Uncovering Community Smells in Machine Learning-Enabled Systems: Causes, Effects, and Mitigation Strategies

GIUSY ANNUNZIATA, University of Salerno, Italy
STEFANO LAMBIASE, University of Salerno, Italy
DAMIAN A. TAMBURRI, University of Sannio, Italy
WILLEM-JAN VAN DEN HEUVEL, Jheronimus Academy of Data Science, Netherlands
FABIO PALOMBA, University of Salerno, Italy
GEMMA CATOLINO, University of Salerno, Italy
FILOMENA FERRUCCI, University of Salerno, Italy
ANDREA DE LUCIA, University of Salerno, Italy

Successful software development hinges on effective communication and collaboration, which are significantly influenced by human and social dynamics. Poor management of these elements can lead to the emergence of 'community smells', i.e., negative patterns in socio-technical interactions that gradually accumulate as 'social debt'. This issue is particularly pertinent in machine learning-enabled systems, where diverse actors such as data engineers and software engineers interact at various levels. The unique collaboration context of these systems presents an ideal setting to investigate community smells and their impact on development communities. This paper addresses a gap in the literature by identifying the types, causes, effects, and potential mitigation strategies of community smells in machine learning-enabled systems. Using Partial Least Squares Structural Equation Modeling (PLS-SEM), we developed hypotheses based on existing literature and interviews, and conducted a questionnaire-based study to collect data. Our analysis resulted in the construction and validation of five models that represent the causes, effects, and strategies for five specific community smells. These models can help practitioners identify and address community smells within their organizations, while also providing valuable insights for future research on the socio-technical aspects of machine learning-enabled system communities.

CCS Concepts: • **Partial Least Squares Structural Equation Modeling**; • **Socio-Technical Aspects in Machine Learning-Enabled Systems**; • **Community Smells**;

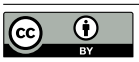
Additional Key Words and Phrases: Socio-Techinical Aspects, ML-Enabled Teams, Partial Least Squares Structural Equation Modeling, PLS-SEM

1 INTRODUCTION

Software development is a collaborative activity, and scholars in the field of software engineering have been investigating how human and social factors can influence this activity. Research highlights the importance of effectively managing these factors, as they can greatly impact the overall success of a software project, especially in Machine Learning (ML) projects which typically involve even more stakeholders and are notoriously failure-prone.[1]

[1]https://hbr.org/2023/11/keep-your-ai-projects-on-track

Authors' addresses: Giusy Annunziata, gannunziata@unisa.it, University of Salerno, Salerno, Italy; Stefano Lambiase, slambiase@unisa.it, University of Salerno, Salerno, Italy; Damian A. Tamburri, datamburri@unisannio.it, University of Sannio, Benevento, Italy; Willem-Jan van den Heuvel, W.J.A.M.vdnHeuvel@tilburguniversity.edu, Jheronimus Academy of Data Science, Hertogenbosch, Netherlands; Fabio Palomba, fpalomba@unisa.it, University of Salerno, Salerno, Italy; Gemma Catolino, gcatolino@unisa.it, University of Salerno, Salerno, Italy; Filomena Ferrucci, fferrucci@unisa.it, University of Salerno, Salerno, Italy; Andrea De Lucia, adelucia@unisa.it, University of Salerno, Salerno, Italy.



Previous research focused on addressing social issues within the context of software development, particularly in relation to the concept of **social debt** [13, 59]. Paraphrasing its original definition, this concept refers *"unforeseen project costs arising from suboptimal interactions within a development community [59]*. From the various aspects influencing social debt, *community smells* have emerged as significant factors; they represent socio-technical characteristics (such as high formality) and patterns (like recurring condescending behavior or abrupt departures) that can contribute to the accumulation of social debt [58, 59].

Recent studies have shifted their focus towards understanding the spread and impact of community smells, as well as identifying the factors associated with their emergence [2, 13, 20, 23, 38]. Some research has investigated how the presence of women and gender diversity within development teams may be connected to the occurrence of community smells [16]. Other works focused on identifying ways for supporting the detection of community smells, thus resulting in tools to perform future investigation [64]. Some efforts have been devoted to identifying mitigation and refactoring strategies for dealing with community smells [17].

Looking at the current state of the art, we identified only a few studies aiming at investigating community smells within software development projects involving ML components [44, 46].

Analyzing the works that investigate on social dynamics in ML-enabled systems has emerged the one of Busquim et al.[12], reports on semi-structured interviews with professionals in software engineering and data science providing a reflexive thematic analysis. The results reveal several challenges that may hinder collaboration between software engineers and data scientists, including differences in technical expertise, unclear job definitions and a lack of documentation supporting the specification of ML-enabled systems. Indeed, the software teams involved in the development of those projects are commonly characterized by the collaboration of individuals which diverse backgrounds, often working within isolated clusters and adhering to specific communication protocols. However, this collaborative setting may also give rise to social anti-patterns, manifested as community smells. Examining these patterns has the potential to uncover crucial insights into the organizational factors contributing to failures in ML-enabled systems [10, 41, 51].

Our investigation stems from these considerations and aims to study community smells in ML-enabled systems. Our focus is on identifying which community smells appear in such communities, their potential causes, their effects, and their mitigation strategies. To conduct our study, we used the Partial Least Squares Structural Equation Modeling (PLS-SEM) statistical analysis method [31, 54]. Specifically, we formulated our hypotheses (as described in the first step of the application of PLS-SEM) by (1) starting from the work of Mailach and Siegmund [44] who identified 17 socio-technical anti-patterns, 16 causes, and 15 organizational strategies and (2) conducting five interviews with ML-enabled systems developers. To obtain the data to measure each phenomenon and to create our measurement model (the second step of PLS-SEM), we performed a questionnaire-based study.

Our main contribution consists of five models representing causes, effects, and strategies for five community smells, i.e., organizational skirmish, organizational silo, lone wolf, prima donna, and black cloud [58, 59]. Those smells were selected after a preliminary analysis (explained in Section 4) that shows their inherent direct relationship with the socio-technical anti-patterns of ML software.

Practitioners may use these models to identify smells in their organization, resolve them, and understand how to mitigate them in the future. Moreover, our investigation sheds light on potential novel research contributions concerning socio-technical aspects in communities developing AI- and ML-enabled systems.

Structure of the Paper. Section 2 of the paper delves into the background and related work. Section 3 outlines the study, detailing its objectives and motivation. The following sections explain various aspects of the research study. Specifically, Section 4 discuss the preliminary analysis conducted.

Sections 5 and 6, respectively, describe the development of the Structural Models and Measurement Models. Section 7 details the data collection process. Section 8 explains the statistical analysis performed on the Structural and Measurement Models to address the research questions. Finally, Section 9 discusses the implications of the results, the potential threats to the study, and the measures taken to mitigate them.

2 BACKGROUND AND RELATED WORK

In this section, we report an overview of the research method used in this study, i.e., PLS-SEM. Then we provide an overview of the research on community smells, followed by a review of work that is related to ours in the context of ML-enabled systems.

2.1 Partial Least Squares Structural Equation Modeling (PLS-SEM)

Structural Equation Modeling (SEM) is a robust multivariate analysis technique widely recognized for its ability to handle complex structures and models effectively [31]. SEM is particularly useful for analyzing complex models where variables may be both *non-latent*—classically observable and directly measurable variables—and *latent*, meaning they represent underlying traits, qualities, or phenomena that are not directly measurable but can be estimated through multiple indicators [31, 55]. This method excels in scenarios where classical linear regression falls short, such as when variables (latent and non-latent) are interconnected through a series of paths (also known as *hypotheses*) and the relationships are not limited to a single type. Moreover, SEM allows for multiple dependent and independent variables to be simultaneously analyzed, providing a comprehensive view of their interrelationships.

One of the most intriguing aspects of SEM is the concept of latent variables and their measurement. Latent variables are constructs that cannot be directly observed or measured; instead, their measurement is achieved through a set of observable indicators, often referred to as *indexes* [31]. For instance, in marketing research, "Customer Satisfaction" is a latent variable that represents an abstract construct. Although it cannot be directly measured, it can be estimated using multiple observable indicators such as Overall Satisfaction Rating, Recommendation Likelihood, and Service Quality Perception. Typically, researchers use questionnaires with validated items and scales as indexes to measure these latent constructs, ensuring accurate and reliable estimation of the underlying variables.

In general, a latent variable can be measured by combining different indexes by the mean of two types of measurements: *formative* and *reflective* [31]. Reflective measures assume that the latent variable causes the observed indicators. In this approach, the indicators are manifestations of the underlying latent construct, and any change in the latent variable is expected to cause changes in all of its indicators. Reflective indicators are assumed to be interchangeable, meaning each indicator should capture the same underlying construct [31]. Formative measures assume that the indicators cause the latent variable. Here, the latent construct is viewed as being formed or defined by its indicators. Each indicator captures a different aspect of the construct, meaning indicators are not interchangeable and do not necessarily correlate with each other [31]. In general, researchers tend to prefer reflective measurements because of the interchangeability property.

Among the various SEM techniques, Partial Least Squares Structural Equation Modeling (PLS-SEM) [31, 54] stands out for its unique applicability in specific research contexts, particularly when the focus is on predictive accuracy and theory development. PLS-SEM is particularly suited for predictive analysis and theory development, making it popular in disciplines such as marketing, information systems, and social sciences [31]. Unlike other SEM techniques, PLS-SEM does not impose strict assumptions about data distribution and sample size, allowing for greater flexibility in research design. The method works by maximizing the explained variance of the dependent

variables, thereby enhancing the predictive accuracy of the model. PLS-SEM's iterative algorithm generates latent variable scores, which are used to estimate path coefficients and assess the quality of the measurement and structural models.

In the context of Partial Least Squares Structural Equation Modeling, the analysis is typically divided into two main components: the *measurement model* and the *structural model* [31, 54, 55]. Both components serve distinct purposes and require different validation processes. The measurement model, also known as the outer model, focuses on the relationships between latent variables and their observed indicators. It specifies how well the observed variables (indicators) represent the latent constructs. The structural model, also known as the inner model, focuses on the relationships between latent variables. It specifies the hypothesized relationships or paths among the latent constructs in the model.

Despite the potential of adopting PLS-SEM in research, few studies have adopted this approach [54]. In the context of Software Engineering research, Trinkenreich et al. [61] adopted PLS-SEM to understand practitioners' sense of belonging to a community and the causes that may reinforce it. Results show that a sense of community helps individuals feel valued, satisfied, and involved. However, how a sense of community is created and what factors influence it needs to be clarified. Another work, also proposed by Trinkenreich et al. [62], focused on studying the relationship between the organizational culture of a developer and burnout, demonstrating that the role of national culture is attractive to large multinational organizations.

