Beyond Technical Aspects: How do Community Smells Influence the Intensity of Code Smells?

Fabio Palomba,[1] Damian A. Tamburri,[2] Francesca Arcelli Fontana,[3] Rocco Oliveto,[4] Andy Zaidman,[5] Alexander Serebrenik[2]

[1]**University of Zurich, Switzerland** — [2]Eindhoven University of Technology, The Netherlands
[3]University of Milano-Bicocca, Italy — [4]University of Molise, Italy
[5]Delft University of Technology, The Netherlands

Software engineering is, by nature, a *social* activity and, therefore, social issues might impact software development

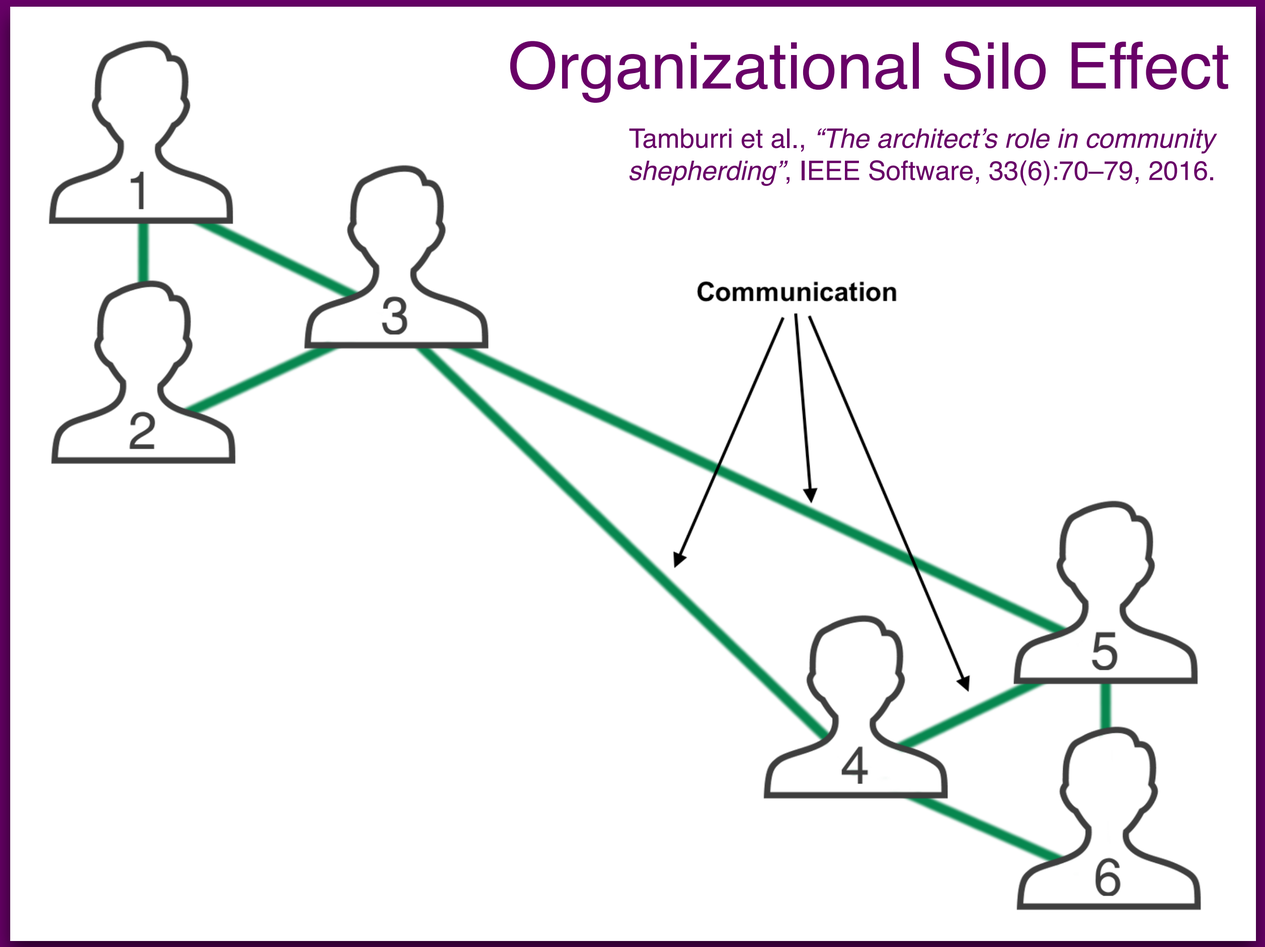
[Conway M. E. - *“How Do committees invent”* Datamation, 14, 4 (1968), 28–31]

Community Smells

A set of socio-technical characteristics and patterns, which may lead to the emergence of social debt

Community Smells

A set of socio-technical characteristics and patterns, which may lead to the emergence of social debt



Community Smells

Some preliminary information

Community smells in OSS

Community smells **regularly** affect open-source communities

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

TSE 2019

Community Smells

Some preliminary information

Community smells in OSS

Community smells **regularly** affect open-source communities

Developers perceive them as highly **harmful for the sustainability** of software projects

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

TSE 2019

Community Smells

Some preliminary information

Community smells in OSS

Community smells **regularly** affect open-source communities

Developers perceive them as highly **harmful for the sustainability** of software projects

Sometimes, developers deal with them **changing the underlying community structure**

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

TSE 2019

Community Smells

Some preliminary information

Community smells in OSS

Community smells **regularly** affect open-source communities

Developers perceive them as highly **harmful for the sustainability** of software projects

Sometimes, developers deal with them **changing the underlying community structure**

Some socio-technical factors can **predict** their emergence

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

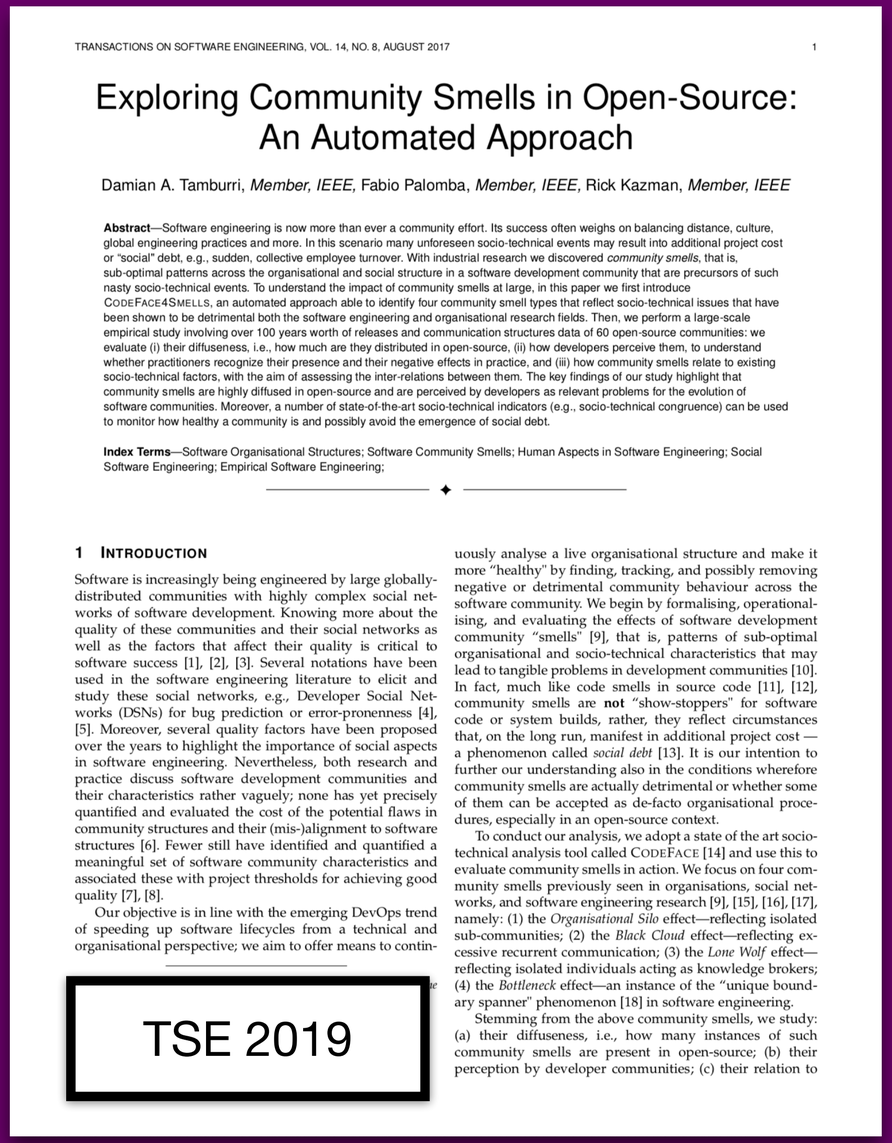
TSE 2019

Community Smells

Some preliminary information

To sum up

Community smells have clear relations with **social** debt



TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 14, NO. 8, AUGUST 2017 1

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

TSE 2019

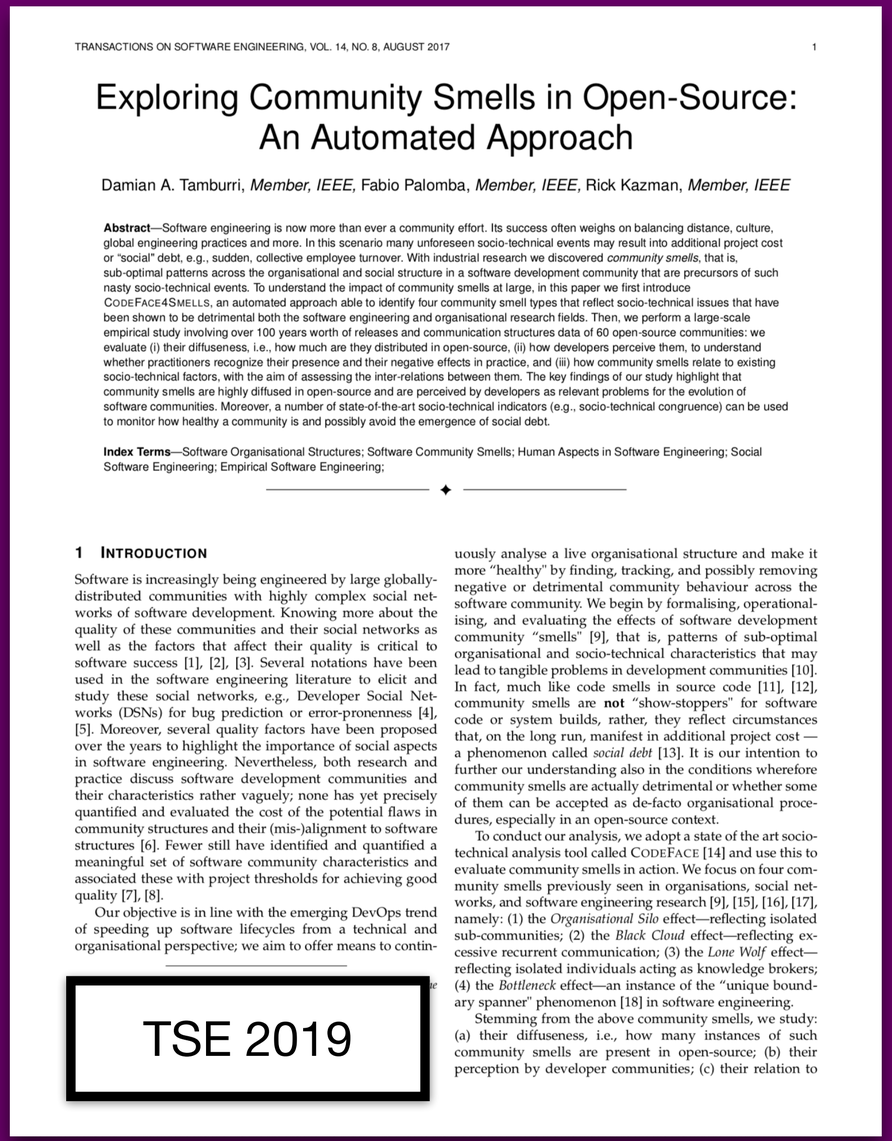
Community Smells

Some preliminary information

To sum up

Community smells have clear relations with **social** debt

Is the presence of community smells related to **technical** issues?



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

TSE 2019

Beyond Technical Aspects

Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?

Fabio Palomba, *Member, IEEE*, Damian A. Tamburri, *Member, IEEE*,
Francesca Arcelli Fontana, *Member, IEEE*, Rocco Oliveto, *Member, IEEE*,
Andy Zaidman, *Member, IEEE*, Alexander Serebrenik, *Senior Member, IEEE*.

Abstract—Code smells are poor implementation choices applied by developers during software evolution that often lead to critical flaws or failure. Much in the same way, community smells reflect the presence of organizational and socio-technical issues within a software community that may lead to additional project costs. Recent empirical studies provide evidence that community smells are often—if not always—connected to circumstances such as code smells. In this paper we look deeper into this connection by conducting a mixed-methods empirical study of 117 releases from 9 open-source systems. The qualitative and quantitative sides of our mixed-methods study were run in parallel and assume a mutually-confirmative connotation. On the one hand, we survey 162 developers of the 9 considered systems to investigate whether developers perceive relationship between community smells and the code smells found in those projects. On the other hand, we perform a fine-grained analysis into the 117 releases of our dataset to measure the extent to which community smells impact code smell intensity (i.e., criticality). We then propose a code smell intensity prediction model that relies on both technical and community-related aspects. The results of both sides of our mixed-methods study lead to one conclusion: community-related factors contribute to the intensity of code smells. This conclusion supports the joint use of community and code smells detection as a mechanism for the joint management of technical and social problems around software development communities.

Index Terms—Code smells, organizational structure, community smells, mixed-methods study

1 Introduction

Software engineering is, by nature, a "social" activity that involves organizations, developers, and stakeholders who are responsible for leading to the definition of a software product that meets the expected requirements [1]. The social interactions among the involved actors can represent the key to success but can also be a critical issue possibly causing additional project costs from an organizational and socio-technical perspective [1], [2].

In the recent past, the research community devoted effort to understanding so-called *social debt* [3], which refers to the presence of non-cohesive development communities whose members have communication or coordination issues that make them unable to tackle a certain development problem and that can lead to unforeseen project cost. One of the recent advances in this research field is represented by the definition of *community smells*, which were defined by Tamburri *et al.* [2], [4] as a set of socio-technical characteristics (e.g., high formality) and patterns (*e.g.,* repeated condescending behavior, or rage-quitting), which may lead to the emergence of social debt. From a more actionable and analytical perspective, community smells are nothing more than *motifs* over a graph [5]; motifs are recurrent and statistically significant sub-graphs or patterns over a graph detectable using either the structural properties and fashions of the graph or the graph salient features and characteristics (e.g., colors in the case of a colored graph). For example, the *organizational silo effect* [4] is a recurring network sub-structure featuring highly decoupled community structures.

In turn, community smells are often connected to circumstances such as technical debt [6], *i.e.,* the implementation of a poor implementation solution that will make the maintainability of the source code harder.

In this paper we aim at empirically exploring the relation between social and technical debt, by investigating the connection between two noticeable symptoms behind such types of debt: community and code smells. The latter refer to poor implementation decisions [7] that may lead to a decrease of maintainability [8] and an increase of the overall project costs [9].

> *We conjecture that the presence of community smells can influence the persistence of code smells, as the circumstances reflected by community smells (e.g., lack of communication or coordination between team members) may lead the code to be less maintainable, making code smells worse and worse over time.*

Our empirical investigation features a convergence mixed-methods approach [10], [11], [12] (see Figure 1) where quantitative and qualitative research are run in parallel over

• F. Palomba is with the University of Zurich, Switzerland. E-mail:

TSE 2019

Beyond Technical Aspects

Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?

Fabio Palomba, *Member, IEEE*, Damian A. Tamburri, *Member, IEEE*,
Francesca Arcelli Fontana, *Member, IEEE*, Rocco Oliveto, *Member, IEEE*,
Andy Zaidman, *Member, IEEE*, Alexander Serebrenik, *Senior Member, IEEE*.

Abstract—Code smells are poor implementation choices applied by developers during software evolution that often lead to critical flaws or failure. Much in the same way, community smells reflect the presence of organizational and socio-technical issues within a software community that may lead to additional project costs. Recent empirical studies provide evidence that community smells are often—if not always—connected to circumstances such as code smells. In this paper we look deeper into this connection by conducting a mixed-methods empirical study of 117 releases from 9 open-source systems. The qualitative and quantitative sides of our mixed-methods study were run in parallel and assume a mutually-confirmative connotation. On the one hand, we survey 162 developers of the 9 considered systems to investigate whether developers perceive relationship between community smells and the code smells found in those projects. On the other hand, we perform a fine-grained analysis into the 117 releases of our dataset to measure the extent to which community smells impact code smell intensity (i.e., criticality). We then propose a code smell intensity prediction model that relies on both technical and community-related aspects. The results of both sides of our mixed-methods study lead to one conclusion: community-related factors contribute to the intensity of code smells. This conclusion supports the joint use of community and code smells detection as a mechanism for the joint management of technical and social problems around software development communities.

Index Terms—Code smells, organizational structure, community smells, mixed-methods study

1 Introduction

Software engineering is, by nature, a "social" activity that involves organizations, developers, and stakeholders who are responsible for leading to the definition of a software product that meets the expected requirements [1]. The social interactions among the involved actors can represent the key to success but can also be a critical issue possibly causing additional project costs from an organizational and socio-technical perspective [1], [2].

In the recent past, the research community devoted effort to understanding so-called *social debt* [3], which refers to the presence of non-cohesive development communities whose members have communication or coordination issues that make them unable to tackle a certain development problem and that can lead to unforeseen project cost. One of the recent advances in this research field is represented by the definition of *community smells*, which were defined by Tamburri *et al.* [2], [4] as a set of socio-technical characteristics (e.g., high formality) and patterns (*e.g.,* repeated condescending behavior, or rage-quitting), which may lead to the emergence of social debt. From a more actionable and analytical perspective, community smells are nothing more than *motifs* over a graph [5]; motifs are recurrent and statistically significant sub-graphs or patterns over a graph detectable using either the structural properties and fashions of the graph or the graph salient features and characteristics (e.g., colors in the case of a colored graph). For example, the *organizational silo effect* [4] is a recurring network sub-structure featuring highly decoupled community structures.

