality models for software communities," *Communications of the ACM*, vol. -, p. Under Review, 2017.

[54] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "A novel approach for estimating truck factors," in *Program Comprehension (ICPC), 2016 IEEE 24th International Conference on*. IEEE, 2016, pp. 1–10.

[55] D. A. Tamburri and E. D. Nitto, "When software architecture leads to social debt." in *Software Architecture (WICSA), 2015 12th Working IEEE/IFIP Conference on*, L. Bass, P. Lago, and P. Kruchten, Eds. IEEE Computer Society, 2015, pp. 61–64.

[56] H. Sharp and H. Robinson, "Some social factors of software engineering: the maverick, community and technical practices," in *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4. ACM, 2005, pp. 1–6.

[57] K. Krippendorff, *Content Analysis: An Introduction to Its Methodology (second edition)*. Sage Publications, 2004.

[58] J.-Y. Antoine, J. Villaneau, and A. Lefeuvre, "Weighted krippendorff's alpha is a more reliable metrics for multi-coders ordinal annotations: experimental studies on emotion, opinion and coreference annotation." in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, G. Bouma and Y. Parmentier, Eds. The Association for Computer Linguistics, 2014, pp. 550–559. [Online]. Available: http://dblp.uni-trier.de/db/conf/eacl/eacl2014.html#AntoineVL14

[59] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. 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*. ACM, 2015, pp. 3789–3798.

[60] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *Proceedings of the 2013 international conference on software engineering*. IEEE Press, 2013, pp. 712–721.

[61] N. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd ed. London, UK: International Thomson Computer Press, 1997.

[62] D. Riehle, P. Riemer, C. Kolassa, and M. Schmidt, "Paid vs. volunteer work in open source," in *2014 47th Hawaii International Conference on System Sciences*. IEEE, 2014, pp. 3286–3295.

[63] J. Cho, "Globalization and global software development," *Issues in information systems*, vol. 8, no. 2, pp. 287–290, 2007.

[64] 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 *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 2008, pp. 2–11.

[65] M. Torchiano, F. Ricca, and A. Marchetto, "Is my project's truck factor low?: theoretical and empirical considerations about the truck factor threshold." in *WETSoM*, G. Concas, E. D. Tempero, H. Zhang, and M. D. Penta, Eds. ACM, 2011, pp. 12–18. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/wetsom2011.html#TorchianoRM11

[66] R. Kazman, D. Goldenson, I. Monarch, W. Nichols, and Valetto, "Evaluating the effects of architectural documentation: A case study of a large scale open source project," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 220–260, 2016.

[67] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley, "Identification of coordination requirements: implications for the design of collaboration and awareness tools," in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. ACM, 2006, pp. 353–362.

[68] P. M. Carron, K. Kaski, and R. Dunbar, "Calling dunbar's numbers." *Social Networks*, vol. 47, pp. 151–155, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/socnet/socnet47.html#CarronKD16

[69] B. Goncalves, N. Perra, and A. Vespignani, "Modeling users' activity on twitter networks: Validation of dunbar's number," *PLoS ONE*, vol. 6, no. 8, 2011.

[70] J. Zhao, J. Wu, G. Liu, K. Xu, and G. Chen, "Being rational or aggressive? a revisit to dunbar's number in online social networks," *CoRR*, vol. abs/1011.1547, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1011.html#abs-1011-1547

[71] K. M. Carley, "On the evolution of social and organizational networks," *Special Issue of Research in the Sociology of Organizations on Networks In and Around Organizations*, vol. 16, pp. 3–30, 1999.

[72] D. A. Tamburri, F. Palomba, A. Serebrenik, and A. Zaidman, "Discovering community types in open-source: A systematic approach and its evaluation," Jul 2017.

[73] S. Magnoni, D. A. Tamburri, and E. D. Nitto, "A socio-technical quality model for software engineering organisational structures: An empirical study," *Copyright - Politecnico di Milano Master Thesis Series*, vol. https://goo.gl/Y9R4KY, 2016.

[74] M. Joblin, W. Mauerer, S. Apel, J. Siegmund, and D. Riehle, "From developer networks to verified communities: A fine-grained approach." in *Proc. Int'l Conf. on Software Engineering (ICSE)*, A. Bertolino, G. Canfora, and S. G. Elbaum, Eds. IEEE Computer Society, 2015, pp. 563–573.