2.2 Socio-Technical Anti-Patterns in ML-enabled systems

Machine Learning (ML)-enabled systems are defined as software systems provided with a machine learning component [46]. Those systems have become all-pervasive in our lives, integrating into various aspects—from professional to personal—and supporting us in all our activities [11].

Implementing ML-enabled systems is a complex activity, especially concerning the machine learning components, which may lead to the inadvertent introduction of technical debts [46]. An example of this is code smells—sub-optimal coding practices that hinder the readability, maintainability, or scalability of a system. In the context of machine learning applications, specific types of code smells have been identified that uniquely impact these systems, such as issues related to data handling, model management, and hyperparameter configuration, which can lead to reduced performance, increased technical debt, and greater difficulty in model updates, as highlighted by Zhang et al. [66]. Other example are the data smells-Foidl et al. [26] categorize these data smells and identify their causes, including flawed data collection processes and improper handling, while also highlighting their severe consequences for AI-based systems, such as skewed model predictions and degraded performance. Similarly, Recupito et al. [52] extended the catalog of identified data smells with new data smells and empirically analysis of prevalence and data quality. The complexity derived from technical debts increases the company's demand for professional figures with specific skills in the ML context, e.g., machine learning experts, data scientists, and data engineers. In the process of hiring such figures other than more classical ones—like software engineers, developers, and testers—*heterogeneity in the professional background arose in development teams.* Despite such heterogeneity could improve the development process, it might also lead to the emergence of socio-technical challenges, thus impacting the quality of the developed software [46].

One notable study examining social dynamics in ML-enabled systems is by Busquim et al. [12]. They conducted semi-structured interviews with software engineering and data science professionals, followed by a reflexive thematic analysis. Their findings highlight several challenges that can impede collaboration between these groups, such as disparities in technical expertise, ambiguous job roles, and a lack of documentation for ML system specifications.

Nahar et al. [46] investigated the main collaboration and communication issues during the development of ML-enabled systems. They found that most problems are caused by miscommunication between participants with different backgrounds. The heterogeneity of developers leads to having different technical languages, which makes communication with each other and similarly with the manager compact. Specifically, Mailach and Siegmund [44] investigated socio-technical aspects in ML-enabled systems. Their work starts with 210 videos on the ML-OPS community.[2] Following the application of relevance filters, they obtain 73 videos relevant. Analyzing these videos they extracted a list of 17 socio-technical anti-pattern, which describe developers' bad behavior; they also list 16 possible causes and 15 organizational strategies to adopt. The study highlights an open question: *Are the socio-technical anti-patterns identified specific to the development of ML-enabled systems or more general socio-technical challenges in software engineering?*

2.3 Community Smells

In the realm of the social dimensions of software development, Tamburri et al. [58, 59] introduced the concept of **"community smells"**. These are socio-technical patterns that have the potential to result in social debt. For example, the Organizational Silo effect is a specific community smell that signifies a scenario in which a software community becomes fragmented into isolated compartments where communication primarily occurs through just one or two members in each compartment.

In terms of impact, Palomba et al. [48] unveiled a connection between community-related factors and code smells, suggesting that the former can exacerbate the latter. Consequently, there has been a proposal for a predictive model that encompasses both technical and community-related dimensions. Also, Eken et al. [23] investigate the effect of community smells on bug prediction, comparing their effect with code smells-related information. Moreover, Tamburri et al. [60] introduced an automated method for detecting community smells in open-source software development domains. They shed light on the prevalence of these smells and their perceived impact on the evolution of software communities. Also, Almarimi et al. [2] introduced a tool to identify community smells, starting from the repository on GitHub of software projects. Meanwhile, Huang et al. [38] study how to predict the community smells starting from the sentiments of individual developers, building a developer-oriented and sentiment-aware community smell prediction model. Additionally, Palomba et al. [50] emphasized the significance of socio-technical metrics, such as congruence and communicability, as effective predictors of the emergence of community smell. In a separate study, De Stefano et al. [19] delved into the relationship between community patterns, organizational or social structure types, and factors related to the quality of software products and processes in open-source repositories on GitHub. They employed association rule mining to identify common associations between community patterns and community smells, which are detrimental patterns within the organizational structure of software development communities, potentially leading to social debt.

Lambiase et al. [43] explored how cultural and geographical diversity influences the presence of community smells and their association with social debt in open-source software communities. Their study unveiled that these factors impact community smells, either exacerbating or alleviating certain issues. These insights offer valuable guidance for software development managers seeking to address organizational structures and related phenomena.

Catolino et al. [15] conducted research into how socio-technical factors impact the variability of community smells, with the aim to study exploring refactoring and mitigation strategies for community smells. Additionally, they examined the methods developers use to eliminate such smells, as described in their work [17]. Through a survey involving 76 experts, the authors were able to extract and consolidate a set of common refactoring techniques typically employed by practitioners

[2]ML-Ops Community: https://mlops.community/

to address the four community smell types identifiable through CodeFace4Smells, a software tool designed to identify and analyze code smells in software source code. Furthermore, Catolino et al. [16] made an additional contribution by demonstrating how the occurrence of community smells could potentially be reduced through increased gender diversity. Previous work primarily focused on understanding community smells, their identification, and their role in software systems. However, limited attention has been given to how these smells arise in ML-enabled systems or whether there are differences between the community smells identified in the literature and the socio-technical issues specific to the ML-enabled systems context. This is the main objective that our work aims to satisfy.

2.4 Main Contribution with Respect of the State of the Art

Although there are some works on socio-technical anti-patterns in ML-enabled systems [12, 44], no effort has been devoted to analyze how community smells occur in these systems. We believe this represents a missed opportunity.

Using "community smells" rather than "socio-technical anti-patterns" in research on human aspects in ML-enabled systems is crucial for several reasons. First, community smells offer a well-established framework to measure complex socio-technical phenomena such as communication and collaboration [15, 59]. This framework helps in understanding social debt, making research findings more practical for software practitioners [48, 50, 58]. Additionally, common terminology fosters the growth of knowledge, increasing research interest and consolidation in this area [17, 50]. Therefore, our work aims to in-deep the assumption provided by Nahar et al. [46] and study the relationship between the collaboration and communication issues in Teams involved in the development of ML-enabled systems [44], mapping them with the well-known community smells [13]. By doing so, we not only extend existing knowledge through interviews and advanced statistical methods but also provide practical theories for recommendation tools, making the findings accessible for practitioners and providing a foundation for future research [31, 54].

In summary, the contribution of this work is significant for several reasons. Firstly, community smells have become an established framework for characterizing socio-technical aspects of software development, offering a practical way to measure complex phenomena such as communication and collaboration. Investigating these aspects, also in the context of ML-enabled systems, also improves the understanding of social debt, making it more workable for practitioners. Our work builds on existing research, either by extending it through interviews that provide new insights, or by reinforcing it with statistical methods (e.g., PLS-SEM) to handle complex theoretical models. Finally, our contribution is practical and applicable while providing a solid basis for future research on community smells in ML-enabled systems.

3 OVERVIEW OF THE STUDY DESIGN

In the following section, we discuss and motivate the objective of our study as well as the research questions. Moreover, we introduce the research method adopted to address our questions.

3.1 Rationale and Motivation

As mentioned above, our work aims to provide a first step towards understanding community smells in the context of the ML-enabled systems by studying whether the socio-technical anti-patterns identified in this context may be causes, effects, or strategies for already known community smells. To achieve our goal, we chose to start from the already available literature and seek, through qualitative text-based analysis, to identify potential correspondences between socio-technical phenomena already documented in the study context, which may be causes, effects, or strategies for community smells [13, 44]. By positioning socio-technical phenomena identified in an ML-enabled

context as both the causes and effects of community smells discussed in the literature, we can effectively map these community smells within the ML-enabled context as well (further elaborated in Section 4). As a final step, we conducted a qualitative and quantitative study (through PLS-SEM) to develop evolved theories informed by new findings and assess their accuracy [31, 54].

The contribution proposed in this paper is worthy of attention for several reasons. First, community smells are increasingly an established way to characterize socio-technical aspects in the context of software development [15, 59]. From a practical standpoint, they provide a documented and unified method for trying to measure a complex phenomenon such as communication and collaboration. Beyond that, the literature on community smells allows for a better characterization of the concept of social debt, which is particularly important for the purposes of concretizing research into something usable in the context of practitioners [48, 50, 58]. Beyond that, and partially related to the above, the use of a common terminology within the research community can facilitate the growth of the body of knowledge on the topic with a consequent increase in interest (and thus research and knowledge) on the topic. In this sense, our contribution allows us to use what has already been done to (1) increase and improve knowledge and (2) consolidate it within a larger framework [17, 50]. In addition, our work does not simply take up what has already been done but (1) extends it concretely through a series of interviews (which led to new findings) and (2) reinforces it by means of a set of statistical methods (PLS-SEM) specifically designed to handle complex models of theory [54]. Last, our final contribution is concrete and transferable to the practitioner's field. By providing theories (in our case 5, one for each community smell, better discussed in the next sections), we figure out their integration into recommendation tools, making them available for practitioners in the computer science field [31, 54]. Moreover, other researchers can easily start from our work by extending our models (or taking inspiration from them) to increase the body of knowledge on the faceted field of community smells in an ML-enabled context.

3.2 Research Objective

The aim of this study is to delve deeper into understanding the phenomenon of community smells in the development of ML-enabled software systems. In order to achieve our goal, we start from the already documented literature on *Socio-Technical Anti-Pattern*—a bad practice that causes negative effects or problems that impact social and technical aspects [44]—in ML-enabled systems communities to identify possible causes, effects, and strategies that may be connected with community smells, expanding the already known phenomenon of community smells also in the ML-Enabled context.

The purpose is to offer fresh insights that enable practitioners to heighten their awareness of potential communication and collaboration issues within such diverse software development communities. This perspective is of interest to both researchers and practitioners. Researchers aim to comprehend how the different background within these communities influences the presence of community smells, while practitioners seek to understand and mitigate them. To address this objective, we formulated two research questions.