In turn, community smells are often connected to circumstances such as technical debt [6], *i.e.,* the implementation of a poor implementation solution that will make the maintainability of the source code harder.

In this paper we aim at empirically exploring the relation between social and technical debt, by investigating the connection between two noticeable symptoms behind such types of debt: community and code smells. The latter refer to poor implementation decisions [7] that may lead to a decrease of maintainability [8] and an increase of the overall project costs [9].

> *We conjecture that the presence of community smells can influence the persistence of code smells, as the circumstances reflected by community smells (e.g., lack of communication or coordination between team members) may lead the code to be less maintainable, making code smells worse and worse over time.*

Our empirical investigation features a convergence mixed-methods approach [10], [11], [12] (see Figure 1) where quantitative and qualitative research are run in parallel over

• F. Palomba is with the University of Zurich, Switzerland. E-mail:

TSE 2019

The most challenging part

How to effectively measure the extent to which social debt influences technical debt?

Beyond Technical Aspects

Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?

Fabio Palomba, *Member, IEEE*, Damian A. Tamburri, *Member, IEEE*,
Francesca Arcelli Fontana, *Member, IEEE*, Rocco Oliveto, *Member, IEEE*,
Andy Zaidman, *Member, IEEE*, Alexander Serebrenik, *Senior Member, IEEE*.

Abstract—Code smells are poor implementation choices applied by developers during software evolution that often lead to critical flaws or failure. Much in the same way, community smells reflect the presence of organizational and socio-technical issues within a software community that may lead to additional project costs. Recent empirical studies provide evidence that community smells are often—if not always—connected to circumstances such as code smells. In this paper we look deeper into this connection by conducting a mixed-methods empirical study of 117 releases from 9 open-source systems. The qualitative and quantitative sides of our mixed-methods study were run in parallel and assume a mutually-confirmative connotation. On the one hand, we survey 162 developers of the 9 considered systems to investigate whether developers perceive relationship between community smells and the code smells found in those projects. On the other hand, we perform a fine-grained analysis into the 117 releases of our dataset to measure the extent to which community smells impact code smell intensity (i.e., criticality). We then propose a code smell intensity prediction model that relies on both technical and community-related aspects. The results of both sides of our mixed-methods study lead to one conclusion: community-related factors contribute to the intensity of code smells. This conclusion supports the joint use of community and code smells detection as a mechanism for the joint management of technical and social problems around software development communities.

Index Terms—Code smells, organizational structure, community smells, mixed-methods study

1 Introduction

Software engineering is, by nature, a "social" activity that involves organizations, developers, and stakeholders who are responsible for leading to the definition of a software product that meets the expected requirements [1]. The social interactions among the involved actors can represent the key to success but can also be a critical issue possibly causing additional project costs from an organizational and socio-technical perspective [1], [2].

In the recent past, the research community devoted effort to understanding so-called *social debt* [3], which refers to the presence of non-cohesive development communities whose members have communication or coordination issues that make them unable to tackle a certain development problem and that can lead to unforeseen project cost. One of the recent advances in this research field is represented by the definition of *community smells*, which were defined by Tamburri *et al.* [2], [4] as a set of socio-technical characteristics (e.g., high formality) and patterns (*e.g.,* repeated condescending behavior, or rage-quitting), which may lead to the emergence of social debt. From a more actionable and analytical perspective, community smells are nothing more than *motifs* over a graph [5]; motifs are recurrent and statistically significant sub-graphs or patterns over a graph detectable using either the structural properties and fashions of the graph or the graph salient features and characteristics (e.g., colors in the case of a colored graph). For example, the *organizational silo effect* [4] is a recurring network sub-structure featuring highly decoupled community structures.

In turn, community smells are often connected to circumstances such as technical debt [6], *i.e.,* the implementation of a poor implementation solution that will make the maintainability of the source code harder.

In this paper we aim at empirically exploring the relation between social and technical debt, by investigating the connection between two noticeable symptoms behind such types of debt: community and code smells. The latter refer to poor implementation decisions [7] that may lead to a decrease of maintainability [8] and an increase of the overall project costs [9].

> *We conjecture that the presence of community smells can influence the persistence of code smells, as the circumstances reflected by community smells (e.g., lack of communication or coordination between team members) may lead the code to be less maintainable, making code smells worse and worse over time.*

Our empirical investigation features a convergence mixed-methods approach [10], [11], [12] (see Figure 1) where quantitative and qualitative research are run in parallel over

• F. Palomba is with the University of Zurich, Switzerland. E-mail:

TSE 2019

The most challenging part

How to effectively measure the extent to which social debt influences technical debt?

Code smells

Symptoms of the presence of sub-optimal design/implementation choices in source code

Beyond Technical Aspects

Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?

Fabio Palomba, *Member, IEEE*, Damian A. Tamburri, *Member, IEEE*,
Francesca Arcelli Fontana, *Member, IEEE*, Rocco Oliveto, *Member, IEEE*,
Andy Zaidman, *Member, IEEE*, Alexander Serebrenik, *Senior Member, IEEE*.

Abstract—Code smells are poor implementation choices applied by developers during software evolution that often lead to critical flaws or failure. Much in the same way, community smells reflect the presence of organizational and socio-technical issues within a software community that may lead to additional project costs. Recent empirical studies provide evidence that community smells are often—if not always—connected to circumstances such as code smells. In this paper we look deeper into this connection by conducting a mixed-methods empirical study of 117 releases from 9 open-source systems. The qualitative and quantitative sides of our mixed-methods study were run in parallel and assume a mutually-confirmative connotation. On the one hand, we survey 162 developers of the 9 considered systems to investigate whether developers perceive relationship between community smells and the code smells found in those projects. On the other hand, we perform a fine-grained analysis into the 117 releases of our dataset to measure the extent to which community smells impact code smell intensity (i.e., criticality). We then propose a code smell intensity prediction model that relies on both technical and community-related aspects. The results of both sides of our mixed-methods study lead to one conclusion: community-related factors contribute to the intensity of code smells. This conclusion supports the joint use of community and code smells detection as a mechanism for the joint management of technical and social problems around software development communities.

Index Terms—Code smells, organizational structure, community smells, mixed-methods study

1 Introduction

Software engineering is, by nature, a "social" activity that involves organizations, developers, and stakeholders who are responsible for leading to the definition of a software product that meets the expected requirements [1]. The social interactions among the involved actors can represent the key to success but can also be a critical issue possibly causing additional project costs from an organizational and socio-technical perspective [1], [2].

In the recent past, the research community devoted effort to understanding so-called *social debt* [3], which refers to the presence of non-cohesive development communities whose members have communication or coordination issues that make them unable to tackle a certain development problem and that can lead to unforeseen project cost. One of the recent advances in this research field is represented by the definition of *community smells*, which were defined by Tamburri *et al.* [2], [4] as a set of socio-technical characteristics (e.g., high formality) and patterns (*e.g.,* repeated condescending behavior, or rage-quitting), which may lead to the emergence of social debt. From a more actionable and analytical perspective, community smells are nothing more than *motifs* over a graph [5]; motifs are recurrent and statistically significant sub-graphs or patterns over a graph detectable using either the structural properties and fashions of the graph or the graph salient features and characteristics (e.g., colors in the case of a colored graph). For example, the *organizational silo effect* [4] is a recurring network sub-structure featuring highly decoupled community structures.

In turn, community smells are often connected to circumstances such as technical debt [6], *i.e.,* the implementation of a poor implementation solution that will make the maintainability of the source code harder.

In this paper we aim at empirically exploring the relation between social and technical debt, by investigating the connection between two noticeable symptoms behind such types of debt: community and code smells. The latter refer to poor implementation decisions [7] that may lead to a decrease of maintainability [8] and an increase of the overall project costs [9].

> *We conjecture that the presence of community smells can influence the persistence of code smells, as the circumstances reflected by community smells (e.g., lack of communication or coordination between team members) may lead the code to be less maintainable, making code smells worse and worse over time.*

Our empirical investigation features a convergence mixed-methods approach [10], [11], [12] (see Figure 1) where quantitative and qualitative research are run in parallel over

• F. Palomba is with the University of Zurich, Switzerland. E-mail:

TSE 2019

The most challenging part

How to effectively measure the extent to which social debt influences technical debt?

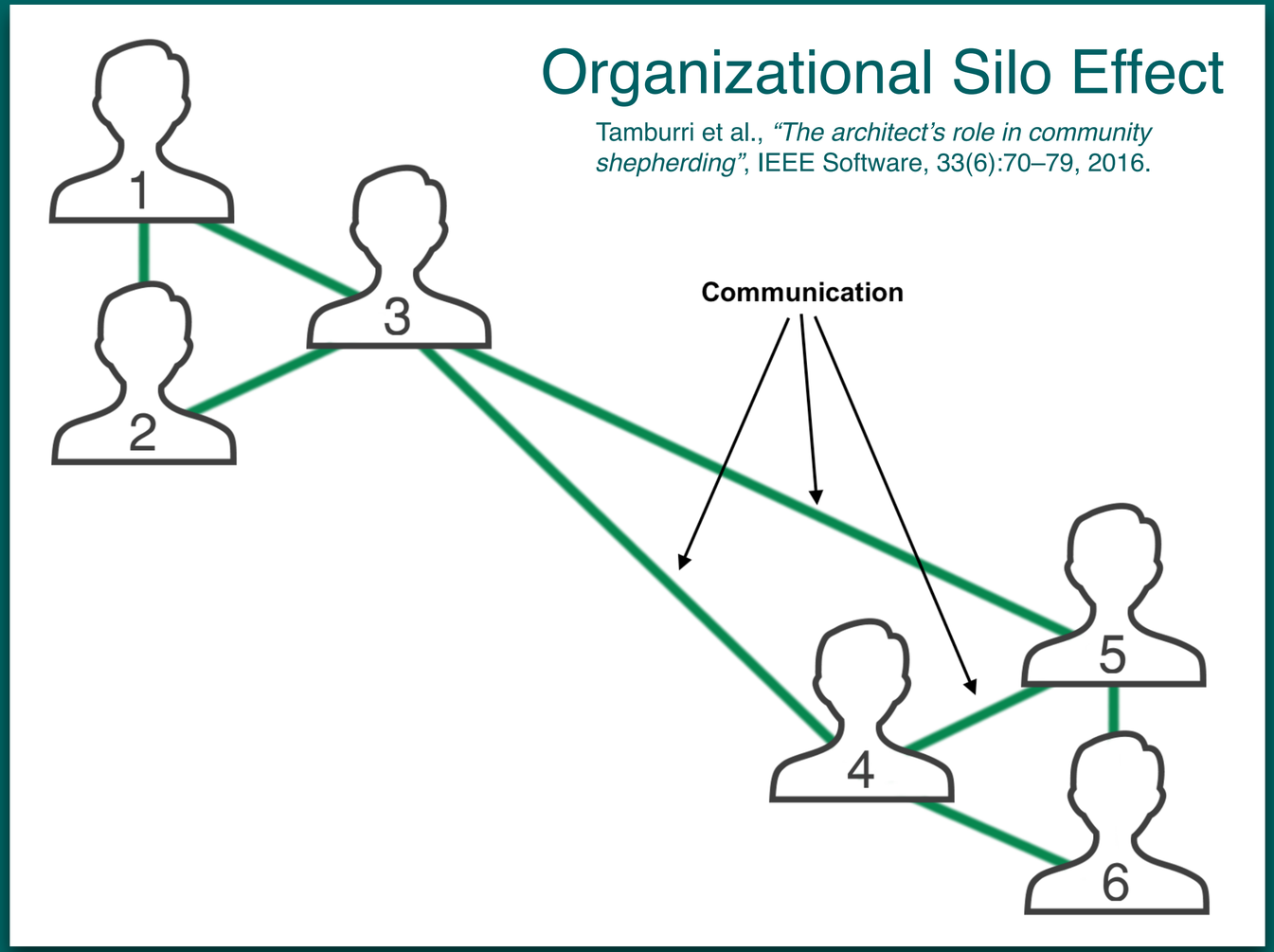
Code smells

Symptoms of the presence of sub-optimal design/implementation choices in source code

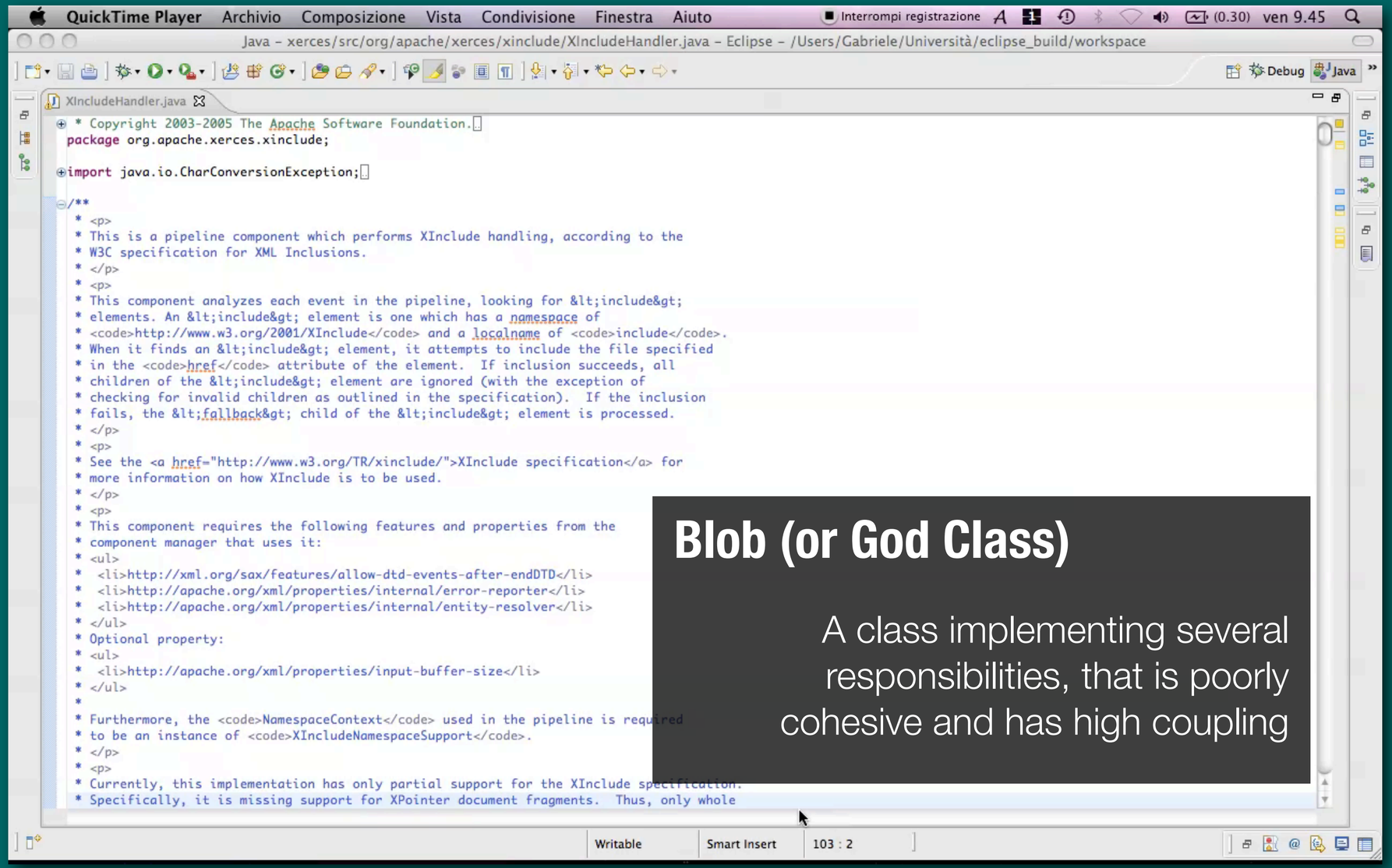
Mixed-methods research

Complementing mining software repository analyses with qualitative investigations to establish causality