[75] L. E. C. da Rocha, A. E. Thorson, R. Lambiotte, and F. Liljeros, "Respondent-driven sampling bias induced by clustering and community structure in social networks." *CoRR*, vol. abs/1503.05826, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1503.html#RochaTLL15

[76] Y. Mao, S. Bolouki, and E. Akyol, "Spread of information with confirmation bias in cyber-social networks." *CoRR*, vol. abs/1803.06377, 2018. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1803.html#abs-1803-06377

[77] M. Conway, "How do committees invent?" *Datamation Journal*, pp. 28–31, April 1968.

[78] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An empirical study of global software development: distance and speed," in *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 81–90. [Online]. Available: http://portal.acm.org/citation.cfm?id=381481&dl=GUIDE&coll=GUIDE&CFID=17819395&CFTOKEN=30991748#

[79] J. Herbsleb and R. Grinter, "Architectures, coordination, and distance: Conway's law and beyond," *Software, IEEE*, vol. 16, no. 5, pp. 63 –70, sep/oct 1999.

[80] 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.

[81] L. Colfer and C. Y. Baldwin, "The mirroring hypothesis: Theory, evidence and exceptions," *working paper*, Feb. 2010. [Online]. Available: http://hbswk.hbs.edu/item/6361.html

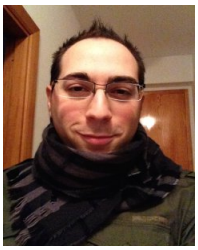
[82] J. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 481–94, 2003.

[83] J. D. Herbsleb and A. Mockus, "Formulation and preliminary test of an empirical theory of coordination in software engineering." in *ESEC / SIGSOFT FSE*, J. Paakki and P. Inverardi, Eds. ACM, 2003, pp. 138–137. [Online]. Available: http://dblp.uni-trier.de/db/conf/sigsoft/fse2003.html#HerbslebM03

[84] C. Bird and N. Nagappan, "Who? where? what? examining distributed development in two large open source projects." in *MSR*, M. Lanza, M. D. Penta, and T. Xie, Eds. IEEE Computer Society, 2012, pp. 237–246. [Online]. Available: http://dblp.uni-trier.de/db/conf/msr/msr2012.html#BirdN12

[85] C. Bird, N. Nagappan, P. T. Devanbu, H. C. Gall, and B. Murphy, "Does distributed development affect software quality? an empirical case study of windows vista." in *ICSE*. IEEE, 2009, pp. 518–528. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/icse2009.html#BirdNDGM09

[86] M. Lavallee and P. N. Robillard, "Why good developers write bad code: An observational case study of the impacts of organizational factors on software quality." in *ICSE (1)*, A. Bertolino, G. Canfora, and S. G. Elbaum, Eds. IEEE Computer Society, 2015, pp. 677–687. [Online]. Available: http://dblp.uni-trier.de/db/conf/icse/icse2015-1.html#LavalleeR15a



Damian A. Tamburri is an Assistant Professor at the Jheronimus Academy of Data Science (JADS) and the Technical University of Eindhoven (TU/e). His research interests include social software engineering, advanced software architectures, design, and analysis tools as well as advanced software-engineering methods and analysis tools. He is on the IEEE Software editorial board and is secretary of the International Federation for Information Processing Working Group on Service-Oriented Computing. Contact him at damianandrew.tamburri@polimi.it or dtamburri@acm.org.

Fabio Palomba is a Senior Research Associate at the University of Zurich, Switzerland. He received the European PhD degree in computer science from the University of Salerno, Italy, in 2017. His research interests include software maintenance and evolution, empirical software engineering, change and defect prediction, green mining and mining software repositories. He serves and has served as a program committee member of international conferences such as MSR, ICPC, ICSME, and others. He is member of the IEEE and ACM. Contact him at palomba@ifi.uzh.ch

Rick Kazman is a professor of information technology management at the University of Hawaii and a principal researcher at Carnegie Mellon University Software Engineering Institute. His research interests include software architecture design and analysis tools, software visualization, and software engineering economics. Kazman received a PhD in computer science from Carnegie Mellon University. Contact him at kazman@hawaii.edu.