To achieve our goal, as a first step we decided to investigate whether there was an overlap between smells and socio-technical anti-patterns already documented in the literature. Since community smells are socio-technical phenomena themselves, our conjecture was that there might be an overlap between what has already been identified in the specific field of ML-enabled systems and the definitions of smells. Moreover, previous research [41] suggests that machine learning-enabled communities encounter communication barriers among their members. When AI developers skilled in data science collaborate with external stakeholders lacking similar expertise [51], it often leads to

challenges in communication and teamwork. These issues can pave the way for socio-technical anti-patterns. In our work, we aim to shed light on these challenges by using a tangible representation called "community smell" and analyzing how those are related to socio-technical anti-patterns.

***RQ$_1$**: What Socio-Techincal Anti-Patterns in the context of ML-Enabled Systems are Causes, Effects and Mitigation Strategies related to Community Smells?*

In our next steps, our focus turned to understanding the moderating factors that could influence the relationship between socio-technical patterns and community smells. Specifically, guided from previous literature on human factors in software development [1, 14, 16, 21, 45, 57, 59, 63], we hypothesized that there may be some aspects that deserve to be considered in our study as moderating factors, that is, aspects that could greatly influence our hypotheses. Specifically, we considered gender, roles within the community, and the adopted development model as moderating factors. Albusays et al. [1] highlight how gender disparities, role distribution, and the choice of development model contribute to imbalanced power dynamics, miscommunication, and exclusionary practices; it becomes clear that these factors are critical to understanding how community smells arise and persist in socio-technical systems. The study's results show that underrepresented groups often face barriers in accessing knowledge or decision-making roles, which can lead to the entrenchment of community smells like knowledge silos and bottlenecks [1].

Considering the gender, our choice was guided by earlier studies, like Catolino et al. [16], who highlighted how gender diversity correlate the emergence of community issues. Other studies have demonstrated that there is growing evidence that gender diversity affects software engineering teams' ability to collaborate, communicate, and approach problem-solving. Teams with greater diversity typically exhibit higher levels of creativity and inventiveness, which is especially important for machine learning (ML) projects that call for intricate decision-making and interdisciplinary knowledge [63]. This leads us to consider the potential impact of gender on the connection between community smells and socio-technical anti-patterns.

In terms of roles, research indicates that some roles, such as architects or team leaders, tend to centralize information and decision-making, which may result in anti-patterns like knowledge bottlenecks. Positions requiring greater collaboration, such as cross-functional team members or scrum masters, may these problems by improving team dynamics and communication [14]. Moreover, we acknowledge that diverse cultural backgrounds and varying skill sets come into play within different roles [59]. This diversity can significantly shape how communities operate in socio-technical terms.

Finally, we explore how the chosen Development Model could affect team communication and collaboration. For instance, methodologies like Scrum emphasize constant interaction and knowledge sharing among team members [45]. However, other studies have shown that in distributed Agile teams or large-scale projects, certain anti-patterns, such as coordination breakdowns and bottlenecks, may still arise due to challenges in scaling communication effectively [21]. In ML-enabled systems, Agile principles may sometimes clash with the experimental and iterative nature of machine learning workflows, where tasks like model training and data preparation do not fit neatly into short sprints [57]. This misalignment can further exacerbate socio-technical anti-patterns, particularly concerning task coordination and dependency management across development roles. Based on those observation, we aimed to understand how different Development Models might influence the relationship between community smells and socio-technical anti-patterns. This exploration sought to uncover how these models either facilitate or impede effective communication and collaboration, thus impacting the overall socio-technical dynamics within communities.

***RQ$_2$**: Given that Socio-Technical Anti-Patterns in the context of ML-Enabled Systems can be Causes, Effects, or Mitigation Strategies related to the already known Community Smells, can they be influenced by developers' gender and role and adopted development model?*

3.3 Overview of the Research Process

To answer our research questions, we adopted the research approach summarized in Figure 1. The approach is divided into two macro-steps. In the first one, we conducted a preliminary analysis—by means of a review of papers in literature—to gain knowledge about the domain under study and identify the specific factors to include in our investigation. In this phase, started from these two sources: the SLR on community smells and the literature on socio-technical aspects in ML-enabled systems. These papers provided us with catalogs of community smells and socio-technical anti-patterns. To find connections between these two concepts, we conducted a coding phase (as described in the Preliminary Analysis Section 4), which served as the basis for our hypotheses. In the second one, we adopted *Partial Least Squares Structural Equation Modeling (PLS-SEM)* [31, 54] to (1) develop our hypotheses on the relationships between socio-technical anti-patterns and community smells (our research questions) and (2) evaluate them. The following sections provide an overview of each phase of our research approach.

Our preliminary analysis aimed to gain knowledge about the domain under study by analyzing the pertinent and recent literature. Specifically, we referred to the literature on community smells and on socio-technical aspects of ML-enabled systems. Moreover, since we hypothesized that only some of the community smells and anti-patterns emerge in the ML-enabled context, such literature analysis also aimed to identify the factor on which to focus our research. Such an extraction was conducted through a qualitative analysis approach.

After identifying the socio-technical anti-patterns and community smells part of our investigation, we used *Partial Least Squares Structural Equation Modeling (PLS-SEM)* [31, 54] to (i) develop our hypotheses on the relationships between socio-technical anti-patterns and community smells (our research questions) and (ii) evaluate them.

(1) We used literature and an interview study to build a set of hypotheses on how the socio-technical anti-patterns are related to community smells in ML-enabled context.
(2) We then mapped the identified hypotheses on a directed graph where (i) the nodes represent socio-technical anti-patterns and community smells (which we will refer to from now forward by the term "**constructs**"), and (ii) the arcs represent the hypotheses that a relationship exist between socio-technical anti-patterns and community smells. For example, given an arc from node A to node B, it represents the hypotheses that construct A (e.g., a particular anti-pattern) might influence construct B (e.g., a community smell) to some extent. Such a graph is called in the PLS-SEM as a Structural Model. We built five models, one for each community smell identified in the preliminary phase.
(3) We identified a way to measure the constructs under investigation. PLS-SEM algorithm needs to measure the constructs to evaluate the relationships between them (i.e., our hypotheses). Specifically, following similar work in the context of software engineering and guidelines from the main text used to conduct our study, we decided to collect data through a questionnaire study with practitioners. We developed a set of Likert scale questions (called in PLS-SEM theory, indicators) associated with each construct in our model.
(4) We developed a survey based on the questions identified as indicators for our constructs and administered them by means of Prolific. Before it, we identified a good sample of participants through sampling strategies based on a set of criteria. As an output, we collected data usable from PLS-SEM to measure the construct and evaluate the hypotheses.

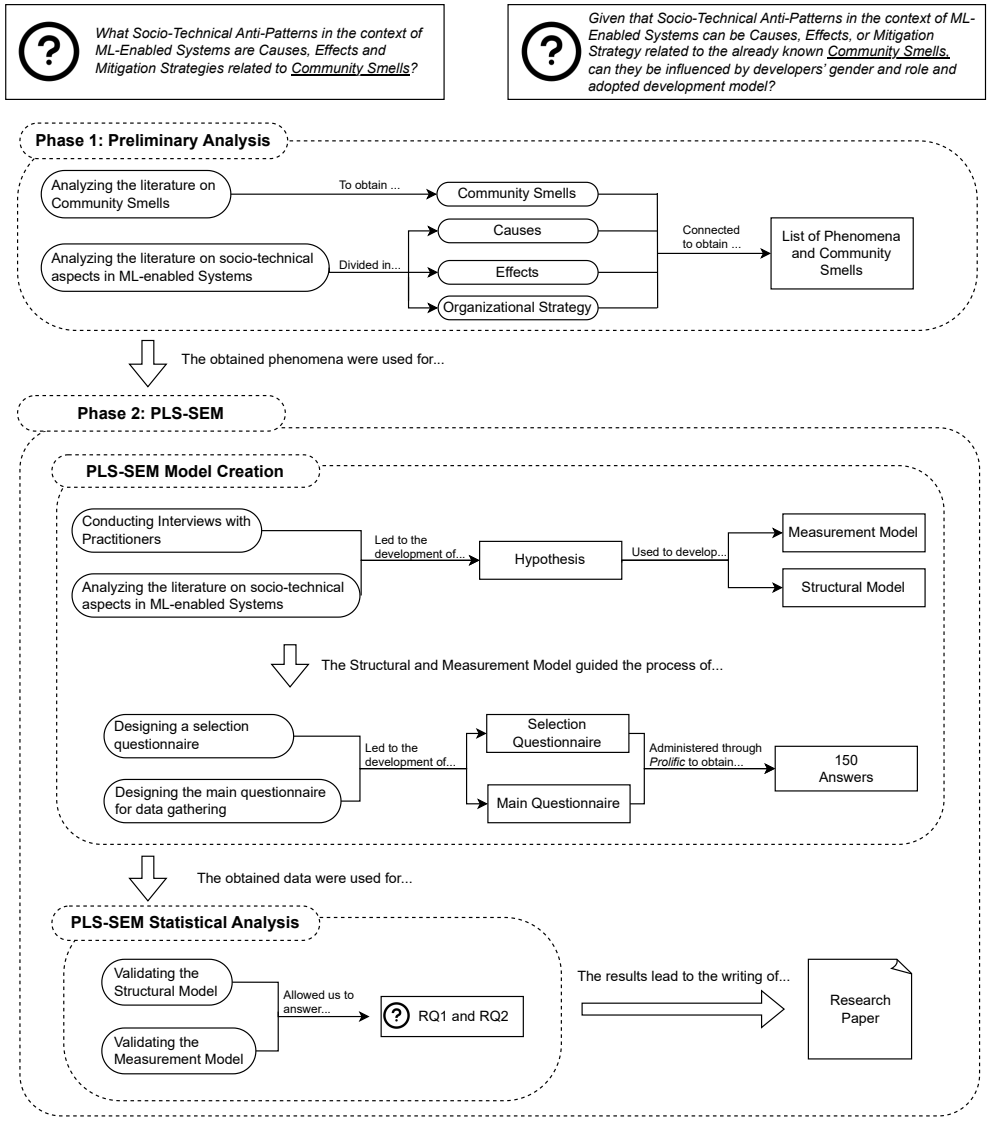


Fig. 1. Research Process

(5) We performed PLS-SEM on (i) our models and (ii) the collected data and, through validation, we answered our first reseaerch questions.
(6) We also conducted a Multi-group analysis to study whether the hypotheses' results varied according to the gender, role, and development model adopted. The data were divided into groups: For the first analysis, Males and Females; For the second analysis, Software engineers, Data Scientists, Data Engineers, and all the others roles of developers which

are merged in 'Others'; For the third analysis, Agile development model and 'Others' that merged all the other development models.