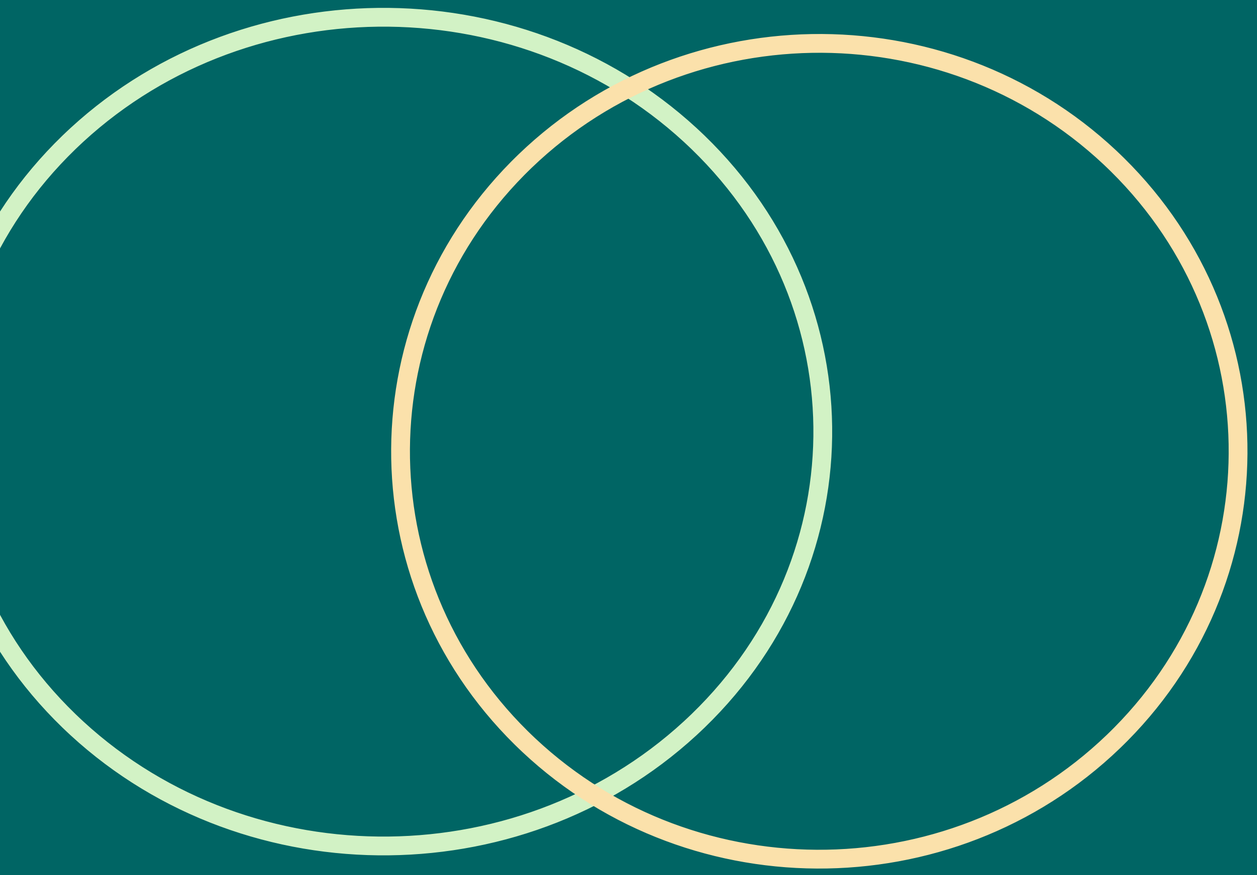
Community versus Code Smells



Community versus Code Smells



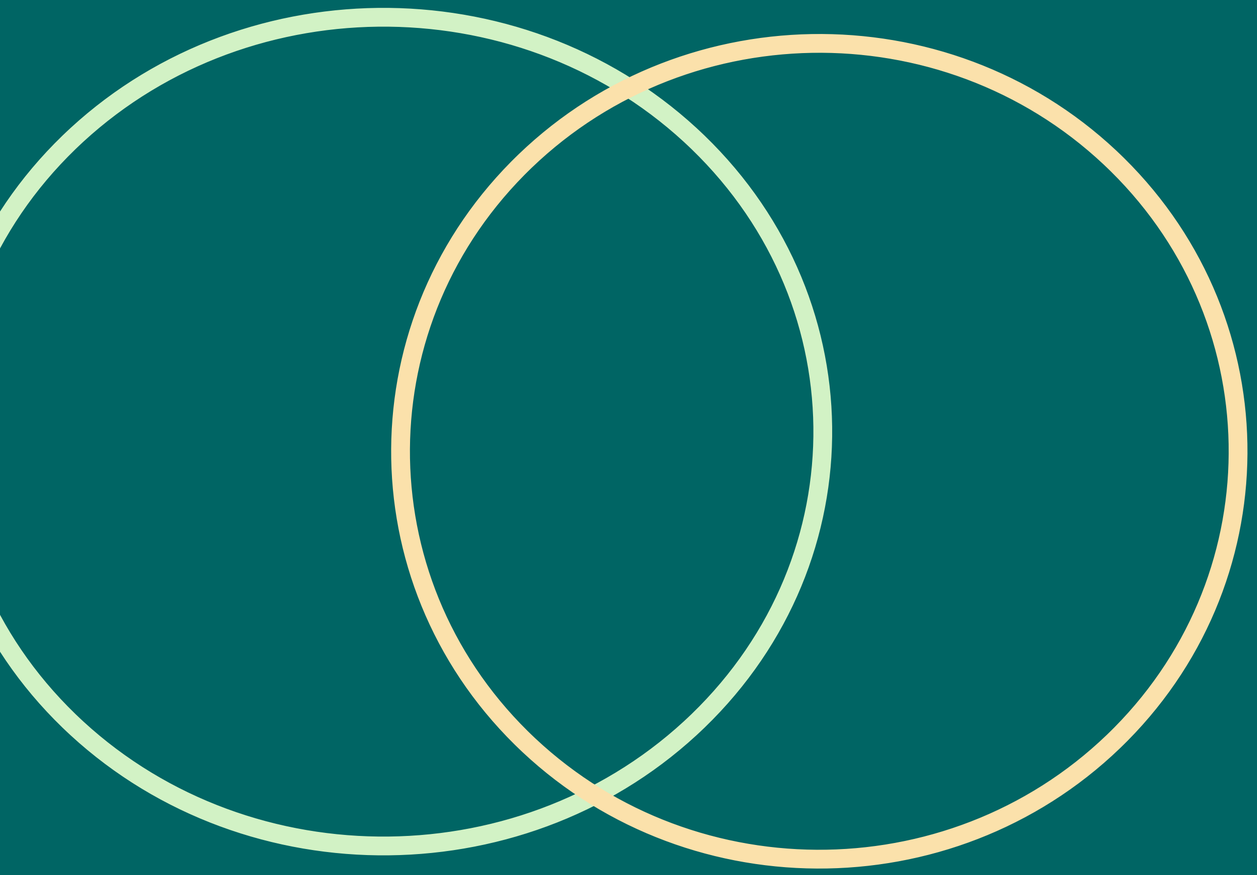
Community versus Code Smells



Conjecture

Sub-communities that do not communicate with each other might be not able to come up with a correct way to modularize the different modules of the systems.

Community versus Code Smells



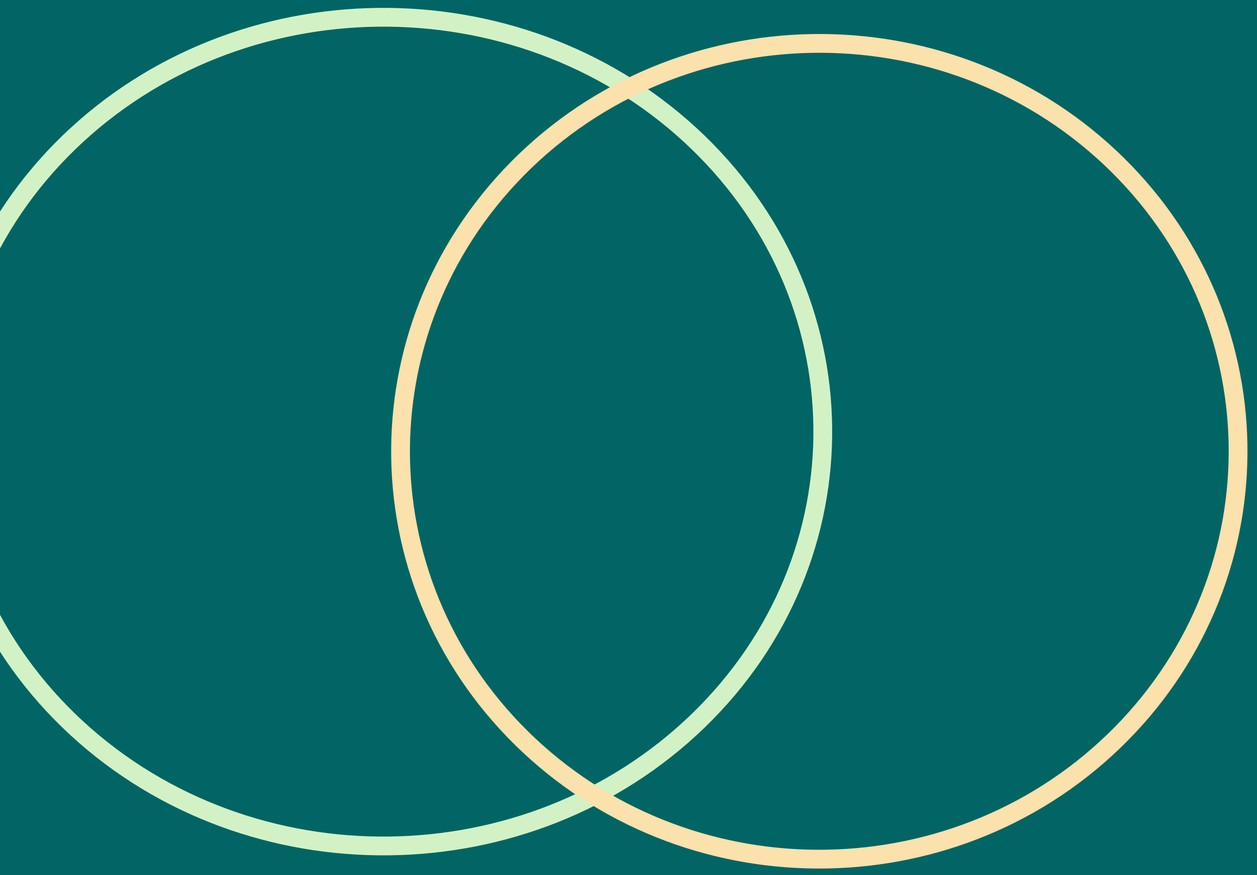
Conjecture

Sub-communities that do not communicate with each other might be not able to come up with a correct way to modularize the different modules of the systems.

Expected result

Thus, they might introduce **architectural** or **code** smells

Community versus Code Smells



Conjecture

Sub-communities that do not communicate with each other might be not able to come up with a correct way to modularize the different modules of the systems.

Expected result

Thus, they might introduce **architectural** or **code** smells

Experimental design

Two simultaneous studies: a survey and an MSR-like analyses

Community versus Code Smells

Survey Study

RQ1: What concerns affect the developers' decision to eliminate o preserve code smells?

Community versus Code Smells

Survey Study

RQ1: What concerns affect the developers' decision to eliminate o preserve code smells?

We e-mailed developers
as soon as they touched
a smelly class

1) Were you aware of this code smell?
2) Can you think of any technical root causes for the smell?
3) What are the reasons or risks that lead you to decide whether
or not to refactor the smell?

Community versus Code Smells

Survey Study

RQ1: What concerns affect the developers' decision to eliminate o preserve code smells?

We e-mailed developers as soon as they touched a smelly class

1) Were you aware of this code smell?
2) Can you think of any technical root causes for the smell?
3) What are the reasons or risks that lead you to decide whether or not to refactor the smell?

70%

of the cases, the decision not to refactor was due to a potential community smell

Community versus Code Smells

Survey Study

RQ1: What concerns affect the developers' decision to eliminate o preserve code smells?

We e-mailed developers as soon as they touched a smelly class

1) Were you aware of this code smell?
2) Can you think of any technical root causes for the smell?
3) What are the reasons or risks that lead you to decide whether or not to refactor the smell?

70%

of the cases, the decision not to refactor was due to a potential community smell

"If the algorithm implemented in the method would be split, then the ***developers working on that code would become crazy*** *since they are able to work pretty well on the existing code."*

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

RQ3: What is the performance of a community-aware code smell intensity prediction models?

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

RQ3: What is the performance of a community-aware code smell intensity prediction models?

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm 

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model 

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm 

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model 

Validation based on performance indicators (e.g., AUC-ROC)

User-based validation of the results

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

Technical metrics

LOC

CBO

commits

Code Churns

Project Tenure

Code Smell Persistence

Commit Tenure

Fault-proneness

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

Technical metrics

Socio-technical metrics

Socio-technical congruence

Truck-factor

Turnover

Core-Periphery Ratio

Smelly Quitters

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm 

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

Technical metrics

Socio-technical metrics

Dependent variable

Five code smell types, i.e., Long Method, Feature Envy, Blob, Spaghetti Code, and Misplaced Class

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

Technical metrics

Socio-technical metrics

Dependent variable

Five code smell types, i.e., Long Method, Feature Envy, Blob, Spaghetti Code, and Misplaced Class

Variation of severity over subsequent releases of 9 systems

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

For all smells

community smells are statistically more significant than all other socio-technical metrics

Community versus Code Smells

Predicting code smell intensity using community smells

RQ2: To what extent can community smells explain the increase of code smell intensity?

Application of the information gain algorithm 

Selection of appropriate socio-technical metrics able to control for the variation of code smell intensity in source code

For all smells

community smells are statistically more significant than all other socio-technical metrics

For all smells

community smells are statistically more significant than some technical factors, even if some code and process metrics are better predictors

Community versus Code Smells

Predicting code smell intensity using community smells

Three baselines

Only Technical

Technical + Community smells

Technical + Socio-Technical + Community smells

RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model

Validation based on performance indicators (e.g., AUC-ROC)

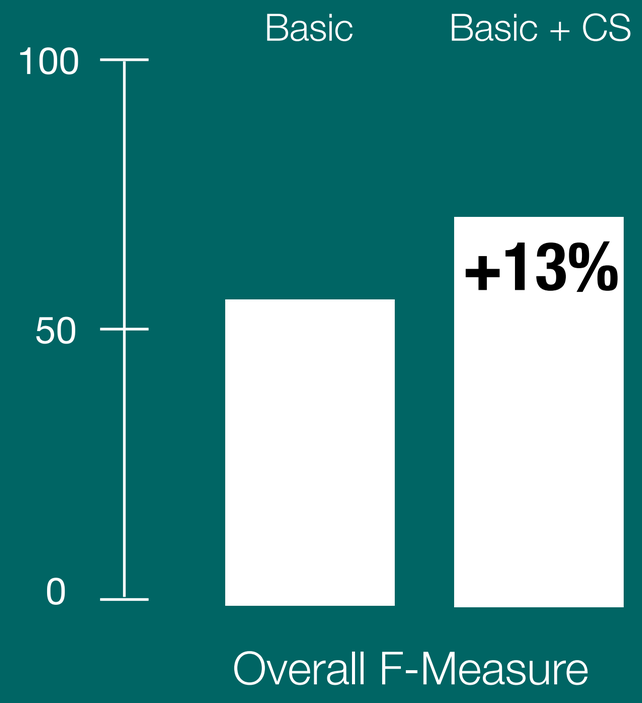
User-based validation of the results

Community versus Code Smells

Predicting code smell intensity using community smells

Three baselines

Results



RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model

Validation based on performance indicators (e.g., AUC-ROC)

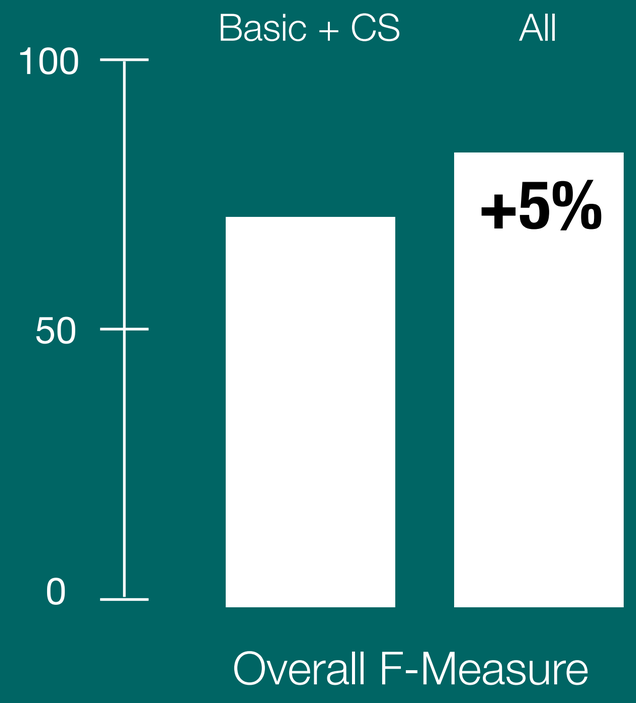
User-based validation of the results

Community versus Code Smells

Predicting code smell intensity using community smells

Three baselines

Results



RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model

Validation based on performance indicators (e.g., AUC-ROC)

User-based validation of the results

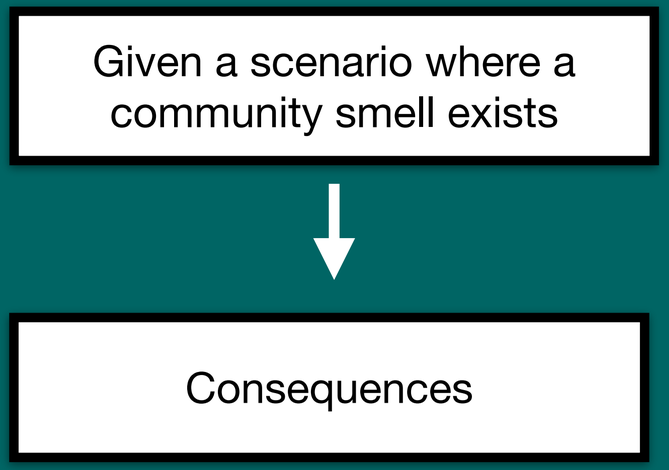
Community versus Code Smells

Predicting code smell intensity using community smells

Three baselines

Results

Ten project managers were asked to reason about possible scenarios occurring in a real context



RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model

Validation based on performance indicators (e.g., AUC-ROC)

User-based validation of the results

Community versus Code Smells

Predicting code smell intensity using community smells

Three baselines

Results

Ten project managers were asked to reason about possible scenarios occurring in a real context

*"The creation of **extremely complex and poorly cohesive classes in presence of non-communicating sub-teams is quite common** because in such a scenario developers do not share information with each other about the ideal structure to implement, therefore creating classes that perform a lot of different things".*

RQ3: What is the performance of a community-aware code smell intensity prediction models?

Definition of a community-aware prediction model

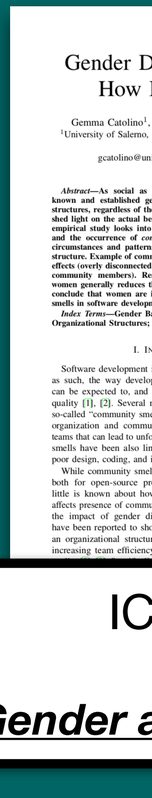
Validation based on performance indicators (e.g., AUC-ROC)

User-based validation of the results

Community versus Code Smells

Conclusion and Open Issues

Community smells can impact source code quality


Gender D
How
Gemma Catolino[1],
[1]University of Salerno,
gcatolino@un
Abstract—As social as
known and established ge
structures, regardless of th
shed light on the actual be
empirical study looks into
and the occurrence of *co*
circumstances and pattern
structure. Example of comm
effects (overly disconnected
community members). Re
women generally reduces t
conclude that women are
smells in software develop
Index Terms—Gender Ba
Organizational Structures;
I. In
Software development
as such, the way develo
can be expected to, and
quality [1], [2]. Several
so-called "community sme
organization and commu
teams that can lead to unfo
smells have been also lin
poor design, coding, and
While community smel
both for open-source pr
little is known about ho
affects presence of commu
the impact of gender di
have been reported to she
an organizational structur
increasing team efficiency

IC

Gender a

Community versus Code Smells

Conclusion and Open Issues

Community smells can impact source code quality

How to deal with community smells?

Gender Diversity and Women in Software Teams: How Do They Affect Community Smells?

Gemma Catolino[1], Fabio Palomba[2], Damian A. Tamburri[3, 4], Alexander Serebrenik[4], Filomena Ferrucci[1]
[1]University of Salerno, Italy, [2]University of Zurich, Switzerland, [3]Jeronimus Academy of Data Science, The Netherlands,
[4]Eindhoven University of Technology, The Netherlands
gcatolino@unisa.it, palomba@ifi.uzh.ch, d.a.tamburri@tue.nl, a.serebrenik@tue.nl, fferrucci@unisa.it

***Abstract*—As social as software engineers are, there is a known and established gender imbalance in our community structures, regardless of their open- or closed-source nature. To shed light on the actual benefits of achieving such balance, this empirical study looks into the relations between such balance and the occurrence of *community smells*, that is, sub-optimal circumstances and patterns across the software organizational structure. Example of community smells are Organizational Silo effects (overly disconnected sub-groups) or Lone Wolves (defiant community members). Results indicate that the presence of women generally reduces the amount of community smells. We conclude that women are instrumental to reducing community smells in software development teams.**

***Index Terms*—Gender Balance; Community Smells; Software Organizational Structures; Empirical Study**

I. Introduction

Software development is an inherently social activity, and as such, the way developers communicate and collaborate can be expected to, and has been found to affect software quality [1], [2]. Several recent studies have focused on the so-called "community smells", patterns indicating suboptimal organization and communication of software development teams that can lead to unforeseen project costs [3]. Community smells have been also linked to code smells, indications of poor design, coding, and implementation choices [4], [5].

While community smells have been studied and described both for open-source projects and closed-source projects, little is known about how *composition* of a software team affects presence of community smells, and in particular, about the impact of gender diversity on them. Indeed, women have been reported to show the socio-emotional behavior in an organizational structure [6] and discourage conflict [7], increasing team efficiency [7] and mediating organizational

well as on how to indirectly improving also the quality of the product being developed [4], [5].

We seek therefore to verify our conjecture, and assess the role of gender diversity and women participation on community smells. More specifically, in this study we consider four of the community smells defined and operationalized by Tamburri et al. [3], [4], i.e., *Organizational Silo*, *Black Cloud*, *Lone Wolf*, and *Radio Silence*. All of them refer to suboptimal characteristics of a development community that might be mitigated by the presence of women.

As original and novel contribution of this study, we build a statistical model relating gender balance (as reflected by the well-known Blau-Index [12]) as well as the *number of women in a team* to the aforementioned community smells. Moreover, we control for a wide array of socio-technical and social-networks factors of influence (e.g., the well-known *socio-technical congruence* [13]), in order to verify whether the variables of interest remain statistically significant also in presence of such additional factors.

The results of our study show that the expected relations between gender balance and participation of women as mediators against the proliferation of community smells are valid in the cases of *Black Cloud* and *Radio Silence*, while we found only partial relations between the variables of interest and *Organizational Silo* and *Lone Wolf*. For example, as expected the role of women as mediators for increased organizational quality is not equal across all community smells. In particular, their role is increasingly important for those smells which affect the quantities and qualities of communication across the organizational structure (e.g., the *Radio Silence* effect) as opposed to the organizational arrangement of the structure (e.g., the *Organizational Silo* effect). These results indicate that the presence of women plays a beneficial role in *mitigating*

ICSE 2019 (SEIS)

Today at 14

Gender and Trust - Van-Horne

Community versus Code Smells

Conclusion and Open Issues

Community smells can impact source code quality

How to deal with community smells?

A better understanding of the impact of community smells on other technical aspects

Gender Diversity and Women in Software Teams: How Do They Affect Community Smells?

Gemma Catolino[1], Fabio Palomba[2], Damian A. Tamburri[3,4], Alexander Serebrenik[4], Filomena Ferrucci[1]
[1]University of Salerno, Italy, [2]University of Zurich, Switzerland, [3]Jeronimus Academy of Data Science, The Netherlands,
[4]Eindhoven University of Technology, The Netherlands
gcatolino@unisa.it, palomba@ifi.uzh.ch, d.a.tamburri@tue.nl, a.serebrenik@tue.nl, fferrucci@unisa.it

***Abstract*—As social as software engineers are, there is a known and established gender imbalance in our community structures, regardless of their open- or closed-source nature. To shed light on the actual benefits of achieving such balance, this empirical study looks into the relations between such balance and the occurrence of *community smells*, that is, sub-optimal circumstances and patterns across the software organizational structure. Example of community smells are Organizational Silo effects (overly disconnected sub-groups) or Lone Wolves (defiant community members). Results indicate that the presence of women generally reduces the amount of community smells. We conclude that women are instrumental to reducing community smells in software development teams.**

***Index Terms*—Gender Balance; Community Smells; Software Organizational Structures; Empirical Study**

I. Introduction

Software development is an inherently social activity, and as such, the way developers communicate and collaborate can be expected to, and has been found to affect software quality [1], [2]. Several recent studies have focused on the so-called "community smells", patterns indicating suboptimal organization and communication of software development teams that can lead to unforeseen project costs [3]. Community smells have been also linked to code smells, indications of poor design, coding, and implementation choices [4], [5].

While community smells have been studied and described both for open-source projects and closed-source projects, little is known about how *composition* of a software team affects presence of community smells, and in particular, about the impact of gender diversity on them. Indeed, women have been reported to show the socio-emotional behavior in an organizational structure [6] and discourage conflict [7], increasing team efficiency [7] and mediating organizational

well as on how to indirectly improving also the quality of the product being developed [4], [5].

We seek therefore to verify our conjecture, and assess the role of gender diversity and women participation on community smells. More specifically, in this study we consider four of the community smells defined and operationalized by Tamburri et al. [3], [4], i.e., *Organizational Silo*, *Black Cloud*, *Lone Wolf*, and *Radio Silence*. All of them refer to suboptimal characteristics of a development community that might be mitigated by the presence of women.

As original and novel contribution of this study, we build a statistical model relating gender balance (as reflected by the well-known Blau-Index [12]) as well as the *number of women in a team* to the aforementioned community smells. Moreover, we control for a wide array of socio-technical and social-networks factors of influence (e.g., the well-known *socio-technical congruence* [13]), in order to verify whether the variables of interest remain statistically significant also in presence of such additional factors.

The results of our study show that the expected relations between gender balance and participation of women as mediators against the proliferation of community smells are valid in the cases of *Black Cloud* and *Radio Silence*, while we found only partial relations between the variables of interest and *Organizational Silo* and *Lone Wolf*. For example, as expected the role of women as mediators for increased organizational quality is not equal across all community smells. In particular, their role is increasingly important for those smells which affect the quantities and qualities of communication across the organizational structure (e.g., the *Radio Silence* effect) as opposed to the organizational arrangement of the structure (e.g., the *Organizational Silo* effect). These results indicate that the presence of women plays a beneficial role in *mitigating*

ICSE 2019 (SEIS)

Today at 14

Gender and Trust - Van-Horne

Community versus Code Smells

Conclusion and Open Issues

Community smells can impact source code quality

How to deal with community smells?

A better understanding of the impact of community smells on other technical aspects

Gender Diversity and Women in Software Teams: How Do They Affect Community Smells?

Gemma Catolino[1], Fabio Palomba[2], Damian A. Tamburri[3,4], Alexander Serebrenik[4], Filomena Ferrucci[1]
[1]University of Salerno, Italy, [2]University of Zurich, Switzerland, [3]Jeronimus Academy of Data Science, The Netherlands,
[4]Eindhoven University of Technology, The Netherlands
gcatolino@unisa.it, palomba@ifi.uzh.ch, d.a.tamburri@tue.nl, a.serebrenik@tue.nl, fferrucci@unisa.it

***Abstract*—As social as software engineers are, there is a known and established gender imbalance in our community structures, regardless of their open- or closed-source nature. To shed light on the actual benefits of achieving such balance, this empirical study looks into the relations between such balance and the occurrence of *community smells*, that is, sub-optimal circumstances and patterns across the software organizational structure. Example of community smells are Organizational Silo effects (overly disconnected sub-groups) or Lone Wolves (defiant community members). Results indicate that the presence of women generally reduces the amount of community smells. We conclude that women are instrumental to reducing community smells in software development teams.**

***Index Terms*—Gender Balance; Community Smells; Software Organizational Structures; Empirical Study**

I. Introduction

Software development is an inherently social activity, and as such, the way developers communicate and collaborate can be expected to, and has been found to affect software quality [1], [2]. Several recent studies have focused on the so-called "community smells", patterns indicating suboptimal organization and communication of software development teams that can lead to unforeseen project costs [3]. Community smells have been also linked to code smells, indications of poor design, coding, and implementation choices [4], [5].

While community smells have been studied and described both for open-source projects and closed-source projects, little is known about how *composition* of a software team affects presence of community smells, and in particular, about the impact of gender diversity on them. Indeed, women have been reported to show the socio-emotional behavior in an organizational structure [6] and discourage conflict [7], increasing team efficiency [7] and mediating organizational

well as on how to indirectly improving also the quality of the product being developed [4], [5].

We seek therefore to verify our conjecture, and assess the role of gender diversity and women participation on community smells. More specifically, in this study we consider four of the community smells defined and operationalized by Tamburri et al. [3], [4], i.e., *Organizational Silo*, *Black Cloud*, *Lone Wolf*, and *Radio Silence*. All of them refer to suboptimal characteristics of a development community that might be mitigated by the presence of women.

As original and novel contribution of this study, we build a statistical model relating gender balance (as reflected by the well-known Blau-Index [12]) as well as the *number of women in a team* to the aforementioned community smells. Moreover, we control for a wide array of socio-technical and social-networks factors of influence (e.g., the well-known *socio-technical congruence* [13]), in order to verify whether the variables of interest remain statistically significant also in presence of such additional factors.

The results of our study show that the expected relations between gender balance and participation of women as mediators against the proliferation of community smells are valid in the cases of *Black Cloud* and *Radio Silence*, while we found only partial relations between the variables of interest and *Organizational Silo* and *Lone Wolf*. For example, as expected the role of women as mediators for increased organizational quality is not equal across all community smells. In particular, their role is increasingly important for those smells which affect the quantities and qualities of communication across the organizational structure (e.g., the *Radio Silence* effect) as opposed to the organizational arrangement of the structure (e.g., the *Organizational Silo* effect). These results indicate that the presence of women plays a beneficial role in *mitigating*

ICSE 2019 (SEIS)

Today at 14

Gender and Trust - Van-Horne

Fabio Palomba

fabiopalomba3

Predicting the Emergence of Community Smells using Socio-Technical Metrics: a ML Approach

Fabio Palomba,[1] Damian A. Tamburri[2]

[1]**Software Engineering Research Lab, University of Salerno, Italy**
[2]Jheronimus Academy of Data Science, The Netherlands

sesa lab
SOFTWARE ENGINEERING SALERNO

Software engineering is, by nature, a *social* activity; socio-technical concerns might impact various software development activities

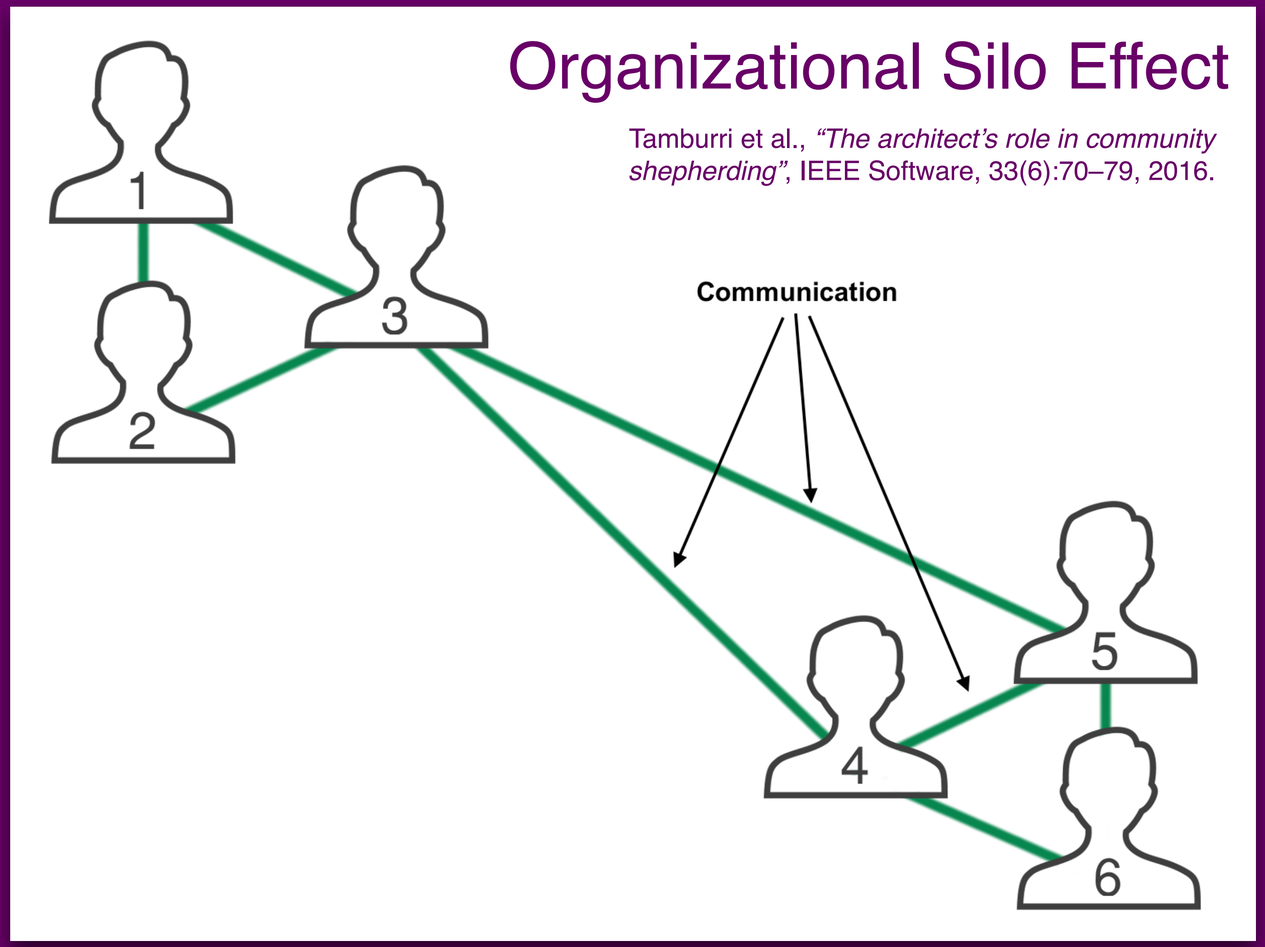
[Conway M. E. - *“How Do committees invent”* Datamation, 14, 4 (1968), 28–31]

Community Smells

A set of socio-technical characteristics and patterns, which may lead to the emergence of social debt

Community Smells

A set of socio-technical characteristics and patterns, which may lead to the emergence of social debt



Community Smells
Do They Really Smell Bad?

Community smells in OSS

Community smells **regularly** affect open-source communities

FOCUS: THE ROLE OF THE SOFTWARE ARCHITECT

The Architect's Role in Community Shepherding

Damian A. Tamburri, Politecnico di Milano

Rick Kazman, University of Hawaii

Hamed Fahimi, CGI

// Community shepherds guide development projects' social and organizational structures, look for early warning signs of trouble, and help mitigate that trouble. Knowing community smells—ways in which software communities might be "buggy"—can help with this. //

SOFTWARE ARCHITECTS DON'T just design architecture components or champion architecture qualities; they often must guide and harmonize as DevOps, open source strategies, transitions to agile development, and corporate acquisitions.

To help architects in their role erds, we pre- to tell when a ems and what roblems. We view of com-

IEEE-SW 2016

Interior Design for Software Architects

What do you do when you want to reorganize the furniture in your room? You take pencil and paper and draw a blueprint of the room and its elements (the desk and wardrobes, that "exotic" table your partner got who-knows-where, and so on), including the most important properties (size, shape, fragility, and so on). Now let's say your partner wants to keep that exotic table where it is without even removing the dreadful tapestry on top of it. What do you do? You get some counseling.

Well, the same story happens in software architecture practice.[1] Architects often must act as key community figures or counselors who bridge the gap between the development organization, including its subgroups, and the software architecture, including its properties and implementation details. Consequently, those software architectures (the blueprints in this case) are often augmented with views that express authority,[2] task divisions, elements, relations, and more. These characteristics reflect organizational and community types known to organizations and social-network researchers.[3,4] (A community type is a social network in which social or organizational characteristics are constantly evident.[3] For example, a project team is small [size], cross-functional [average cognitive distance], and time-bound [duration].)

Returning to our room: Counseling isn't going well, but you love your partner and decide to give in on the table—but you never forget. If this happens regularly, tension or even defiance might ensue.

Well, the same story happens in software architecture practice.[5] Architects have often reported trouble

Community Smells
Do They Really Smell Bad?

Community smells in OSS

Community smells **regularly** affect open-source communities

Community smells lead to the **reduction** of communicability, socio-technical congruence, and **increase** the turnover rate

Understanding Community Smells Variability:
A Statistical Approach

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

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

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

I. Introduction

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

The research on community smells has gained attention over the last years. Recent work has established the interaction between community smells and their technical counterparts [10] as well as, more generally, technical debt in its various forms [11], [12]. Moreover, researchers have been investigating how community smells appear [2], [13], what are the major factors that may influence community smells [13]–[15], and how developers refactor community smells in practice [16]. For instance, Tamburri *et al.* [13] showed that community smells

ICSE 2021

participation [14], [15], even though there exist specific actions that can be performed to modify the structure of a community and get rid of these smells [16].

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

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

To sum up, our paper offers three main contributions:

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

Community Smells

Do They Really Smell Bad?

Community smells in OSS

Community smells **regularly** affect open-source communities

Community smells lead to the **reduction** of communicability, socio-technical congruence, and **increase** the turnover rate

The emergence of community smells **induce** the emergence of technical problems, like code smells, other than **precluding** the application of refactoring activities

Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?