In the following, for each step conducted, the article first presents the research method and choices made, and after it followed by an illustration of the obtained results. All the data used in our study are available in our online appendix [4].

4 PRELIMINARY ANALYSIS

In the following, we introduce the basics of our preliminary analysis and associated findings.

4.1 Overall Approach

Our preliminary analysis aimed to gain a deeper understanding of the domain under study by reviewing recent and relevant literature on community smells and the socio-technical aspects of ML-enabled systems. The primary objective of this analysis was to identify the community smells to focus our research on and to construct our models.

To achieve this, we conducted a coding phase based on the catalog of smells provided in [13] and the socio-technical anti-patterns outlined in [44]. Using a qualitative analysis approach, we formulated initial hypotheses about how socio-technical anti-patterns are related to community smells within an ML-enabled context. The detailed methodology for this extraction and qualitative analysis is described in the following sections.

4.1.1 Analysis of Socio-technical Anti-Patterns in ML-Enabled Systems. Our initial step consisted of collecting relevant knowledge on socio-technical anti-patterns in the context of ML-enabled systems. Our focus was directed to a recent publication of Mailach and Siegmund [44]. Their work reported on 17 socio-technical anti-patterns (in ML-enabled systems), their potential causes, and recommendations. We conjectured that we could use such anti-patterns to identify possible causes, effects, and strategies for already known community smells in order to place them in the context of ML-enabled systems [13, 44]. Our goal is to understand community smells in ML-enabled systems by examining whether socio-technical anti-patterns in this context are causes, effects, or strategies for known community smells[15, 17, 50, 59]. Through qualitative and quantitative studies, we extend the existing literature, develop new theories, and provide practical recommendations for practitioners, facilitating the growth and consolidation of knowledge in this field [31, 54].

We decided to conduct a *deductive coding* [65] analysis step on the socio-technical anti-patterns description. More precisely, we conducted *structural coding* [65], i.e., a categorization process of text according to a specific structure, with a structure of three elements:

(1) the causes of the socio-technical anti-patterns;
(2) the effects of the socio-technical anti-patterns;
(3) the organizational strategy (or strategies) that could mitigate or counter the pattern's effects.

To identify the socio-technical anti-patterns, we conducted a coding phase. Figure 2 shows an example of how the coding phase was carried out to derive the causes and effects associated with a socio-technical anti-pattern. The descriptions of the socio-technical anti-patterns used during this phase were based on the work of Mailach and Siegmund [44]. Additionally, Figure 3 illustrates the overall coding process, demonstrating how we established the connections between causes, effects, organizational strategies, and community smells.

First and second authors of the paper conducted the coding process, starting from the description of all the Socio-Techinical Anti-Pattern identified by Mailach and Siegmund [44]. The coding process was conducted collaboratively due to the few number of sources analyzed. Specifically, both researchers were present during all coding sessions, with the tasks divided between them.

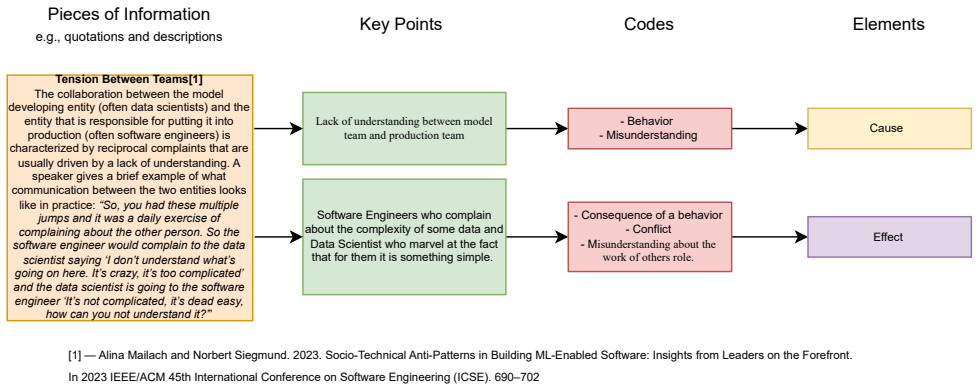


Fig. 2. Example of Structural Coding Process to extract Causes, Effects and Organizational Strategies from Socio-Technical Anti-Pattern in the context of ML-enabled Systems

Upon completing the initial coding, the researchers jointly reviewed and refined each other's work. Any discrepancies were resolved through in-person discussions, ensuring consistency and accuracy. Doing the deductive process, we aimed to obtain a deeper understanding of the phenomenon and organize the information in a way that makes it more suitable to (1) connect with community smells and (2) provide readable and understandable findings, both for practitioners and researchers.

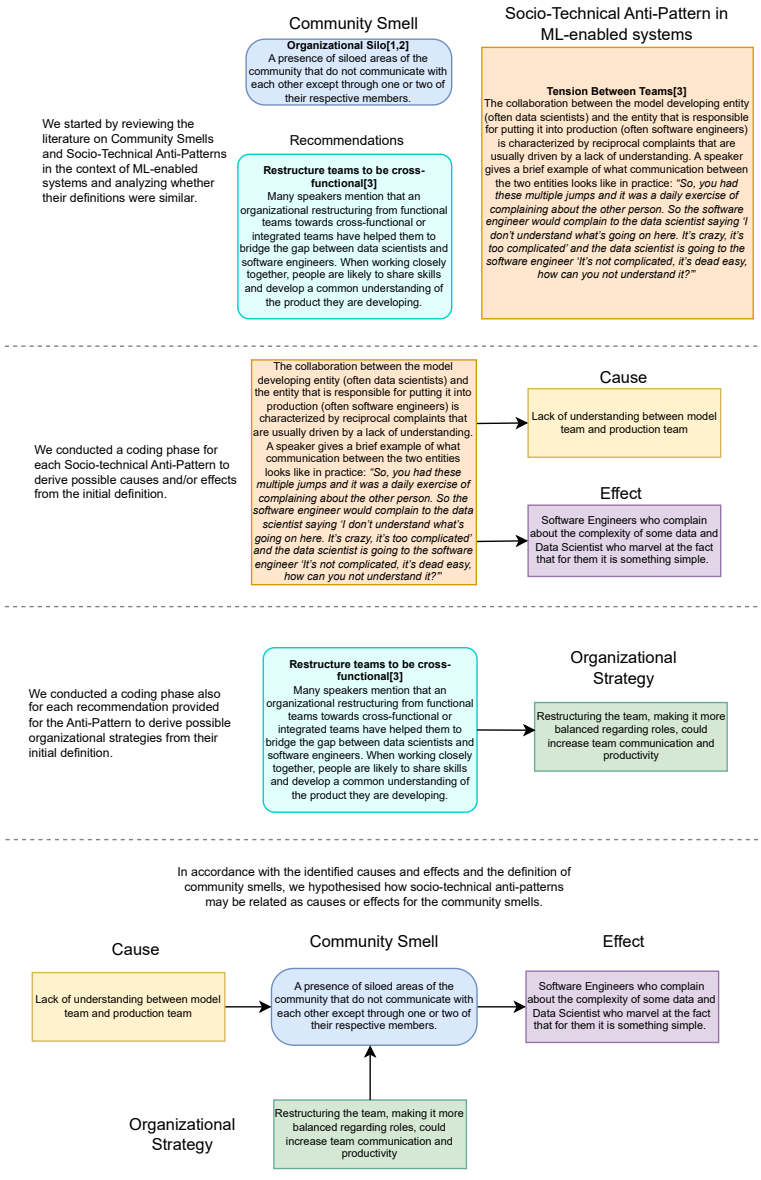
4.1.2 Community Smells. We started by collecting a catalog of community smells. To this purpose, we took advantage of the recent systematic literature review conducted by Caballero-Espinosa et al. [13], which cataloged and provided a description for all community smells identified until 2023. Starting from the community smells collected by Caballero-Espinosa et al. [13], we used their description to conduct a deductive coding process aimed at connecting the community smells with causes and effects identified in the previous step (the analysis of socio-technical anti-patterns) [44]. We aimed to both identify a potential set of smells to analyze in the next phase of the study and start developing the hypotheses that was the foundation for the PLS-SEM path modeling step.

4.2 Findings

From the (first) deductive coding step done on the socio-technical anti-patterns [44], we obtained 7 causes, 6 effects, and 3 organizational strategies linked between them. The full list of the obtained causes, effects, organizational strategies, and community smells is reported in the *Structural Coding Process* file in our online appendix [4].

In the next phase, we compared the description of the community smells [13] with the items obtained in the previous step. Such mapping resulted in a successful connection between the causes and effects obtained by the coding of the ML-enabled system's socio-technical aspects [44] and their organizational strategies, with the community smells [13]. Since the connection was still complex to represent and analyze, we focused our attention on the 5 smells with the most significant number of connections and with the description that fit best with that of the socio-technical anti patterns analysedx. These smells are:

- **Organizational Skirmish** indicates a scenario where teams have differences in their organizational cultures. It makes the work of project managers difficult. The impact is notorious on productivity, e.g., project delays [13, 59].



[1] — Eduardo Caballero-Espinosa, Jeffrey C Carver, and Kimberly Stowers. 2022. Community smells—The sources of social debt: A systematic literature review. Information and Software Technology (2022)
[2] — Damian A. Tamburri, Fabio Palomba, and Rick Kazman. 2021. Exploring Community Smells in Open-Source: An Automated Approach. IEEE Transactions on Software Engineering 47, 3 (March 2021), 630–652
[3] — Alina Mailach and Norbert Siegmund. 2023. Socio-Technical Anti-Patterns in Building ML-Enabled Software: Insights from Leaders on the Forefront. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). 690–702

Fig. 3. Example of how we conjecture the correlation between Community Smell and Cause and Effect from Socio-Technical Anti-Pattern in the context of ML-enabled Systems, after the coding phase

- **Organizational Silo** indicates the presence of siloed areas of the community that do not communicate with each other except through one or two of their respective members [13, 60].
- **Lone Wolf** is used to represent teammates carrying out their work irrespective or regardless of their peers. Reflects poor communication on the project need [13, 16, 60].
- **Prima Donna** indicates the presence of teammates working in isolation adopting superiority behaviour. They are unwilling to welcome the change of legacy products and support from other teammates. These teammates prevent the organization from innovative solutions or processes and effective communication and collaboration [2, 13, 59].
- **Black Cloud** appears when an organization does not provide the conditions for social interactions and effective communication between teammates. Thus, the conditions do not support the exchange of knowledge during software development processes, e.g., professional experience or understanding of projects in progress [13, 49, 59].