Fabio Palomba, *Member, IEEE*, Damian A. Tamburri, *Member, IEEE*,
Francesca Arcelli Fontana, *Member, IEEE*, Rocco Oliveto, *Member, IEEE*,
Andy Zaidman, *Member, IEEE*, Alexander Serebrenik, *Senior Member, IEEE*.

Abstract—Code smells are poor implementation choices applied by developers during software evolution that often lead to critical flaws or failure. Much in the same way, community smells reflect the presence of organizational and socio-technical issues within a software community that may lead to additional project costs. Recent empirical studies provide evidence that community smells are often—if not always—connected to circumstances such as code smells. In this paper we look deeper into this connection by conducting a mixed-methods empirical study of 117 releases from 9 open-source systems. The qualitative and quantitative sides of our mixed-methods study were run in parallel and assume a mutually-confirmative connotation. On the one hand, we survey 162 developers of the 9 considered systems to investigate whether developers perceive relationship between community smells and the code smells found in those projects. On the other hand, we perform a fine-grained analysis into the 117 releases of our dataset to measure the extent to which community smells impact code smell intensity (i.e., criticality). We then propose a code smell intensity prediction model that relies on both technical and community-related aspects. The results of both sides of our mixed-methods study lead to one conclusion: community-related factors contribute to the intensity of code smells. This conclusion supports the joint use of community and code smells detection as a mechanism for the joint management of technical and social problems around software development communities.

Index Terms—Code smells, organizational structure, community smells, mixed-methods study

1 Introduction

Software engineering is, by nature, a "social" activity that involves organizations, developers, and stakeholders who are responsible for leading to the definition of a software product that meets the expected requirements [1]. The social interactions among the involved actors can represent the key to success but can also be a critical issue possibly causing additional project costs from an organizational and socio-technical perspective [1], [2].

In the recent past, the research community devoted effort to understanding so-called *social debt* [3], which refers to the presence of non-cohesive development communities whose members have communication or coordination issues that make them unable to tackle a certain development problem and that can lead to unforeseen project cost. One of the recent advances in this research field is represented by the definition of *community smells*, which were defined by Tamburri *et al.* [2], [4] as a set of socio-technical characteristics (e.g., high formality) and patterns (*e.g.,* repeated condescending behavior, or rage-quitting), which may lead to the emergence of social debt. From a more actionable and analytical perspective, community smells are nothing more than *motifs* over a graph [5]; motifs are recurrent and statistically significant sub-graphs or patterns over a graph detectable using either the structural properties and fashions of the graph or the graph salient features and characteristics (e.g., colors in the case of a colored graph). For example, the *organizational silo effect* [4] is a recurring network sub-structure featuring highly decoupled community structures.

In turn, community smells are often connected to circumstances such as technical debt [6], *i.e.,* the implementation of a poor implementation solution that will make the maintainability of the source code harder.

In this paper we aim at empirically exploring the relation between social and technical debt, by investigating the connection between two noticeable symptoms behind such types of debt: community and code smells. The latter refer to poor implementation decisions [7] that may lead to a decrease of maintainability [8] and an increase of the overall project costs [9].

> *We conjecture that the presence of community smells can influence the persistence of code smells, as the circumstances reflected by community smells (e.g., lack of communication or coordination between team members) may lead the code to be less maintainable, making code smells worse and worse over time.*

Our empirical investigation features a convergence mixed-methods approach [10], [11], [12] (see Figure 1) where quantitative and qualitative research are run in parallel over

- *F. Palomba is with the University of Zurich, Switzerland; E-mail: palomba@ifi.uzh.ch*
- *D. A. Tamburri and A. Serebrenik are with Eindhoven University of*

TSE 2018

Community Smells
Detecting Them and Beyond

Dealing with Community Smells in Practice

Researchers have first proposed automated detection approaches

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

TSE 2019

Community Smells
Detecting Them and Beyond

Dealing with Community Smells in Practice

Researchers have first proposed automated detection approaches

More recently, AI has been shown as a promising alternative

2020 ACM/IEEE 15th International Conference on Global Software Engineering (ICGSE)

On the Detection of Community Smells Using Genetic Programming-based Ensemble Classifier Chain

Nuri Almarimi
ETS Montreal, University of Quebec
Montreal, QC, Canada
nuri.almarimi.1@ens.etsmtl.ca

Ali Ouni
ETS Montreal, University of Quebec
Montreal, QC, Canada
ali.ouni@etsmtl.ca

Moataz Chouchen
ETS Montreal, University of Quebec
Montreal, QC, Canada
moataz.chouchen.1@ens.etsmtl.ca

Islem Saidani
ETS Montreal, University of Quebec
Montreal, QC, Canada
islem.saidani.1@ens.etsmtl.ca

Mohamed Wiem Mkaouer
Rochester Institute of Technology
Rochester, NY, USA
mwmvse@rit.edu

ABSTRACT

Community smells are symptoms of organizational and social issues within the software development community that often increase the project costs and impact software quality. Recent studies have identified a variety of community smells and defined them as sub-optimal patterns connected to organizational-social structures in the software development community such as the lack of communication, coordination and collaboration. Recognizing the advantages of the early detection of potential community smells in a software project, we introduce a novel approach that learns from various community organizational and social practices to provide an automated support for detecting community smells. In particular, our approach learns from a set of interleaving organizational-social symptoms that characterize the existence of community smell instances in a software project. We build a multi-label learning model to detect 8 common types of community smells. We use the ensemble classifier chain (ECC) model that transforms multi-label problems into several single-label problems which are solved using genetic programming (GP) to find the optimal detection rules for each smell type. To evaluate the performance of our approach, we conducted an empirical study on a benchmark of 103 open source projects and 407 community smell instances. The statistical tests of our results show that our approach can detect the eight considered smell types with an average F-measure of 89% achieving a better performance compared to different state-of-the-art techniques. Furthermore, we found that the most influential factors that best characterize community smells include the social network density and closeness centrality as well as the standard deviation of the number of developers per time zone and per community.

CCS CONCEPTS

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

KEYWORDS

Community smells, Social debt, Socio-technical factors, Multi-label learning, Genetic programming, Search-based software engineering

ACM Reference Format:
Nuri Almarimi, Ali Ouni, Moataz Chouchen, Islem Saidani, and Mohamed Wiem Mkaouer. 2020. On the Detection of Community Smells Using Genetic Programming-based Ensemble Classifier Chain. In *15th IEEE/ACM International Conference on Global Software Engineering (ICGSE '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3372787.3390439

ICGSE 2020

1 INTRODUCTION

Modern software engineering is increasingly dependent on the well-being of large globally distributed communities and their social networks in software development. Knowing more about the organizational structures of these communities and their social characteristics as well as the factors that affect their quality is critical to software projects success [17, 33, 49].

Recent studies explored a set of socio-technical patterns that can negatively impact the organizational health of software projects and coined them as *community smells* [52, 54]. Community smells are connected to circumstances based on poor organizational and social practices that lead to the occurrence of social debt [51, 54]. Social debt is connected to negative organized structures that often lead to short and/or long term social issues within a project. These problems could manifest in several forms, *e.g.*, lack of communications, collaboration or coordination among members in a software development community. For example, one of the common community smells, is the "*organizational silo effect*" (OSE) [45, 56] which manifests as a recurring social network sub-structure featuring highly decoupled developer's community structure. From an analytical perspective, the OSE smell can be interpreted as a set of patterns over a social network graph and could be detectable using different graph connectivity characteristics.

Detecting community smells is still, to some extent, a difficult, time consuming, and manual process. Indeed, there is no consensual way to translate formal definition and symptoms into actionable detection rules. Typically, the number of potential bad organizational practices often exceeds the resources available to address them. In many cases, mature software projects are forced to be developed with both known and unknown poor socio-technical community

Community Smells
Detecting Them and Beyond

Dealing with Community Smells in Practice

Researchers have first proposed automated detection approaches

More recently, AI has been shown as a promising alternative

Researchers have also investigated how to mitigate their effects

Gender Diversity and Women in Software Teams: How Do They Affect Community Smells?

Gemma Catolino[1], Fabio Palomba[2], Damian A. Tamburri[3,4], Alexander Serebrenik[4], Filomena Ferrucci[1]
[1]University of Salerno, Italy, [2]University of Zurich, Switzerland, [3]Jeronimus Academy of Data Science, The Netherlands,
[4]Eindhoven University of Technology, The Netherlands
gcatolino@unisa.it, palomba@ifi.uzh.ch, d.a.tamburri@tue.nl, a.serebrenik@tue.nl, fferrucci@unisa.it

***Abstract*—As social as software engineers are, there is a known and established gender imbalance in our community structures, regardless of their open- or closed-source nature. To shed light on the actual benefits of achieving such balance, this empirical study looks into the relations between such balance and the occurrence of *community smells*, that is, sub-optimal circumstances and patterns across the software organizational structure. Example of community smells are Organizational Silo effects (overly disconnected sub-groups) or Lone Wolves (defiant community members). Results indicate that the presence of women generally reduces the amount of community smells. We conclude that women are instrumental to reducing community smells in software development teams.**

***Index Terms*—Gender Balance; Community Smells; Software Organizational Structures; Empirical Study**

I. Introduction

Software development is an inherently social activity, and as such, the way developers communicate and collaborate can be expected to, and has been found to affect software quality [1], [2]. Several recent studies have focused on the so-called "community smells", patterns indicating suboptimal organization and communication of software development teams that can lead to unforeseen project costs [3]. Community smells have been also linked to code smells, indications of poor design, coding, and implementation choices [4], [5].

While community smells have been studied and described both for open-source projects and closed-source projects, little is known about how *composition* of a software team affects presence of community smells, and in particular, about the impact of gender diversity on them. Indeed, women have been reported to show the socio-emotional behavior in an organizational structure [6] and discourage conflict [7], increasing team efficiency [7] and mediating organizational quality [8], [9]. Specifically in the software engineering context, communication is a crucial factor in project success as reported by architects of a feminine expertise [10], while more gender-diverse teams have been shown to be more productive than less gender-diverse ones [11].

Based on such previous findings, in this paper we conjec-

ICSE 2019

well as on how to indirectly improving also the quality of the product being developed [4], [5].

We seek therefore to verify our conjecture, and assess the role of gender diversity and women participation on community smells. More specifically, in this study we consider four of the community smells defined and operationalized by Tamburri et al. [3], [4], i.e., *Organizational Silo*, *Black Cloud*, *Lone Wolf*, and *Radio Silence*. All of them refer to suboptimal characteristics of a development community that might be mitigated by the presence of women.

As original and novel contribution of this study, we build a statistical model relating gender balance (as reflected by the well-known Blau-Index [12]) as well as the *number of women in a team* to the aforementioned community smells. Moreover, we control for a wide array of socio-technical and social-networks factors of influence (e.g., the well-known *socio-technical congruence* [13]), in order to verify whether the variables of interest remain statistically significant also in presence of such additional factors.

The results of our study show that the expected relations between gender balance and participation of women as mediators against the proliferation of community smells are valid in the cases of *Black Cloud* and *Radio Silence*, while we found only partial relations between the variables of interest and *Organizational Silo* and *Lone Wolf*. For example, as expected the role of women as mediators for increased organizational quality is not equal across all community smells. In particular, their role is increasingly important for those smells which affect the quantities and qualities of communication across the organizational structure (e.g., the *Radio Silence* effect) as opposed to the organizational arrangement of the structure (e.g., the *Organizational Silo* effect). These results indicate that the presence of women plays a beneficial role in *mitigating* strains faced by complex organizations. Finally, the results encourage further the study into the empirical relation between gender balance and the complex relations constituting software engineering organizational structures.

Reproducibility. In order to enable the *full* replication of our study, we provide a comprehensive package containing all data and scripts used as additional contribution [14].

Structure of the paper. In Section II we discuss the research methodology adopted to build the statistical model while in Section III we report the results of the study. Section IV

Community Smells
Detecting Them and Beyond

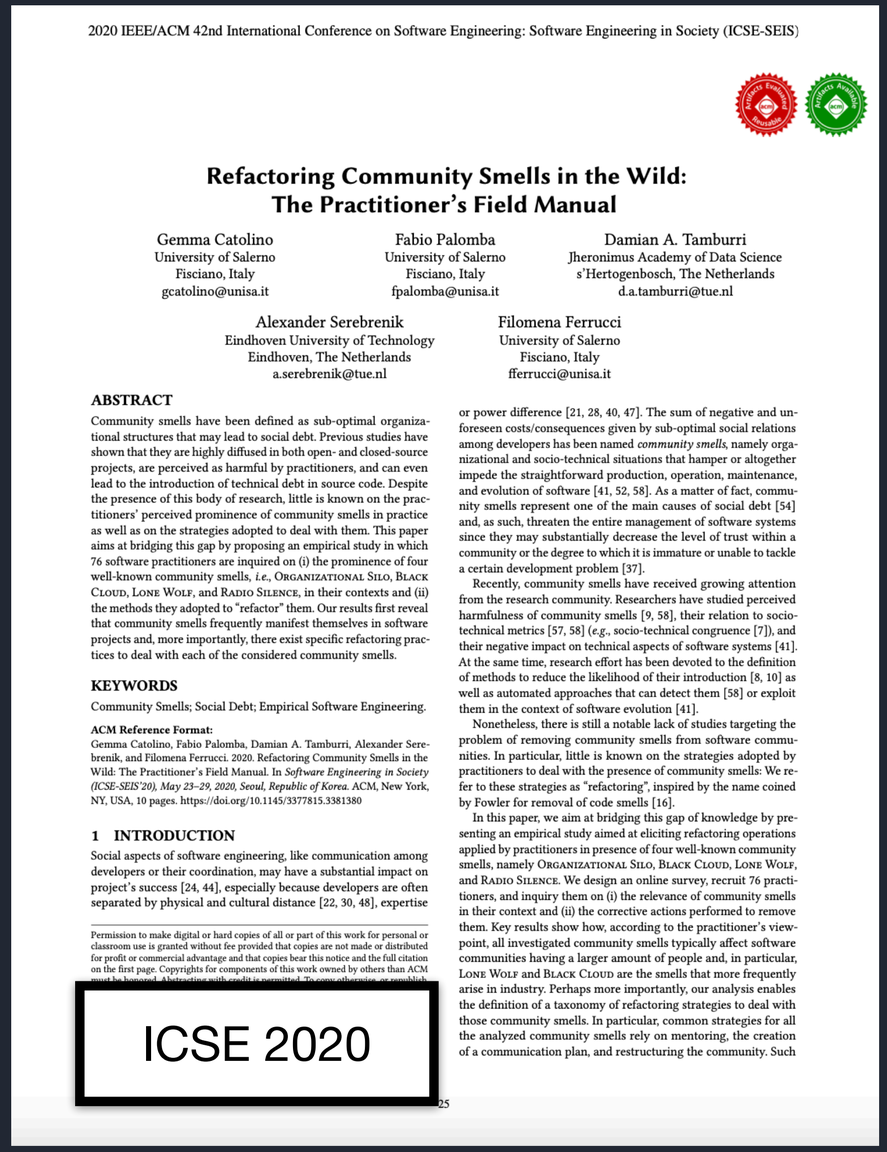
Dealing with Community Smells in Practice

Researchers have first proposed automated detection approaches

More recently, AI has been shown as a promising alternative

Researchers have also investigated how to mitigate their effects

Until the definition of “refactoring” strategies, i.e., how to restructure the community to remove community smells



2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)

Refactoring Community Smells in the Wild: The Practitioner’s Field Manual

Gemma Catolino
University of Salerno
Fisciano, Italy
gcatolino@unisa.it

Fabio Palomba
University of Salerno
Fisciano, Italy
fpalomba@unisa.it

Damian A. Tamburri
Jheronimus Academy of Data Science
s’Hertogenbosch, The Netherlands
d.a.tamburri@tue.nl

Alexander Serebrenik
Eindhoven University of Technology
Eindhoven, The Netherlands
a.serebrenik@tue.nl

Filomena Ferrucci
University of Salerno
Fisciano, Italy
fferrucci@unisa.it

ABSTRACT

Community smells have been defined as sub-optimal organizational structures that may lead to social debt. Previous studies have shown that they are highly diffused in both open- and closed-source projects, are perceived as harmful by practitioners, and can even lead to the introduction of technical debt in source code. Despite the presence of this body of research, little is known on the practitioners’ perceived prominence of community smells in practice as well as on the strategies adopted to deal with them. This paper aims at bridging this gap by proposing an empirical study in which 76 software practitioners are inquired on (i) the prominence of four well-known community smells, *i.e.*, Organizational Silo, Black Cloud, Lone Wolf, and Radio Silence, in their contexts and (ii) the methods they adopted to “refactor” them. Our results first reveal that community smells frequently manifest themselves in software projects and, more importantly, there exist specific refactoring practices to deal with each of the considered community smells.

KEYWORDS

Community Smells; Social Debt; Empirical Software Engineering.

ACM Reference Format:
Gemma Catolino, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, and Filomena Ferrucci. 2020. Refactoring Community Smells in the Wild: The Practitioner’s Field Manual. In *Software Engineering in Society (ICSE-SEIS’20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3377815.3381380

1 INTRODUCTION

Social aspects of software engineering, like communication among developers or their coordination, may have a substantial impact on project’s success [24, 44], especially because developers are often separated by physical and cultural distance [22, 30, 48], expertise or power difference [21, 28, 40, 47]. The sum of negative and unforeseen costs/consequences given by sub-optimal social relations among developers has been named *community smells*, namely organizational and socio-technical situations that hamper or altogether impede the straightforward production, operation, maintenance, and evolution of software [41, 52, 58]. As a matter of fact, community smells represent one of the main causes of social debt [54] and, as such, threaten the entire management of software systems since they may substantially decrease the level of trust within a community or the degree to which it is immature or unable to tackle a certain development problem [37].

Recently, community smells have received growing attention from the research community. Researchers have studied perceived harmfulness of community smells [9, 58], their relation to socio-technical metrics [57, 58] (*e.g.*, socio-technical congruence [7]), and their negative impact on technical aspects of software systems [41]. At the same time, research effort has been devoted to the definition of methods to reduce the likelihood of their introduction [8, 10] as well as automated approaches that can detect them [58] or exploit them in the context of software evolution [41].