A complete list of the final set of phenomena—i.e., causes, effects, and organizational strategies—is reported in Table 1. Following a deductive approach, based on the definition of community smells and the causes, effects, and organizational strategies extracted by the coding, we linked the strategies and effects after connecting the causes to the community smells. In such a way, we treated community smells like a mediator effect. Moreover, we decided to separate the research data per community smells. In such a way, we obtained 5 groups of hypotheses (one for each smell) that were the input for the first phase of the PLS-SEM method described in the next section. In the folder "Hypothesis graphs" of the online appendix, it is possible to find the cause-and-effect graphs that led us to develop the first hypotheses, based only on the coding phase [4].

5 PLS-SEM STRUCTURAL MODEL DEFINITION

This section reports the steps performed to obtain the first part of a PLS-SEM path model, i.e., the Structural Model, as well as the model itself. The Structural Model represents the constructs forming the theory model and the relationships between them (in terms of prediction), while the last one describes how the constructs will be measured and represented through a set of indexes.

5.1 Structural Model Development Approach

The output of the preliminary phase (Section 4) was a set of hypotheses that describe the relations between community smells and socio-technical phenomena extracted from literature in ML-enabled systems. Nevertheless, this list of hypotheses needed to be refined and could not be used to create the structural model of PLS-SEM. For such a reason, following the guidelines used in this study [31], we conducted a further qualitative analysis. Specifically, we started from the hypotheses obtained in the previous step (which were elicited by literature), and we conducted a *semi-structured interview study* [18, 37] aimed at (1) strengthening and augmenting (by means of practitioners' confirmation) our hypotheses on the previously identified 5 community smells.

5.1.1 Interviews Design. We conducted semi-structured interviews [18, 37], an approach that blends predetermined questions (aimed at exploring the primary subject of the research) with open-ended inquiries (to uncover unanticipated information types). This qualitative approach is frequently employed to collect comprehensive insights, views, and personal narratives from participants, ensuring uniformity and comparability throughout the interviews. Considering the intricate nature of the research topic (namely, collaboration and communication patterns), semi-structured interviews were chosen as the most effective technique to obtain perspectives from experts with

Table 1. Last set of phenomena extracted from the work of Mailach and Siegmund [44].

ID	Name	Description	Type
TBT	Tension Between Teams	Lack of understanding between model team and production team.	Cause
NCI	Redundant ML infrastructure and tools	Team members do not understand the presence of a unified infrastructure provided by the company for communication.	Cause
SWI	Shadow IT	The company provides infrastructure, but team members tend not to use it to communicate little and through informal channels.	Cause
TBMD	Tension between management and data scientists	Lack of understanding between data scientists and management figures in charge of managing the team, such as the team leader.	Cause
CoCT	Clash of cultures and tools	Different teams, composed by different roles tend to use different tools, languages, and terminology.	Cause
MEC	Missing engineering competence	Data scientists and Software Engineers alike lack skills to understand the other side sufficiently.	Cause
MIBT	Missing intersections between teams	All teams work on different use cases, which creates low communication between them, and they don't know each other's developments.	Cause
LRC	Long Release Cycle	The product team needs to rewrite the whole model, so they start a long phase to resolve technical challenges	Effect
ETBT	Misunderstanding between different roles	Software Engineers don't understand the complexity of the work of Data Scientists and vice versa.	Effect
ENCI	Redundant Infrastructure	Teams lead to a redundant infrastructure and accompanying tools in the organization.	Effect
ESWI	Ignoring Organizational Standard	Team members tend to ignore the security and other standards provided by the organization.	Effect
ETBMD	Managerial frustration	Data scientists tend to focus more on research rather than on the needs of the business, causing frustration for those managing the project	Effect
SIS	Staff with insufficient skills	People are hired who have a different background than that required for the specific task to which he is assigned.	Effect
RTCF	Restructure team to be cross-functional	Restructuring the team, making it more balanced regarding roles, could increase team communication and productivity.	Strategy
PP	Pair data scientists and software engineers	Using pair programming can help developers increase their skills.	Strategy
TF	Introducing a mediator	A figure who can understand the technical language of different roles would improve communication and collaboration.	Strategy

significant experience in distributed software development. Moreover, such approaches were applied to constructing the hypotheses for the PLS-SEM methods, and they are (1) recommended by guidelines [31] and (2) used in similar studies.

To design the semi-structured interview protocol, we relied on the guidelines provided by Hove and Anda [37]. The structure of the interviews and their results are available in the online appendix [4]. The protocol was organized in 7 sections described following:

- **Start and icebreaker**: Both the interviewees and interviewers introduced themselves, providing an overview of their work, their research topic, and general activities to break the ice and put the interviewees at ease.
- **Question about experience and collaboration**: Questions about current and past work experiences, roles held during them, and current roles held. Interviewers ask about how many collaborations there are with developers in the same role and with developers in other roles.
- **Question about Team Working**: Questions concerning the interviewees' perception of the collaborations with developers in other roles and in the same role. Interviewers ask

if there are developers isolating or overpowering others and if there are collaboration or rivalry.
- **Question about Background and Skills**: Interviewers ask what perceive differences in background and skills with developers in other roles.
- **Question about Communication**: Questions about internal communication within the team and with other teams. If it is constant, interviewers ask what is the frequency, and if the organization provides special platforms.
- **Question on Strategies**: Interviewers ask the interviewees to propose strategies to improve the issues they told you about, e.g., communicative, collaborative, etc.
- **Conclusion**: If there were aspects of interest to explore, interviewers ask about them. Conclusion of the interview and thanks.

The first draft of the interview protocol was created by the first two authors of the papers and subsequently honed through discussions with the remaining authors. It was then tested for clarity and comprehension with two practitioners and two researchers from our network. According to the feedback obtained, a supplementary slide that outlined the questions was introduced to aid participants in maintaining focus.

5.1.2 Participants Sample Strategy. In selecting participants, we employed a convenience sampling strategy [7, 29], a non-random technique where participants are chosen based on their accessibility, readiness, or eagerness to participate in the study. Participation was entirely on a volunteer basis. Although this method facilitated rapid recruitment of subjects, it poses challenges to the broad applicability of our results [29]. To mitigate this concern, we intentionally targeted individuals who satisfied certain predefined conditions, which encompassed the following:

- **Role**: Software Engineer, Data Scientist, or Data Engineer.
- **Experience**: More than one year of work experience.
- **Collaboration**: Who collaborates or has collaborated in the past with a Software Engineer, Data Scientist, or Data Engineer.

Software Engineers, Data Engineers, and Data Scientists play distinct yet interrelated roles in ML-enabled systems [47]. Data Engineers manage data pipelines by transforming and optimizing data for both model training and operational use, ensuring its quality and accessibility [33]. Data Scientists focus on developing, testing, and refining ML models to meet specific problem requirements, leveraging advanced statistical and machine learning techniques. Software Engineers are responsible for building reliable and scalable software systems, ensuring the seamless integration of ML models into broader applications while addressing key aspects such as performance and security [12, 47]. While these roles differ in their focus—system functionality, data readiness, and model creation—they collaborate closely to align software, data, and ML components into cohesive and effective solutions [12, 33, 47].

We contacted 10 potential participants using a questionnaire sent by e-mail. In the questionnaire, we asked for confirmation to participate in the interviews and information to arrange the meeting. Moreover, we asked for registering the interview for data analysis purposes. Ultimately, we conducted interviews with 5 participants through online meetings of approximately 60 minutes. Although it may seem a small sample size, it was sufficient due to the challenges of recruiting participants and the depth of the topics and insights covered. The duration of each session allowed for rich and detailed data collection, which would not have been feasible with a larger sample. The online appendix contains interview transcripts and the most significant excerpts [4].

5.1.3 Interview Analysis. As mentioned in the previous sections, the interviews were intended to go to reinforce our hypotheses and possibly add new ones. For this reason, we opted once

again for a deductive approach, using the hypotheses already obtained as a basis and looking for information in the interview transcripts that would strengthen and/or weaken them. Beyond that, given the flexible nature of the semi-structured interviews on the text fragments not associated with the hypotheses already identified, an additional coding phase was conducted in order to identify potential causes, effects, or strategies to be associated with one of the 5 community smells considered in the study [27].

This process was conducted by the first author of the article and further refined through meetings with the second and third authors. Finally, the final hypotheses model obtained was jointly evaluated by all authors of the article. The final outputs were (1) the list of hypotheses (divided by the 5 community smells), (2) some quotations from the interviews supporting the hypotheses, and (3) the graph of the structural model (based on the hypotheses) input to the next stage.

5.2 Structural Model Development—Results

We conjectured hypotheses to create our Structural Models that collate community smells to cause, effect, and organizational strategies. Starting from the phenomena identified in the preliminary analysis, we formulated 25 hypotheses based on the literature and the definitions extrapolated from it [13, 16, 44, 48, 49, 58–60]; after conjectured the initial connections, we refined and comforted it with interviews with practitioners in the ML-enabled systems field. The hypotheses obtained are summarized in Table 2 and following described.

Organizational Skirmish — The literature highlights that software engineers and data scientists tend to have different backgrounds; in particular, according to their backgrounds, they adopt different tools, languages, and technical terminology in their works. Those differences lead to a lack of understanding during the collaboration, in which, on one side, software engineers struggle to understand the complexity of data science models, and on the other side, data scientists believe that the models are easy to understand [44]. Since then, Organizational Skirmish has been defined as a scenario in which teams have differences over their organizational cultures. [59], we hypnotize that the different backgrounds, understanding, and knowledge between data scientists and software engineering can be related to the occurrence of Organizational Skirmish.

Starting from our hypothesis, we refined it with the interviews. For example, P_4 said, "*Software Engineers come from a pure computer science background, unlike Data Scientists with different backgrounds, such as a math degree*". Moreover, P_3 believed that "*In the teams, everyone has the patience to explain some concepts if someone doesn't know something*". In Addition, P_1 said "*Usually, Data Scientists explain to colleagues how the models work and how they are trained; in pair programming, Software Engineers give feedback on the code and how to improve maintainability*". Starting from that information, we conjecture the following hypotheses:

Hypothesis 1.1 ($H_{1.1}$)—The missing engineering competence has a positive association with Organizational Skirmish.
Hypothesis 1.2 ($H_{1.2}$)—Different cultures, terms, and tools have a positive association with Organizational Skirmish.
Hypothesis 1.3 ($H_{1.3}$)—Lack of understanding between teams has a positive association with Organizational Skirmish.

Mailach and Siegmund [44], in their study, observe that restructuring the team brings benefits such as information and knowledge exchange. Among the most adopted practices to foster information exchanges between developers with different backgrounds is pair programming or the introduction of intermediary developers who share different backgrounds. Based on those observations, we conjecture that those recommendations may be an organizational strategy to mitigate the occurrence of Organizational Skirmish, which are defined as a scenario where teams have

differences over their organizational cultures. It makes the work of project managers difficult [13]. We strengthen our conjecture with the interview; P_2 said "*Even if the manager does not require it, team members conduct pair programming to share information and knowledge and increase the quality of the code*". On the other hand, P_2 affirmed "*Pair programming is applied in our company mainly between data engineers and people in the same role or software engineers, somewhat less often with data scientists*". Starting from those quotes, we define the hypotheses:

Hypothesis 1.4 ($H_{1.4}$)—Restructure teams to be cross-functional has a reverse association with Organizational Skirmish.
Hypothesis 1.5 ($H_{1.5}$)—Pair data science and software engineers have a reverse association with Organizational Skirmish.
Hypothesis 1.6 ($H_{1.6}$)—Translation work between the different roles and common understanding of the goal has a reverse association with Organizational Skirmish.

Moreover, Mailach and Siegmund [44] also observed that when developers with different backgrounds are involved, it happens that someone is assigned to a task but does not have the appropriate skills to perform it. They accept tasks unaware of how complex they may be. This may result in a problem with the task's realization, leading to subsequent refactoring of the product by other developers and a delay in delivery. Since that Tamburri et al. [59] and Caballero-Espinosa et al. [13] highlight that the cultural difference identified in the Organizational Skirmish smell, impact the productivity and collaboration of the team, leading to project delays, we suppose that the observation of Mailach and Siegmund [44] may be consequence of the occurrence of Organizational Skirmish smell. In Addition to those observation, during the interview, P_4 affirmed "*Sometimes, the difference in the background is the cause of problems in communication and collaboration, maybe because of the technologies or technical language used. This problem requires cross-functional figures to interact with the two roles in large contexts*". Further, P_1 stated "*Data Engineers and Software Engineers have the same background in that a data engineer is a software engineer who has chosen to devote to engineering the data and the model.*". P_5 agrees with this quote and adds "*Data scientists come from completely different roots, like mathematicians, so I notice differences in background between them*". Based on those quotes, we conjecture:
Hypothesis 1.7 ($H_{1.7}$)—Organizational Skirmish has a positive association with the creation of a long release cycle.
Hypothesis 1.8 ($H_{1.8}$)—Organizational Skirmish has a positive association with having a staff with insufficient skills.
Hypothesis 1.9 ($H_{1.9}$)—Organizational Skirmish has a positive association with a misunderstanding about other roles' work.

Organizational Silo—Tamburri et al. [60] define the Organizational Silo as a presence of siloed areas of the community that do not communicate with each other except through one or two of their respective members. Based on that definition provided, we consider the siloed areas of the community as the different sub-teams that come into being during the development of an ML-Enabled product, each characterized by programmers of the same role. In particular, Mailach and Siegmund [44] observed that developer with different role, e.g., Data Scientist and Software Engineers, tend to work independently on different aspects and tasks. The communication between the two roles is ineffective and leads to misunderstandings, especially regarding the products produced. In addition, regarding the experience of communication and collaboration between data scientist and software engineers, the interviewer P_4 stated "*Communication between different roles occurs only in cases of necessity and due to tasks assigned; usually, each sub-team works on its tasks, and we check what is done once a week*". From that quote and the literature [13, 44, 60], we define:

Hypothesis 2.1 ($H_{2.1}$)—Missing intersection between teams has a positive association with Organizational Silo.

Hypothesis 2.2 ($H_{2.2}$)—Lack of understanding between teams has a positive association with Organizational Silo.

Mailach and Siegmund [44] investigated the best recommendation to improve collaboration and communication between data scientists and software engineers. They highlight that adopting pair programming and conducting a turnover in the team can foster the exchange of information and knowledge, improving collaboration and communication between the different roles. We assume that such solutions can also be adopted to reduce the effects of the Organisational Silo smell, which holds that the team tends to split into sub-teams [13, 60]. Moreover, our assumption is reinforced by interviewer, in fact P_5 said, " *Our company adopts pair programming between different figures, for example, data scientists and machine learning engineers; this is to foster knowledge exchange*". Meanwhile, P_4 affirmed "*Even if the manager does not require it, team members conduct pair programming to share information and knowledge and increase the quality of the code*". Starting from the literature and from the interview, we conjecture the following hypotheses:

Hypothesis 2.3 ($H_{2.3}$)—Restructure teams to be cross-functional has a reverse association with Organizational Silo.

Hypothesis 2.4 ($H_{2.3}$)—Pair data science and software engineers have a reverse association with Organizational Silo.

In their study Mailach and Siegmund [44] observed that a developer able to cover different roles could be a connection point for sub-teams composed of data scientists and software engineers, respectively. We believe that this recommendation may also be adopted to mitigate the occurrence of the Organizational Silo smell since that it describes a situation in which the team is divided into sub-team that tend to have a lack of collaboration and communication [13]. Morover, according to our interviewers, P_1 stated "*Usually in the teams there are different roles such as Data Scientists and Engineers, MLOps Engineers, and Software Engineers. However, they are divided into teams with the same role to increase the focus on a specific task. They work together only when needed*", also P_2, and P_3 agree with those quote. From this, we define the following hypotheses:

Hypothesis 2.5 ($H_{2.4}$)—Translation work between the different roles and common understanding of the goal has a reverse association with Organizational Silo.

Further, P_1 added "*Data Scientists and Software Engineers, who are divided into different teams, usually tend to work together only when needed*". In particular, working separately also affects the exchange of information and knowledge. This lack leads to a misunderstanding of others' work, where data scientists do not understand the need to apply engineering standards to improve their code, and software engineers do not understand the models build by data scientists [44]. From this observations, we define the following hypotheses:

Hypothesis 2.6 ($H_{2.5}$)—Organizational Silo has a positive association with a misunderstanding about other roles' work.

Lone Wolf—Catolino et al. [16] and Tamburri et al. [60] define the Lone Wolf as a social anti-pattern where *Teammates carry out their work irrespective or regardless of their peers. Reflects poor communication on the project needs.* Starting from those definition, we conjecture that those behaviors may depend on the role and skills of the developer, in which software engineers or data scientists don't want to adopt a collaborative behavior with others, causing problems in the communication between developers and technical debts on the project development Mailach and Siegmund. Managerial figures, such as project managers, may have difficulty understanding the rationale behind such behavior and may not be prepared to deal with it, communicate with such

team members, or resolve technical problems that may occur [48]. However, the adoption of strategy such as refactoring the team, may reduce the creation of sub-teams and its consequences [44]. We investigated to our conjecture further in the interviews. In particular, P_1 believes "*Communication and collaboration have been very fluid, and it's worked nicely. It may depend on the people in the team and by how the manager involves us in activities*". Drawing information from these quotes and taking into account the existing literature that underscores the complexities of managing a team afflicted by these particular social anti-patterns, we formulate the following hypotheses:

Hypothesis 3.1 ($H_{3.1}$)—Unclear communication between managers and data scientists has a positive association with Lone Wolf.

Hypothesis 3.2 ($H_{3.2}$)—Restructure teams to be cross-functional has a reverse association with Lone Wolf.

According to Mailach and Siegmund [44] data scientists tend to work independently. This behavior can affect team collaboration, and it may be a problem for the manager who has to manage the team and interface with them. Moreover, P_3 explains "*We are few in the team and mostly data scientists, and we tend to each work on our tasks*", reinforcing our conjecture that:

Hypothesis 3.3 ($H_{3.3}$)—Lone Wolf has a positive association with Tension between management and data scientists.

Prima Donna—Tamburri et al. [59] define the Prima Donna as a social anti-pattern where *Presence of teammates working in isolation. They are unwilling to welcome the change of legacy products and support from other teammates. These teammates prevent the organization from innovative solutions or processes and effective communication and collaboration.* Starting from the definition of Prima Donna Effects, we assume that such superiority and uncooperative behavior depend on the role and skills of the developers. More precisely, software engineers or data scientists tend to adopt superiority behavior over the other roles, focusing only on their own goals without considering business and team ones. Such behavior is also extended toward managerial roles, who find it difficult to interact, manage, and collaborate with such developers [44]. Morover, we conducted interviews to delve deeper into manifestations and implications of Prima Donna Effects, obtaining that P_2 said "*Sometimes it happens that when there are some problems, a single data scientist decides to take charge of it and solve it themself. Problems such as to tackle stability issues and performance issues in the environment*". According to the quote and literature we conjecture:

Hypothesis 4.1 ($H_{4.1}$)—Unclear communication between managers and data scientists has a positive association with Prima Donna.

Hypothesis 4.3 ($H_{4.3}$)—Prima Donna has a positive association with Tension between management and data scientists.

In Addition, P_4 express that "*Data Scientists tend to see the system in their breadth and completeness and move away from classic software development*". So, we suppose that a possible mitigation to the occurrence of such type of behavior by a team member is to conduct a turnover among team members so that they can collaborate with different people, assimilate, and share knowledge and skills [13, 44, 59]. Based on insight we conjecture:

Hypothesis 4.2 ($H_{4.2}$)—Restructure teams to be cross-functional has a reverse association with Prima Donna.

Black Cloud—Palomba et al. [49] and Tamburri et al. [59] define the Black Cloud as an anti-pattern where *Organizations do not provide the conditions for social interactions and effective communication between teammates. Thus, the conditions do not support the exchange of knowledge during software development processes, e.g., professional experience or understanding of projects in progress.*

Based on this definition, we conjecture our hypotheses that possible reasons behind the occurrence of the Black Cloud smell might be the fact that industries don't provide a communication infrastructure and developers prefer to use informal communications, which they believe are faster and more efficient. [44].

Furthermore, in support and augmentation of them, our interviewee P_1 stated "*The company has not officially provided us with tools for communication; mostly, we use email, classic messaging tools, or video calls*", also P_3 and P_5 agree to that quote. We based the following hypotheses on the literature and the interview:

Hypothesis 5.1 ($H_{5.1}$)—No clear and unified infrastructure in the organization has a positive association with Black Cloud.

Hypothesis 5.2 ($H_{5.2}$)—Low formal communication or information transfer has a positive association with Black Cloud.