Nonetheless, there is still a notable lack of studies targeting the problem of removing community smells from software communities. In particular, little is known on the strategies adopted by practitioners to deal with the presence of community smells: We refer to these strategies as “refactoring”, inspired by the name coined by Fowler for removal of code smells [16].

In this paper, we aim at bridging this gap of knowledge by presenting an empirical study aimed at eliciting refactoring operations applied by practitioners in presence of four well-known community smells, namely Organizational Silo, Black Cloud, Lone Wolf, and Radio Silence. We design an online survey, recruit 76 practitioners, and inquiry them on (i) the relevance of community smells in their context and (ii) the corrective actions performed to remove them. Key results show how, according to the practitioner’s viewpoint, all investigated community smells typically affect software communities having a larger amount of people and, in particular, Lone Wolf and Black Cloud are the smells that more frequently arise in industry. Perhaps more importantly, our analysis enables the definition of a taxonomy of refactoring strategies to deal with those community smells. In particular, common strategies for all the analyzed community smells rely on mentoring, the creation of a communication plan, and restructuring the community. Such

ICSE 2020

Community Smells

Toward their Prediction

So far, researchers have established methods to identify and refactor community smells, but it is still unclear whether and to what extent we can **predict their emergence**!

Community Smells

Toward their Prediction

So far, researchers have established methods to identify and refactor community smells, but it is still unclear whether and to what extent we can **predict their emergence**!

Building an automated mechanism able to alert community shepherds ahead would allow them to **take informed decisions** and possibly act on the community to **mitigate the negative consequences of community smells**

Community Smells
Toward their Prediction

So far, researchers have established methods to identify and refactor community smells, but it is still unclear whether and to what extent we can **predict their emergence**!

Building an automated mechanism able to alert community shepherds ahead would allow them to **take informed decisions** and possibly act on the community to **mitigate the negative consequences of community smells**

Our previous investigations on the matter revealed that **socio-technical metrics are correlated with community smells**

Damian A. Tamburri, Fabio Palomba, Rick Kazman.
Exploring Community Smells in Open-Source: An Automated Approach.
IEEE Transactions on Software Engineering, 2019

Community Smells
Toward their Prediction

So far, researchers have established methods to identify and refactor community smells, but it is still unclear whether and to what extent we can **predict their emergence**!

Building an automated mechanism able to alert community shepherds ahead would allow them to **take informed decisions** and possibly act on the community to **mitigate the negative consequences of community smells**

Our previous investigations on the matter revealed that **socio-technical metrics are correlated with community smells**

Hence, we deepen our analysis and assessed the **predictive power of such socio-technical metrics**

Damian A. Tamburri, Fabio Palomba, Rick Kazman.
Exploring Community Smells in Open-Source: An Automated Approach.
IEEE Transactions on Software Engineering, 2019

Predicting Community Smells

Predicting the Emergence of Community Smells using Socio-Technical Metrics: a Machine-Learning Approach

Fabio Palomba,[1] Damian Andrew Tamburri[2]

[1]*SeSa Lab - University of Salerno, Italy*
[2]*JADE Lab - Eindhoven University of Technology /Jheronimus Academy of Data Science, The Netherlands*
fpalomba@unisa.it, d.a.tamburri@tue.nl

Abstract

Community smells represent sub-optimal conditions appearing within software development communities (*e.g.*, non-communicating sub-teams, deviant contributors, etc.) that may lead to the emergence of social debt and increase the overall project's cost. Previous work has studied these smells under different perspectives, investigating their nature, diffuseness, and impact on technical aspects of source code. Furthermore, it has been shown that some socio-technical metrics like, for instance, the well-known socio-technical congruence, can potentially be employed to foresee their appearance. Yet, there is still a lack of knowledge of the actual predictive power of such socio-technical metrics. In this paper, we aim at tackling this problem by empirically investigating (i) the potential value of socio-technical metrics as predictors of community smells and (ii) what is the performance of within- and cross-project community smell prediction models based on socio-technical metrics. To this aim, we exploit a dataset composed of 60 open-source projects and consider four community smells such as Organizational Silo, Black Cloud, Lone Wolf, and Bottleneck. The key results of our work report that a within-project solution can reach F-Measure and AUC-ROC of 77% and 78%, respectively, while cross-project models still require improvements, being however able to reach an F-Measure of 62% and overcome a random baseline. Among the metrics investigated, socio-technical congruence, communicability, and turnover-related metrics are the most powerful predictors of the emergence of community smells.

Keywords: Community Smells, Social Debt, Empirical Software Engineering;

1. Introduction

Software engineering is de-facto a social activity [1, 2] which involves often thousands if not more stakeholders arranged globally, and often separated by physical and cultural distance (*i.e.*, different cultures and culture clashes [3, 4, 5]), expertise or power distance [6, 7, 8, 9], and more). Literature calls the sum of the negative and unforeseen costs and consequences of such conditions *community smells*, namely sub-optimal organizational and socio-technical situations that hamper or altogether impede the straightforward production, operation, and evolution of software [10, 11, 12]. Previous work established the interaction between community smells and their technical counterparts (*i.e.*, code smells such as *Spaghetti Code*, *God Class* and more [13, 14, 15]) as well as, more generally, technical debt in its various forms [16, 17]. At the same time, community smells have been connected to all sorts of nasty phenomena, e.g., employee turnover [12], bad architecture decisions [11], and more, which often go beyond causing technical issues in software code but may well cause organizational turmoil or even decline [18] of software projects' organizational stability, if not project failure. Therefore, predicting community smells before they manifest along software projects' timelines may prove critical in anticipating sub-optimal organisational and socio-technical conditions before they become unmanageable. In summary, on the one hand, these conclusions from the state of the art indicate that community smells represent a first-class phenomenon to be considered when managing large-scale software systems from an organizational and socio-technical perspective.

On the other hand, previous experience reports and empirical evidence from the state of the art [10, 11, 12] also remarked that predicting community smells at large and anticipating th
of their socio-techni
the types and chara
structure around [1
of their emergence i
and project manage
and possibly re-orga
them [22, 21, 23, 2
investigated various
including their corre
metrics (*e.g.*, socio-t
results showed the
that lead us to believ
potentially useful fo

Predicting Community Smells

Research Questions

RQ_1	What is the predictive power of socio-technical metrics when it comes to community smell prediction?

Predicting Community Smells

Research Questions

RQ_1	What is the predictive power of socio-technical metrics when it comes to community smell prediction?
RQ_2	What is the performance of a community smell prediction model trained using within-project data?

Predicting Community Smells

Research Questions

RQ_1	What is the predictive power of socio-technical metrics when it comes to community smell prediction?
RQ_2	What is the performance of a community smell prediction model trained using within-project data?
RQ_3	What is the performance of a community smell prediction model trained using cross-project data?

Predicting Community Smells

Context and Research Methodology

60

open-source projects with more than
30 contributors and 1k commits

Predicting Community Smells

Context and Research Methodology

60

open-source projects with more than
30 contributors and 1k commits

4

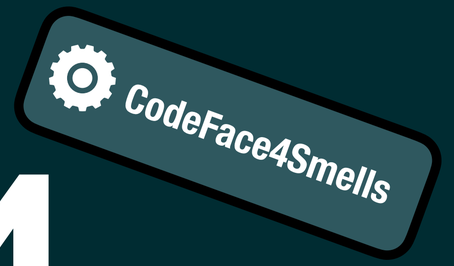
community smells, i.e., Organizational Silo,
Lone Wolf, Bottleneck, Black Cloud

Predicting Community Smells

Context and Research Methodology

60

open-source projects with more than
30 contributors and 1k commits



4

community smells, i.e., Organizational Silo,
Lone Wolf, Bottleneck, Black Cloud

Predicting Community Smells

Context and Research Methodology

CodeFace4Smells

60

open-source projects with more than
30 contributors and 1k commits

4

community smells, i.e., Organizational Silo,
Lone Wolf, Bottleneck, Black Cloud

Predicting Community Smells

Context and Research Methodology

40

metrics of 5 categories

- Developer Social Network metrics
- Socio-technical metrics
- Core-community members metrics
- Turnover metrics
- Social Network Analysis metrics

60

open-source projects with r
30 contributors and 1k c

Predicting Community Smells

Context and Research Methodology

40

metrics of 5 categories

Developer Social Network metrics

Socio-technical metrics

Core-community members metrics

Turnover metrics

Social Network Analysis metrics

60

open-source projects with r
30 contributors and 1k c

Predicting Community Smells

RQ_1. The predictive power of socio-technical metrics

Application of the Gain Ratio algorithm, a measure able to quantity the reduction of entropy provided by each of the considered socio-technical metrics to the model built for predicting community smells.

Predicting Community Smells

RQ_1. The predictive power of socio-technical metrics

Application of the Gain Ratio algorithm, a measure able to quantity the reduction of entropy provided by each of the considered socio-technical metrics to the model built for predicting community smells.

Socio-technical congruence	0.54
Communicability	0.53
Core Global Turnover	0.53
Degree of Centrality	0.48
Core Global Developers	0.47

The smelliness of a community is mainly governed by socio-technical congruence, communicability, and turnover-related metrics

Predicting Community Smells

RQ_1. The predictive power of socio-technical metrics

Application of the Gain Ratio algorithm, a measure able to quantity the reduction of entropy provided by each of the considered socio-technical metrics to the model built for predicting community smells.

Socio-technical congruence	0.54
Communicability	0.53
Core Global Turnover	0.53
Degree of Centrality	0.48
Core Global Developers	0.47

The smelliness of a community is mainly governed by socio-technical congruence, communicability, and turnover-related metrics

More in general, we found that all the considered socio-technical metrics contribute to the reduction of entropy of the model, which means that the correlations identified in our previous work are also meaningful from a predictive perspective.

Community smells can be potentially predicted by means of socio-technical metrics.

Predicting Community Smells

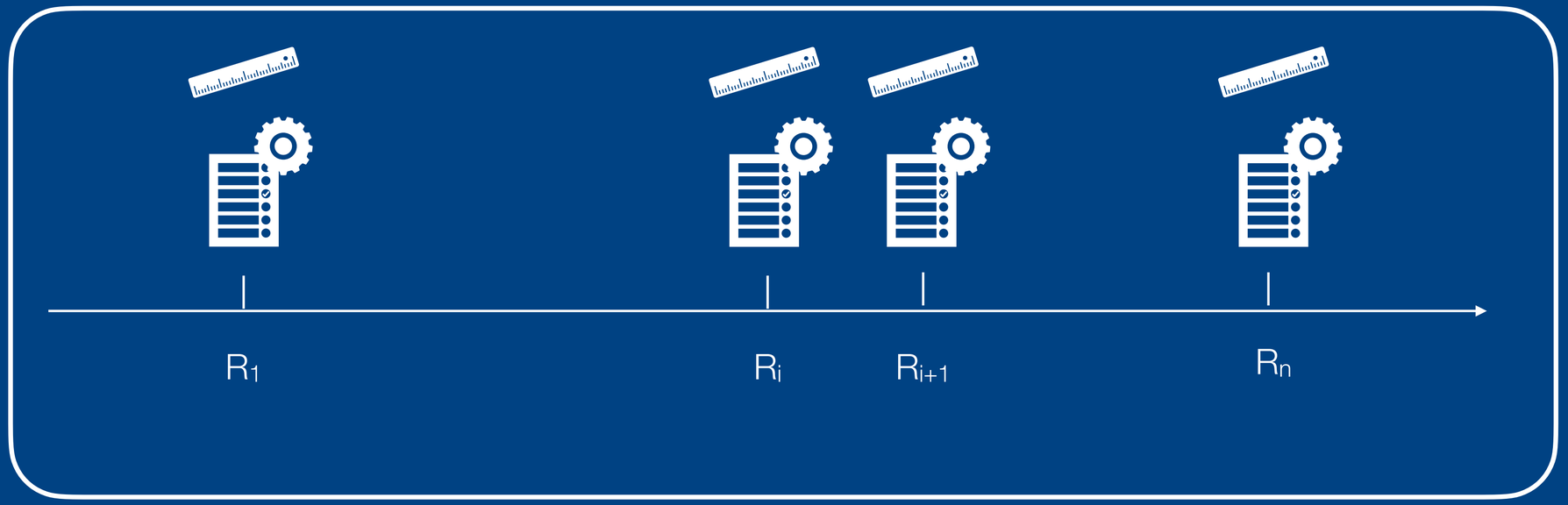
RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

We train one model for each community smell considering both a within- and cross-project strategy.

Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

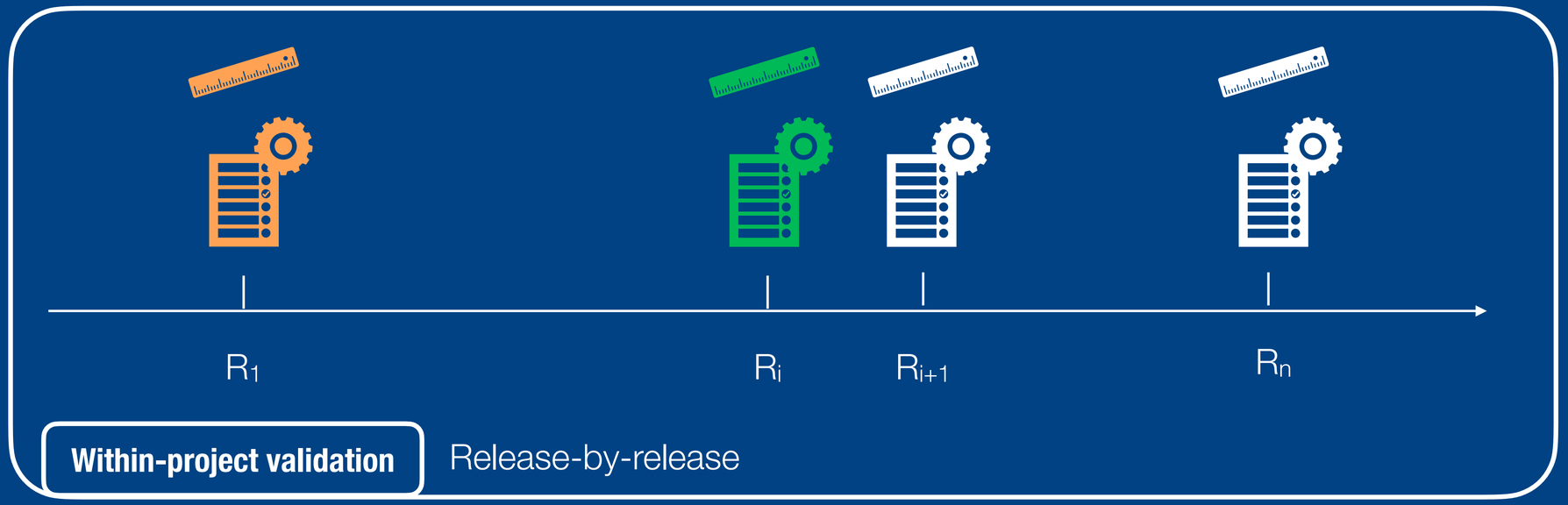
We train one model for each community smell considering both a within- and cross-project strategy.



Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

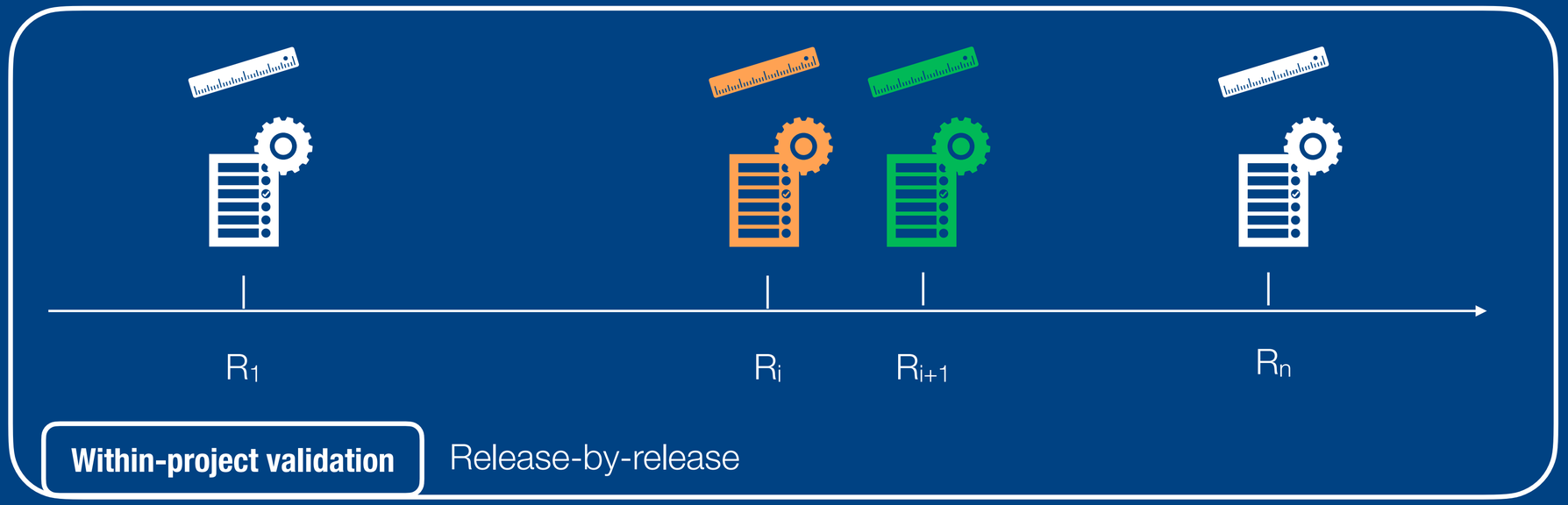
We train one model for each community smell considering both a within- and cross-project strategy.



Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

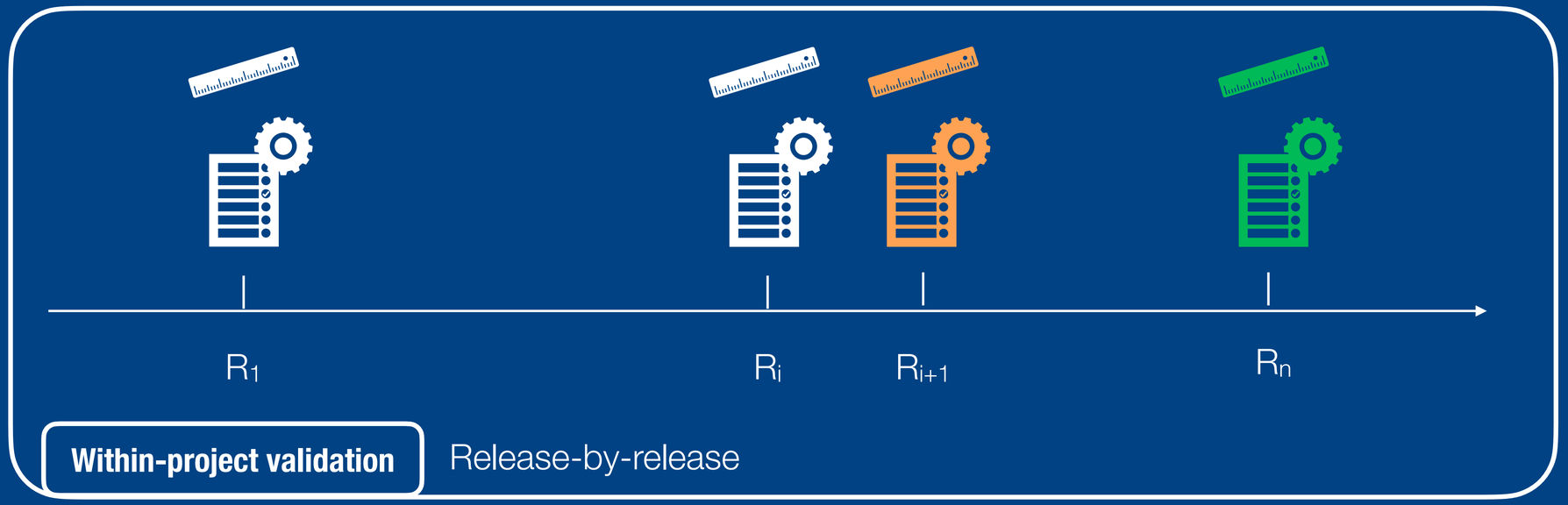
We train one model for each community smell considering both a within- and cross-project strategy.



Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

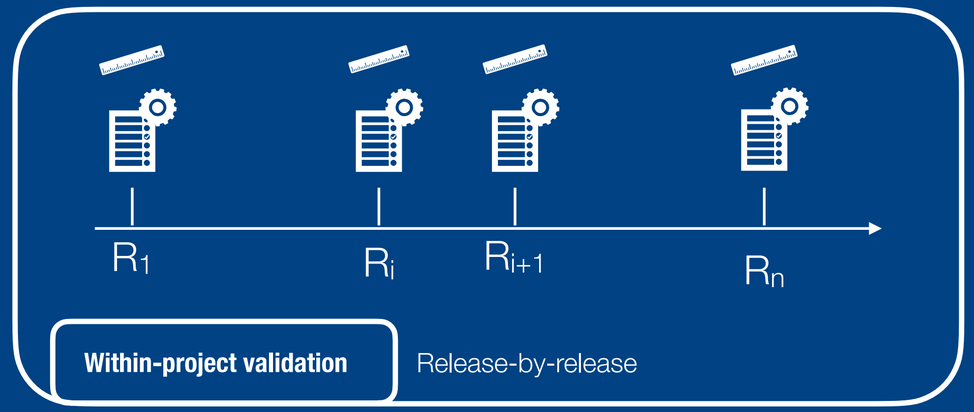
We train one model for each community smell considering both a within- and cross-project strategy.

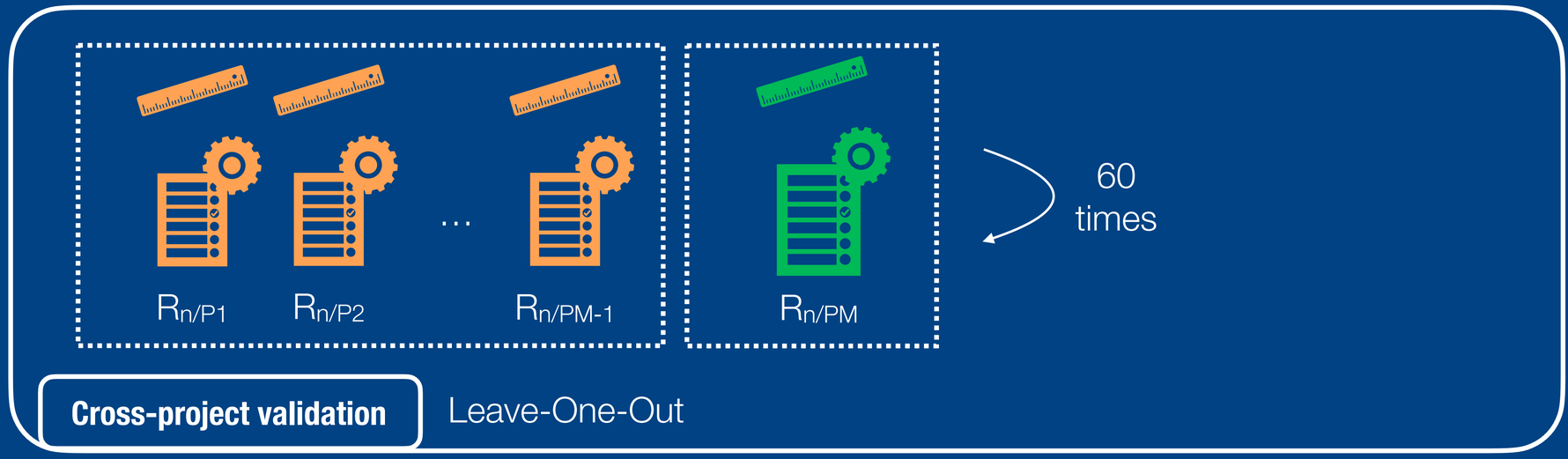


Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

We train one model for each community smell considering both a within- and cross-project strategy.

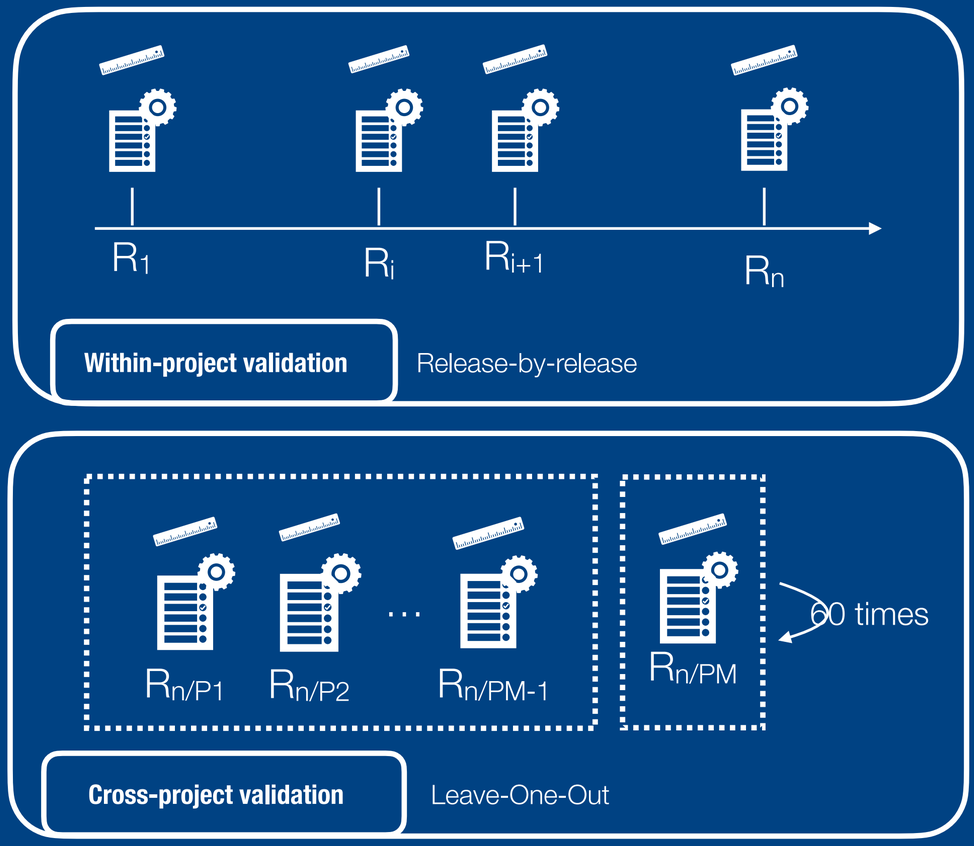




Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

We train one model for each community smell considering both a within- and cross-project strategy.



Configuration of the models

Random Forest, J48, Logistic Regression, Decision Table, Naive Bayes

Feature selection and data balancing techniques applied to deal with common multi-collinearity and balance issues

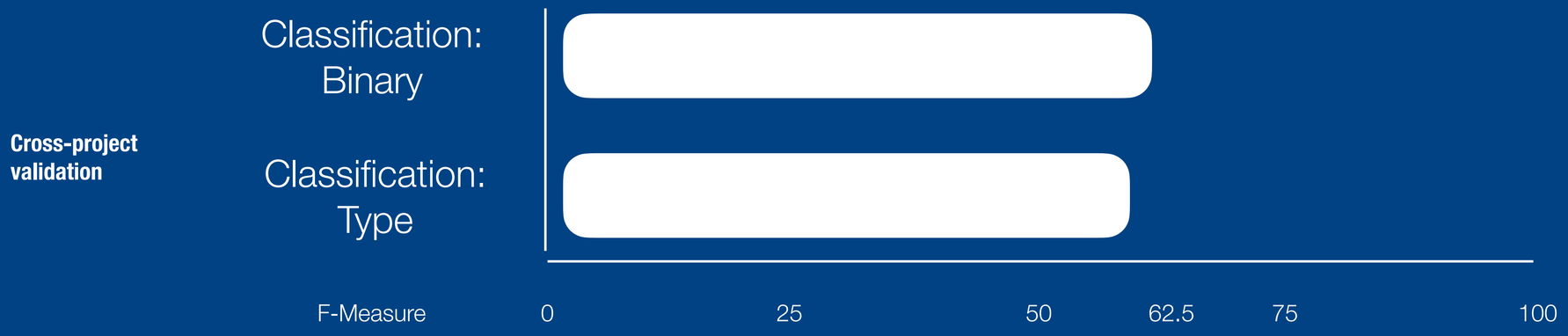
In the cross-project context, data were scaled and normalized to deal with possible data heterogeneity

Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

Predicting Community Smells

RQ_2 - RQ_3. Assessing the Community Smell Prediction Models



Community smells can be predicted by exploiting socio-technical metrics.

Community Smells
Toward their Prediction

So far, researchers have established methods to identify and refactor community smells, but it is still unclear whether and to what extent we can **predict their emergence**!

Building an automated mechanism able to alert community shepherds ahead would allow them to **take informed decisions** and possibly act on the community to **mitigate the negative consequences of community smells**

Our previous investigations on the matter revealed that **socio-technical metrics are correlated with community smells**

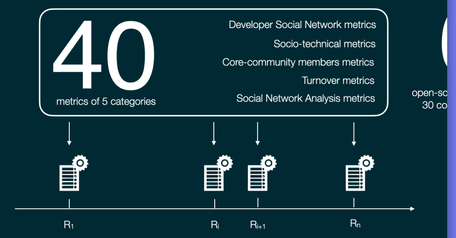
Hence, we deepen our analysis and assessed the **predictive power of such socio-technical metrics**

Damian A. Tamburri, Fabio Palomba, Rick Kazman,
Exploring Community Smells in Open-Source: An Automated Approach.
IEEE Transactions on Software Engineering, 2019

Predicting Community Smells
Research Questions

RQ_1	What is the predictive power of socio-technical metrics when it comes to community smell prediction?
RQ_2	What is the performance of a community smell prediction model trained using within-project data?
RQ_3	What is the performance of a community smell prediction model trained using cross-project data?

Predicting Community Smells
Context and Research Methodology



Predicting Community Smells
RQ_1. The predictive power of socio-technical metrics

Application of the Gain Ratio algorithm, a measure able to quantity the reduction of entropy provided by each of the considered socio-technical metrics to the model built for predicting community smells.

Socio-technical congruence	0.54
Communicability	0.53
Core Global Turnover	0.53
Degree of Centrality	0.48
Core Global Developers	0.47

The smelliness of a community is mainly governed by socio-technical congruence, communicability, and turnover-related metrics

More in general, we found that all the considered socio-technical metrics contribute to the reduction of entropy of the model, which means that the correlations identified in our previous work are also meaningful from a predictive perspective.

Community smells can be potentially predicted by means of socio-technical metrics.

Predicting Community Smells
RQ_2 - RQ_3. Assessing the Community Smell Prediction Models

Community smells can be predicted by exploiting socio-technical metrics.

Predicting the Emergence of Community Smells using Socio-Technical Metrics: A Machine Learning Approach

Fabio Palomba
Assistant Professor
Software Engineering (SeSa) Lab
University of Salerno

fpalomba@unisa.it
@fabiopalomba3
https://fpalomba.github.io

sesa lab
SOFTWARE ENGINEERING
SALERNO

Community Smell Detection and Refactoring in SLACK:
The CADOCS Project

G. Voria, V. Pentangelo, A. Della Porta,
***S. Lambiase**, G. Catolino, F. Palomba, F. Ferrucci*

Stefano Lambiase

slambiase@unisa.it

@StefanoLambiase

https://stefanolambiase.github.io

SOFTWARE ENGINEERING SALERNO

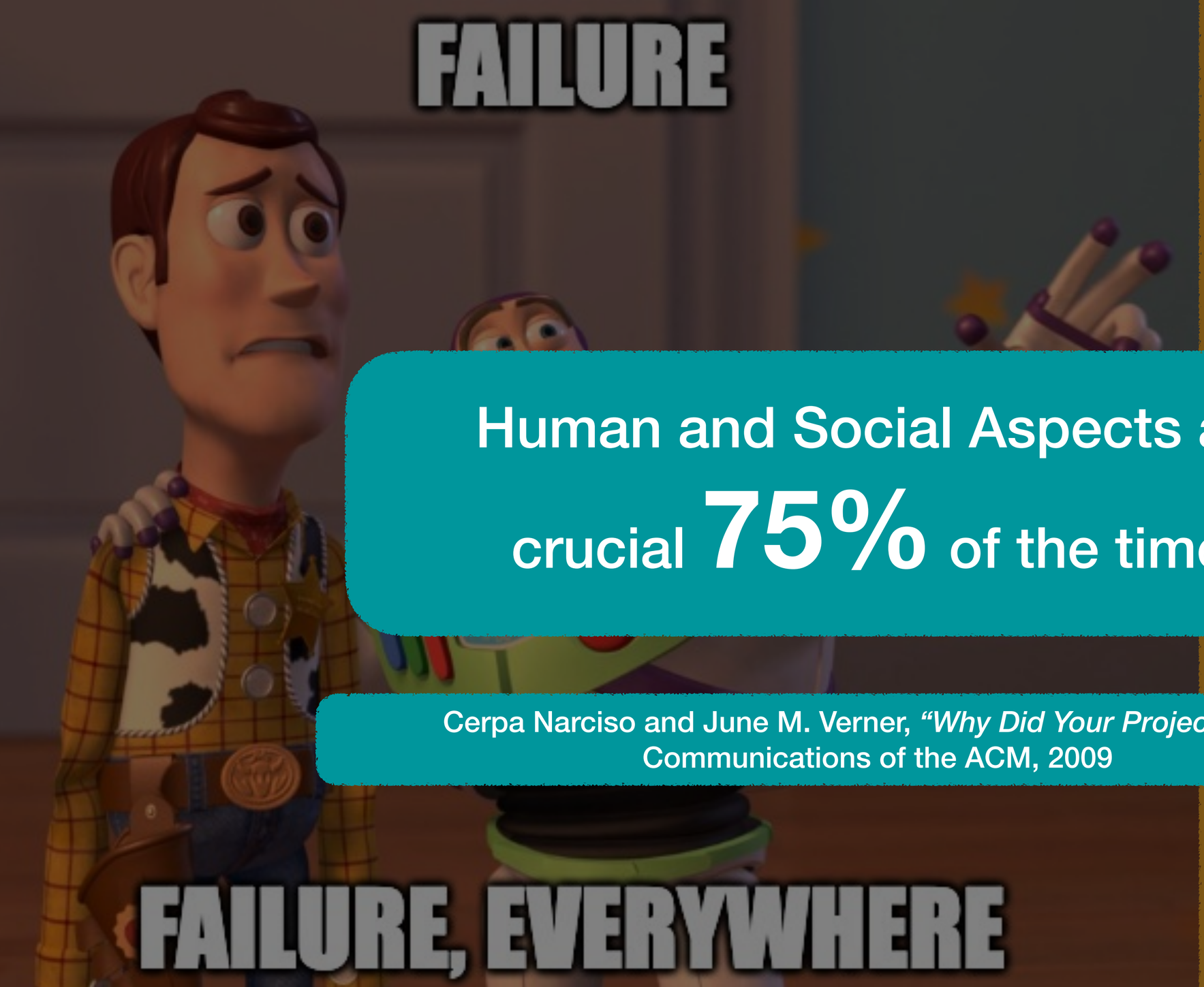
Jheronimus Academy of Data Science

70%

of software projects

FAIL

Jim Johnson,
“CHAOS 2020: Beyond Infinity.”,
Standish Group (2020).



Human and Social Aspects are crucial **75%** of the time!

Cerpa Narciso and June M. Verner, *"Why Did Your Project Fail?"*, Communications of the ACM, 2009

70%

tware projects

FAIL

Jim Johnson, *"CHAOS 2020: Beyond Infinity."*, Standish Group (2020).

Organizational Silo

Black Cloud

Prima-Donnas Effect

Toxic Communication

Community Smells

Organizational Skirmish

Sub-optimal anti-patterns across the organizational and social structure in a software development community that are precursors of alarming and unforeseen socio-technical events

Unhealthy Interaction

Radio Silence

Solution Defiance

Sharing Villainy

Truck Factor Smell

Tamburri et al., *“The architect’s role in community shepherding”*, IEEE Software, 2016

Yoshi
2018

Tamburri et al.

“Discovering community patterns in open-source: a systematic approach and its evaluation”

Empirical Software Engineering, 2018

Tamburri et al.