In addition, in the case in which the industries provide formal, unified infrastructures, developers sometimes ignore them because they prefer to use informal communications, which they believe are faster and more efficient. The adoption of informal communication channels can lead to the redundancy of communication channels and the dispersion of relevant data and information [44]. Moreover, to confirm this observation, P_3 stated, "*Although a formal communication tool is used in the company, we tend to deal with exchanging information informally sometimes to communicate more quickly*". This quote was supported also by P_5, and makes us conjecture the following hypotheses:

Hypothesis 5.3 ($H_{5.3}$)—Black Cloud has a positive association with leads to a redundant infrastructure and accompanying tools in the organization.

Hypothesis 5.4 ($H_{5.4}$)—Black Cloud has a positive association with ignoring organizational standards of communication and collaboration.

Figure 4 show the structural models of the Organizational Skirmish, Organizational Silo, Lone Wolf, Prima Donna and Black Cloud.

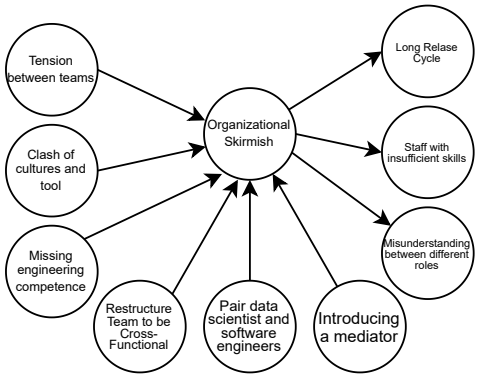
6 MEASUREMENT MODEL DEFINITION

In the following section we are going to expose the the measurement model that characterizes how our constructs (identified in the structural model) are measured by PLS-SEM. Specifically, PLS-SEM distinguishes between two types of constructs. The first are *latent variables*, i.e., constructs that cannot be measured directly, which need to identify different indicators (in PLS-SEM they are called "indexes") to make the measurement. The second are the *non-latent variables* that can be measured directly through a single measurement index.

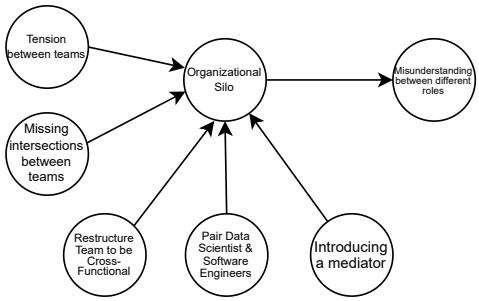
6.1 Measurement Model Development—How we carried it out

To delineate the indices integral to our model, we adopted an inductive methodology complemented by reflective measurement. For the data collection, we decided to use a questionnaire, a method prevalent in similar research, particularly within the software engineering domain [40]. This strategy needed formulating questions (the indices) designed to elicit responses that reflectively indicate the associated construct's influence; in other words, each construct is represented (and measured) by a series of questions acting as indices.

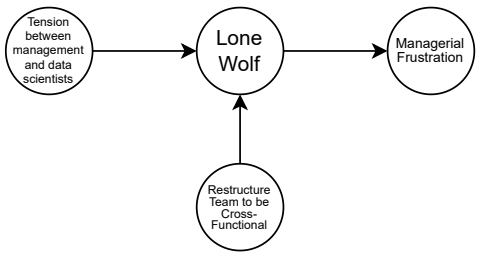
We relied on the literature and data collected through the previously cited interviews to identify the indexes. We represented community smells as latent variables and identified more than two questions as indexes for each of the five. The novelty of community compelled this decision smells as research constructs for which established measurement frameworks are yet to be devised. Regarding the other phenomena (i.e., causes, effects, and organizational strategies), their high level of specificity allowed for direct measurement through single-item queries. Despite the general consensus in



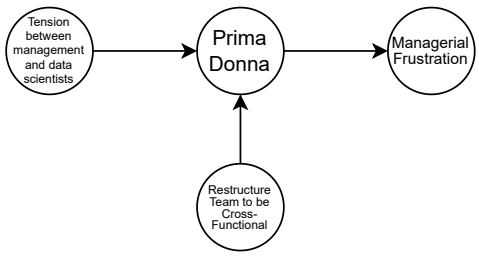
(a) Structural Model of Organizational Skirmish



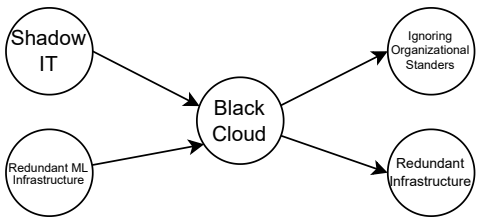
(b) Structural Model of Organizational Silo



(c) Structural Model of Lone Wolf



(d) Structural Model of Prima Donna



(e) Structural Model of Black Cloud

Fig. 4. Structural Models of the five community smells.

academic literature against this approach, it is crucial to acknowledge that the simplistic nature of these phenomena justifies this method.

The definition of the indexes was initially conducted by the first authors. Then, the second and third authors jointly discussed to improve them. Based on the initial set of identified indices, the final determination of the variable types (latent and non-latent) and the selection of items used to measure each variable were made through a thorough and collaborative discussion among all the authors. The authors, who are experts in the study of community smells and the associated metrics of these social patterns, carefully evaluated each index. The expertise of the authors ensured that the variables accurately reflected the theoretical underpinnings of community smells.

Table 2. Hypothesis Overview.

Hypothesis	
Organizational Skirmish	
$\mathbf{H_{1.1}}$	Missing engineering competence ⇀ Organizational Skirmish
$\mathbf{H_{1.2}}$	Clash of cultures and tools ⇀ Organizational Skirmish
$\mathbf{H_{1.3}}$	Tension Between Teams ⇀ Organizational Skirmish
$\mathbf{H_{1.4}}$	Restructure team to be cross-functional ↽ Organizational Skirmish
$\mathbf{H_{1.5}}$	Pair data scientist and software engineers ↽ Organizational Skirmish
$\mathbf{H_{1.6}}$	Introducing a mediator ↽ Organizational Skirmish
$\mathbf{H_{1.7}}$	Organizational Skirmish ⇀ Long Release Cycle
$\mathbf{H_{1.8}}$	Organizational Skirmish ⇀ Staff with insufficient skills
$\mathbf{H_{1.9}}$	Organizational Skirmish ⇀ Misunderstanding between different roles
Organizational Silo	
$\mathbf{H_{2.1}}$	Missing intersections between teams ⇀ Organizational Silo
$\mathbf{H_{2.2}}$	Tension Between Teams ⇀ Organizational Silo
$\mathbf{H_{2.3}}$	Restructure team to be cross-functional ↽ Organizational Silo
$\mathbf{H_{2.4}}$	Pair data scientist and software engineers ↽ Organizational Silo
$\mathbf{H_{2.5}}$	Introducing a mediator ↽ Organizational Silo
$\mathbf{H_{2.6}}$	Organizational Silo ⇀ Misunderstanding between different roles
Lone Wolf	
$\mathbf{H_{3.1}}$	Tension between management and data scientists ⇀ Lone Wolf
$\mathbf{H_{3.2}}$	Restructure team to be cross-functional ↽ Lone Wolf
$\mathbf{H_{3.3}}$	Lone Wolf ⇀ Managerial frustration
Prima Donna	
$\mathbf{H_{4.1}}$	Tension between management and data scientists ⇀ Prima Donna
$\mathbf{H_{4.2}}$	Restructure team to be cross-functional ↽ Prima Donna
$\mathbf{H_{4.3}}$	Prima Donna ⇀ Managerial frustration
Black Cloud	
$\mathbf{H_{5.1}}$	Redundant ML infrastructure and tools ⇀ Black Cloud
$\mathbf{H_{5.2}}$	Shadow IT ⇀ Black Cloud
$\mathbf{H_{5.3}}$	Black Cloud ⇀ Redundant Infrastructure
$\mathbf{H_{5.4}}$	Black Cloud ⇀ Ignoring Organizational Standards

⇀ : positive association ↽ : reverse association

6.2 Measurament Model Definition—Indexes

In the following, we report the choices for each construct of our five models and the rationale. At the end of the section, we report the list of indexes (i.e., the Likert Scale questions) associated with each construct and the path model complete of both the measurement and structural model.

All variables in our model were measured directly or through a set of measurable items; Figure 5 reports the complete Measurement models of the Organizational Skirmish, Organizational Silo, Lone Wolf, Prima Donna and Black Cloud. Each one is mapped with a question of our questionnaire, which will are evaluated using a **Likert Scale**. The Likert Scale question often consists of 5 points question and is used to measure attitudes, opinions, or perceptions in a quantitative way. This type of question is the most commonly used in this type of study and is widely recommended by important general guidelines [31, 61]. Our question are ranging from 1 (“Strongly Disagree”) to 5

Table 3. Questions about the Community Smells.

Construct	Question	ID
Community Smells		
Organizational Skirmish	Different teams, each one with a different role, actively collaborate and work together on shared project ideas	OS1
Organizational Skirmish	Different teams, each one with a different role, often face difficulties in effectively communicating and exchanging information	OS2
Organizational Skirmish	Often conflicts and disagreements between different teams are always smoothly resolved	OS3
Organizational Skirmish	Often, the resources for the different teams, each with a different role, are allocated fairly and based on objective criteria	OS4
Organizational Skirmish	Power struggles or conflicts related to authority or decision-making influence among different teams are often created	OS5
Organizational Skirmish	Different teams work on different use cases and tend to communicate only with the member of their team	OS6
Organizational Silo	Team members from different teams communicate and collaborate with each other	OSE1
Organizational Silo	Different teams proactively share relevant information and updates with others	OSE2
Organizational Silo	Different teams work together on projects or tasks	OSE3
Organizational Silo	Team members from different teams come together for meetings or discussions	OSE4
Organizational Silo	Different teams give priority to the goals in different ways	OSE5
Lone Wolf	There were some team members who tended to work in isolation	LW1
Lone Wolf	There were some team members who tended to work disrespectfully toward their peers	LW2
Lone Wolf	There were some team members who tended to not communicate about their tasks.	LW3
Prima Donna	There were some team members who tended to work in isolation	PD1
Prima Donna	There were some team members who were unwilling to welcome the change of legacy products and support from other teammates	PD2
Prima Donna	There were some team members who prevented the organization from innovative solutions or processes and effective communication and collaboration	PD3
Black Cloud	I experienced a situation where the team members engaged in informal communication to exchange knowledge during software development	BC1
Black Cloud	I experienced a situation in which team members had the opportunity to collaborate and exchange knowledge during software development	BC2
Black Cloud	I experienced a situation where there is a moment of "knowledge sharing" between people of different cultures during software development	BC3
Black Cloud	I experienced a situation in which team members had access to communication technologies that support knowledge exchange during software development	BC4

("Strongly Agree"). The complete list of questions (i.e., indexes) is reported in Tables 3, 4, 5, and 6. In the next section, we provide details on the questionnaire and its administering.