“Exploring community smells in open-source: An automated approach”

Transactions on Software Engineering, 2019

2019
Codeface for Smells

csDetector
2021

Almarimi et al.

“csDetector: An Open Source Tool for Community Smells Detection”

ESEC/FSE, 2021

Yoshi
2018

Tamburri et al.

“Discovering community patterns in open-source: a systematic approach and its evaluation”

Empirical Software Engineering, 2018

Tamburri et al.

“Exploring community smells in open-source: An automated approach”

Transactions on Software Engineering, 2019

2019
Codeface for Smells

csDetector
2021

Almarimi et al.

“csDetector: An Open Source Tool for Community Smells Detection”

ESEC/FSE, 2021

No refactoring suggestions

Not designed for practitioners

CADOCS

A **conversational agent** developed for the Slack platform and designed around two main functional requirements

Detection of Community Smells

Almarimi et al.

“csDetector”

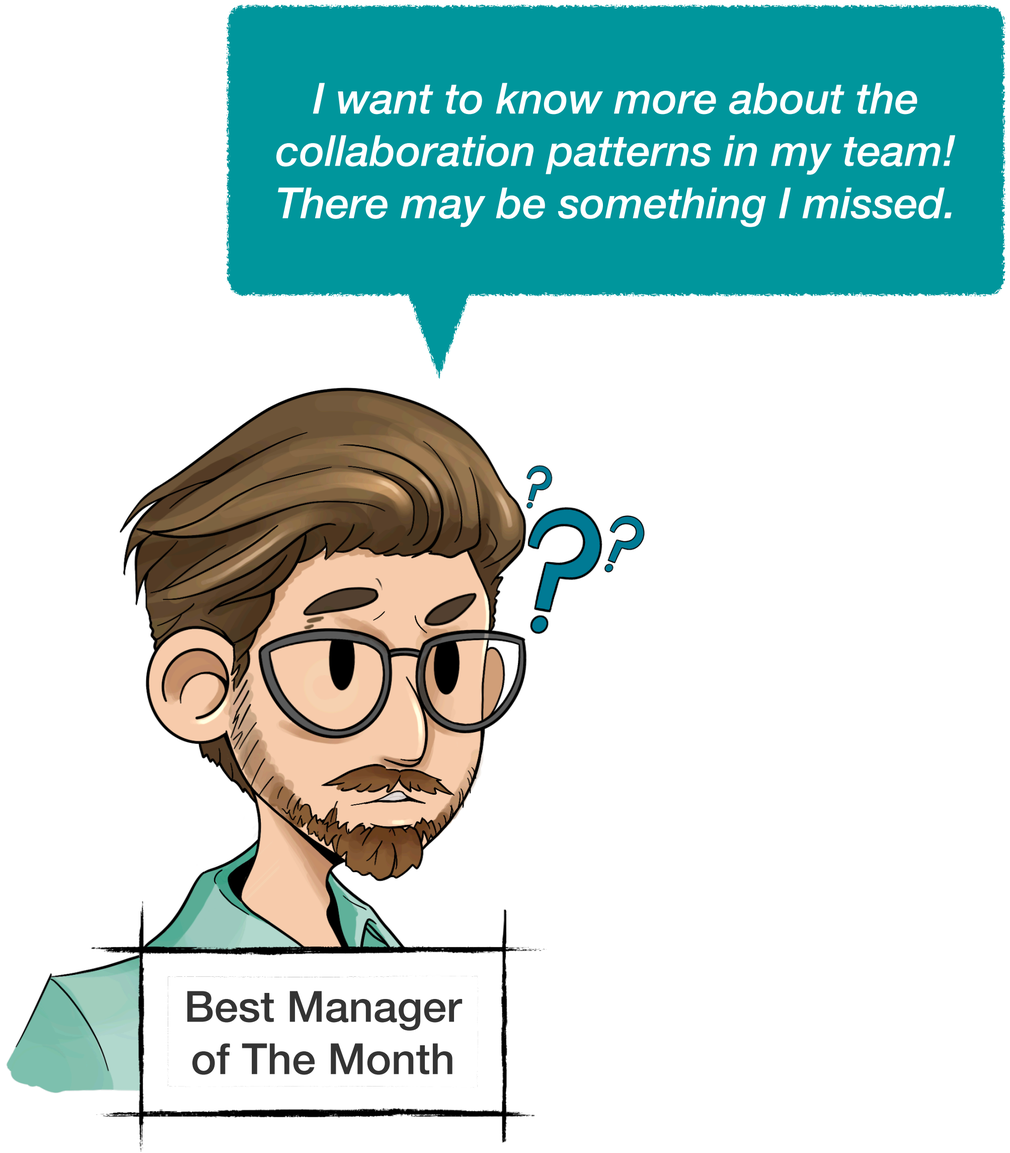
ESEC/FSE 2021

Refactoring of Community Smells

Catolino et al.

"Refactoring community smells in the wild”

ICSE-SEIS 2020


I want to know more about the collaboration patterns in my team! There may be something I missed.
Best Manager of The Month

I want to know more about the collaboration patterns in my team! There may be something I missed.

I want to know more about the collaboration patterns in my team! There may be something I missed.

DOCS

Analysis of the

GitHub repository

↓

Detection of

Community Smells

↓

Generation of

the report

I want to know more about the collaboration patterns in my team! There may be something I missed.

Input

Link to the project GitHub repository

Analysis of the GitHub repository

Detection of Community Smells

Generation of the report

Excellent!

Input

Link to the project
GitHub repository

Output

List of Community Smells
and refactoring strategies

DOCS

Analysis of the
GitHub repository

Detection of
Community Smells

Generation of
the report

How does CADOCS work?

User Request

Smells Identification

Smells Refactoring

GIANMARIO VORIA 16:40
hi cadocs, can you give me community smells in https://github.com/microsoft/QuantumKatas

CADOCS APP 16:41

Hi GIANMARIO 👋

This is the community smells we were able to detect in the repository https://github.com/microsoft/QuantumKatas :

OSE Organizational Silo Effect 👥👤👥

Siloed areas of the community that do not communicate, except through one or two of their respective members.

BCE Black-cloud Effect ⚫☁️⚫

Information overload due to lack of structured communications or cooperation governance.

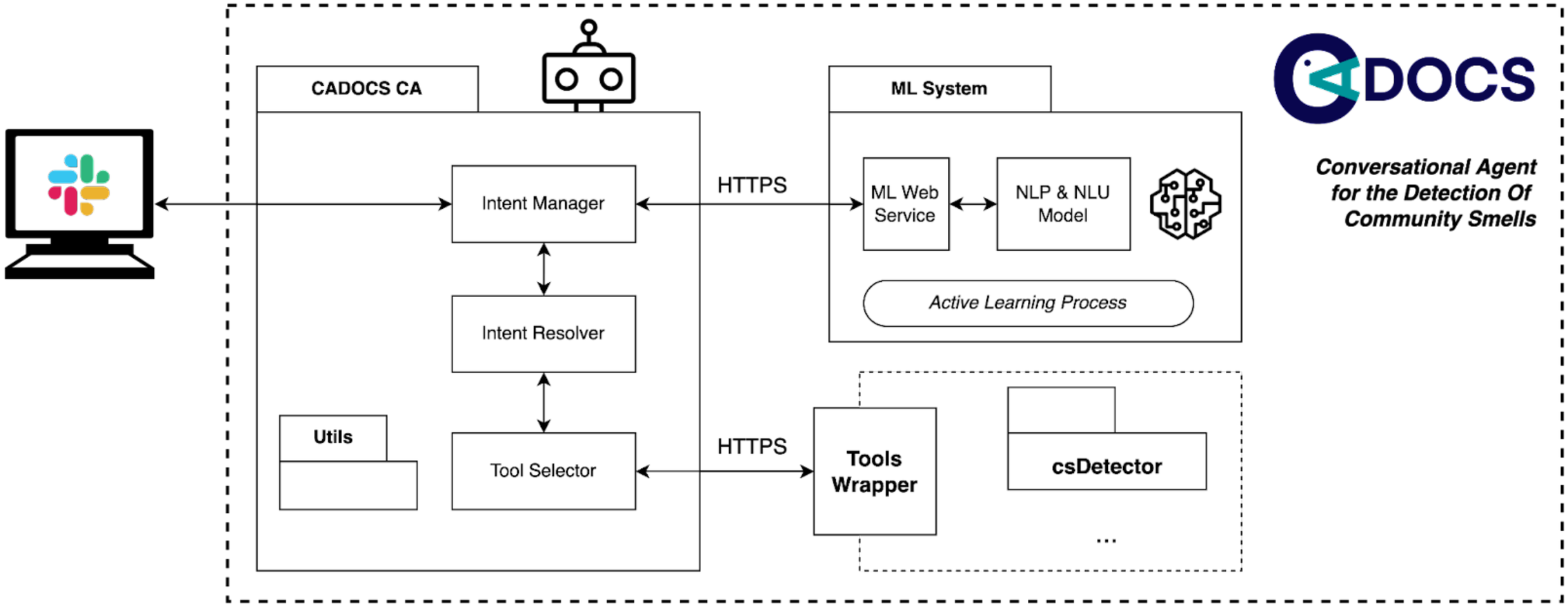
Some possible mitigation strategies are:

> Create communication plan ⭐⭐⭐

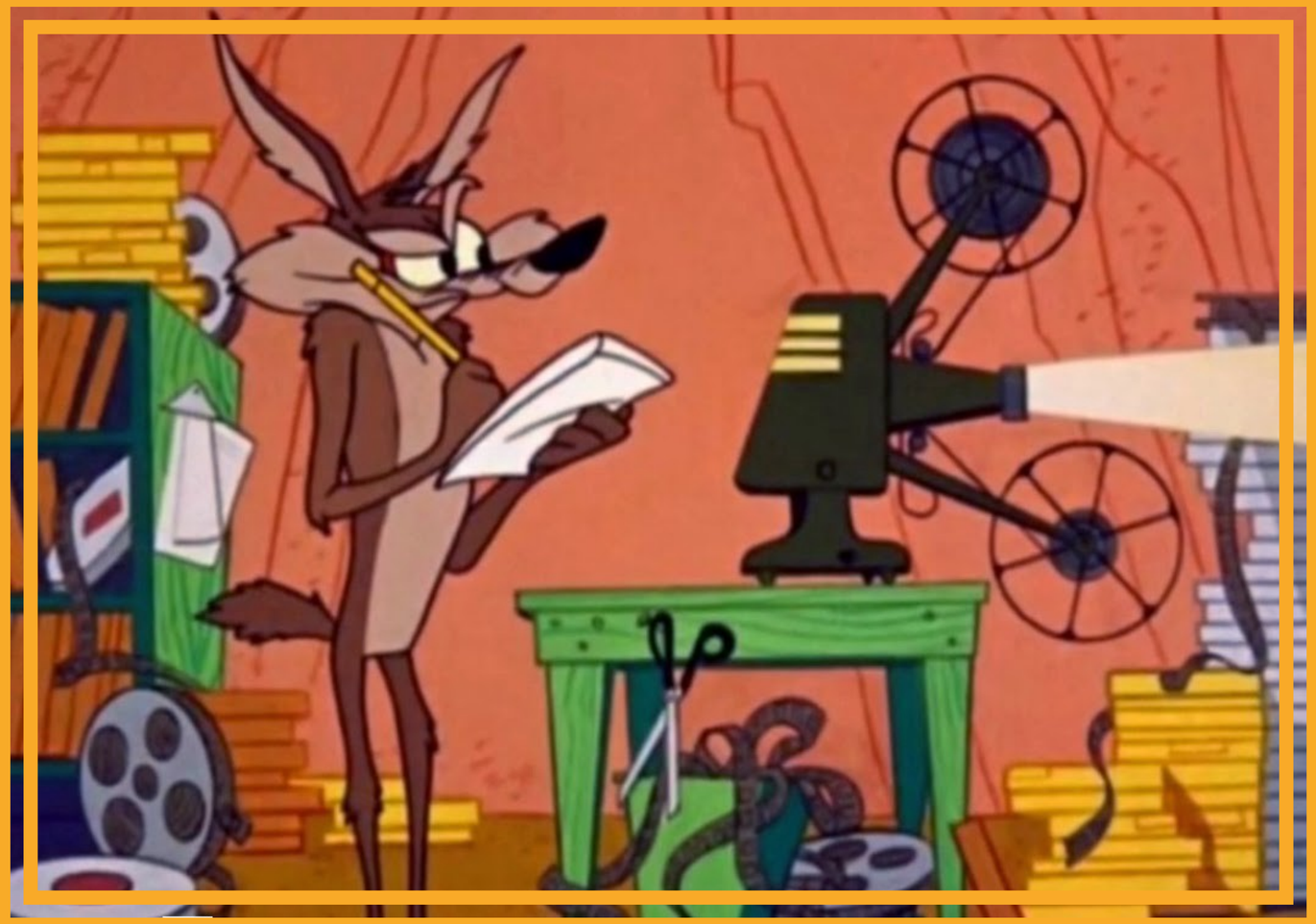
> Restructure the community ⭐⭐

> Introduce a Social sanctioning mechanism ⭐⭐

Architecture of the Tool



S. Srivastava and T. V. Prabhakar, *"A Reference Architecture for Applications with Conversational Components"*, ICSESS 2019



Evaluation

Improving of NLU Over Time

We performed *input testing*, i.e., an approach based on metamorphic testing that aims at identifying potential reasons for unsuccessful training in the used data.

We applied *iterative usability testing*, i.e., a strategy based on an iterative process in which, at each step, users give feedback about the tool's user interface and developers modify the tool accordingly.

Usability Evaluation

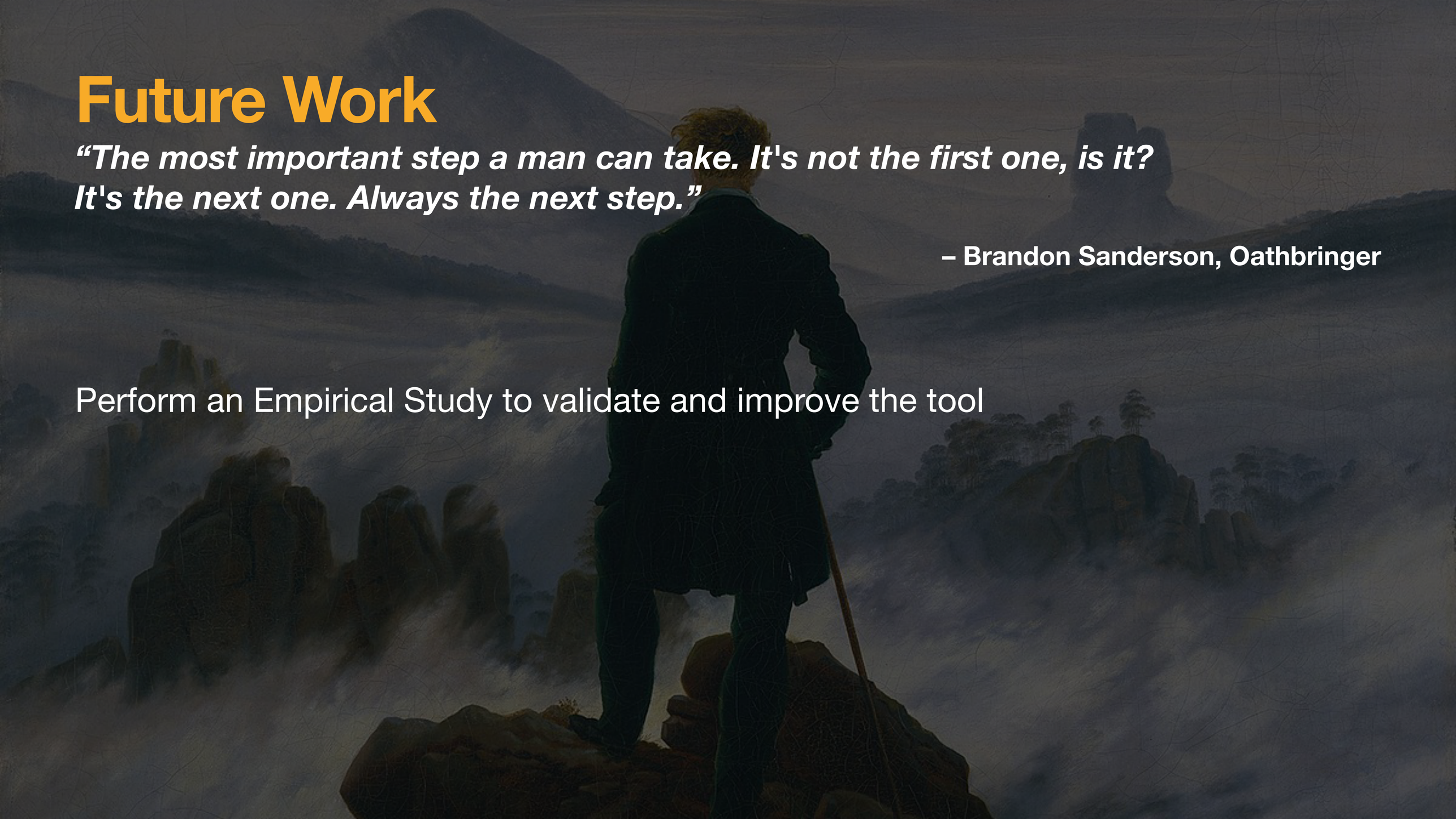
Potential Impact

CADOCS for **Maintenance** and **Evolution**

By providing social information about communities, thanks to which managers could make more informed and faster decisions

CADOCS for **Knowledge Sharing**

By allowing the integration of different tools because of its modular architecture and ease of use and modification



Future Work

“The most important step a man can take. It's not the first one, is it? It's the next one. Always the next step.”

– Brandon Sanderson, Oathbringer

Perform an Empirical Study to validate and improve the tool

Future Work

“The most important step a man can take. It's not the first one, is it? It's the next one. Always the next step.”

– Brandon Sanderson, Oathbringer

Perform an Empirical Study to validate and improve the tool

Increase the number of tools CADOCS can use

Future Work

“The most important step a man can take. It's not the first one, is it? It's the next one. Always the next step.”

– Brandon Sanderson, Oathbringer

Perform an Empirical Study to validate and improve the tool

Increase the number of tools CADOCS can use

Combine different tools for the detection in an “Ensemble learning-fashion”