6.2.1 Community Smells. The community smells are the starting point of our study and the core of our analysis. As variables, we have identified only the community smells that are most evident and most involved in the relationships identified in the preliminary analysis, Section 4. Those variables are described in Section 5.2 and are identified as latent variables measured by different measurable items. The Community Smells are shown in Table 3, with their relevant items represented by the ID, and the question from the questionnaire that provided us with their measurement.

6.2.2 Phenomena—Causes. Starting from the preliminary analysis phase conducted, socio-technical anti-patterns were decomposed into causes and effects, according to their definition. In Table 4 we report the variables identified as **causes**, starting from socio-technical anti-pattern definition and the causes identified by Mailach and Siegmund [44], the relevant items represented by the ID, and the question from the questionnaire that provided us with this measurement.

6.2.3 Phenomena—Effects. Starting from the preliminary analysis phase conducted, socio-technical anti-patterns were decomposed, according to their definition, into cause and effect. In Table 5 we report the variables identified as **effect**, starting from the socio-technical anti-pattern definition identified by Mailach and Siegmund [44], the relevant items represented by the ID, and the question from the questionnaire that provided us with this measurement.

Table 4. Questions about the Causes.

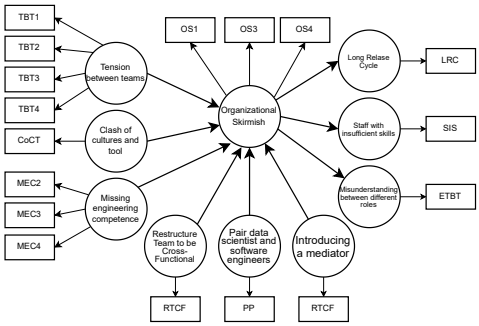
Construct	Question	ID
Causes		
Tension Between Teams	I experienced a situation in which team members misunderstood each other's instructions or messages about what to do	TBT1
Tension Between Teams	I experienced a situation in the definitions of the functional requirement of the system to realize, provided by the product team, were unclear for the others teams	TBT2
Tension Between Teams	I experienced a situation in which team members received contradictory information or instructions about the task to do	TBT3
Tension Between Teams	I experienced a situation in which Data Scientists and Software Engineers have different interpretations of the same information or instructions about the task to do	TBT4
Redundant ML infrastructure and tools	I experienced a situation where is unclear if the company provides specific tools to communicate and collaborate with colleagues	NCI
Shadow IT	I experienced a situation in which the team member used communication channels different from the ones provided by the company	SWI1
Shadow IT	I experienced a situation where the teams engaged in formal communication channels (e.g., meetings, emails) to exchange information	SWI2
Shadow IT	I often experienced a situation where the company applies established protocols to facilitate information sharing among teams	SWI3
Shadow IT	I experienced a situation where the model team's functional requirements were clear and complete for the other teams	SWI4
Tension between management and data science	I experienced a situation where the model team's functional requirements were clear and complete for the other teams	TBMD
Clash of cultures and tools	I experienced a situation in which different groups involved in the same project, each with different roles, use different tools, languages, and terminology	CoCT
Missing engineering competence	I experienced a situation in which people from different teams communicated with each other to ask for clarification about their tasks and work from a colleague who was more experienced or had different competencies	MEC1
Missing engineering competence	I experienced a situation in which team members with a specific role (for example, Software Engineers) easily understood the technical terms and language used by other teams with different roles (for example, Data Scientists and Data Engineers)	MEC2
Missing engineering competence	I experienced a situation where team members knew the processes followed and the status of task of other teams	MEC3
Missing engineering competence	I experienced a situation in which the team members of different teams collaborated closely with each other	MEC4
Missing intersections between teams, documentation, and trust	I experience a situation in which different teams work on different use cases and tend to communicate only with the member of their team	MIBT

Table 5. Questions about the Effects.

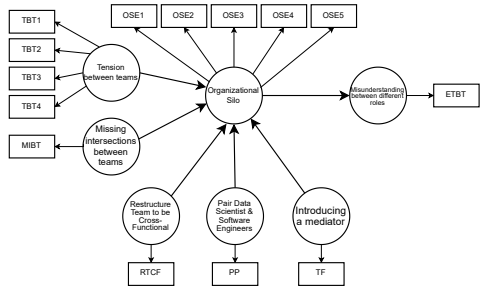
Construct	Question	ID
Effects		
Long Release Cycle	The teams in charge of the model have to rewrite it, creating long intermittent transitions between versions and technical challenges (e.g., ensuring the same properties as the original model)	LRC
Misunderstanding between different roles	I experienced a situation where some simple concepts for Data Scientists can be complex for Software Engineers, and vice versa, and they may complain about the complexity	ETBT
Redundant Infrastructure	I experienced a situation where the organization provided different communication and collaboration tools without a unified one	ENCI
Ignoring Organizational Standard	I experienced a situation where organizational standards, e.g., communication and collaboration, are ignored	ESWI
Managerial frustration	I experienced a situation where Data Scientists/Engineers did not want to be managed by whoever manages the project (Project manager/ Scrum Master /Product Owner)	ETBMD1
Managerial frustration	I experienced a situation where Data Scientists/Engineers did not want to be managed by whoever manages the project (Project manager/ Scrum Master /Product Owner)	ETBMD2
Staff with insufficient skills	I experienced a situation where people with skills for a specific task (for example, a Data Scientist specializing in machine learning) have been assigned to a different task, out of their skills (for example, building and maintaining pipelines where engineering skills are required, a task usually given to a Software Engineer)	SIS

Table 6. Questions about the Organizational Strategies.

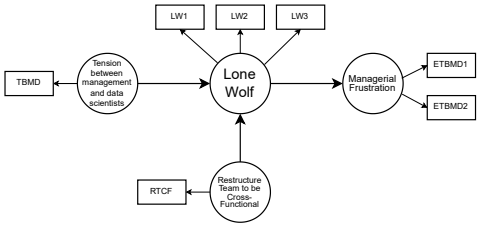
Construct	Question	ID
Organizational Strategies		
Restructure teams to be cross-functional	An organizational restructuring having different roles working closely together would improve collaboration and communication between Data Scientists/Engineers and Software Engineers	RTCF
Pair data scientist and software engineers	The use of pairing and code review helps Data Scientists/Engineers to write quality code that adheres to Software Engineers standards	PP
Introducing a mediator	Having a common language among the different roles often allows the products to be understood as a shared responsibility and fosters empathy for the individual challenges of each field	TF



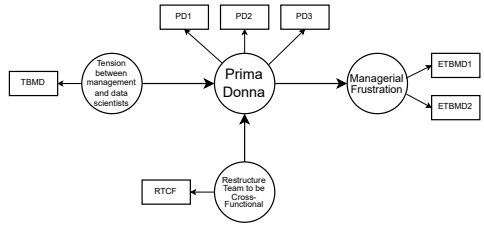
(a) Measurement Model of Organizational Skirmish



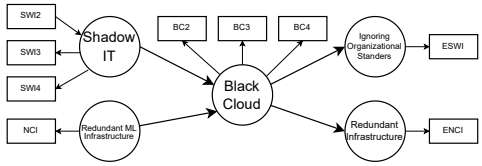
(b) Measurement Model of Organizational Silo



(c) Measurement Model of Lone Wolf



(d) Measurement Model of Prima Donna



(e) Measurement Model of Black Cloud

Fig. 5. Measurement Model of the five community smells.

6.2.4 Phenomena—Organizational Strategies. Mailach and Siegmund [44] identified a set of organizational strategies for the socio-technical anti-pattern in ML-enabled systems. We select a subset of them that could be in a cause-and-effect relationship with the community smells examined. The

organizational strategy are reported in Table 6, with the relevant items represented by the ID, and the question from the questionnaire that provided us with this measurement.

7 DATA COLLECTION—SURVEY

In the preceding sections, we outlined the process of identifying the hypotheses for constructing the structural model. Subsequently, we delineated the corresponding measurement model for each variable. In this section, we describe the data collection stage for model measurement on how we analyze those data to test the hypotheses.

7.1 Survey Design and Procedure

We designed a survey, similar to most studies involving PLS-SEM, to collect the data for our study. Specifically, each question in the survey aimed at collecting data (from a Likert scale) to characterize each index identified in the measurement model definition step. By combining the various data for each index, the PLS-SEM algorithm is able to compute statistics that can allow us to answer our research question. Concretely, we:

(1) used **G*Power** [24] to identify the ideal size of our population of respondents;
(2) developed a screening survey to identify the target population for our study;
(3) developed a survey in which each question corresponds to one of the indexes described in the previous section;
(4) administered the questionnaires and collected data;
(5) conducted preliminary data analysis for reliability reasons.

In the development and dissemination of our survey, we adhered to a variety of guidelines to enhance our methodology and subsequently, our results. Initially, for the survey's design, we utilized the guidelines set forth by Kitchenham and Pfleeger [42], along with Andrews et al. [3], which are widely recognized in software engineering research. Furthermore, we employed the scale and guidelines suggested by Kitchenham and Pfleeger to enhance our survey's clarity and according to similar studies using PLS-SEM. Moreover, in line with Flanigan et al. [25] advice, we ensured the survey remained anonymous to avoid influencing participants' responses. The survey was created using a Google form and was designed to be completed within 10 to 15 minutes.

Concerning the distribution, we relied on Prolific[3] to recruit experts. Prolific is a web-based platform to support researchers in finding participants for survey studies. The platform allows tuning the preference of surveyed, putting constraints, i.e., people must be practitioners and experienced in distributed work. Prolific use an *opt-in* strategy [39]: this implies that participants get voluntarily involved, possibly leading to self-selection or voluntary response bias [34]. In terms of guidelines, we were guided by existing literature [22, 53]. We particularly followed the recommendations by Reid et al. [53], which outlined strategies for executing surveys in the software engineering domain using this platform.

7.1.1 Participant Selection and Selection Survey. As mentioned before, we first aimed to identify our target population. Despite the fact that Prolific offers a well-designed set of criteria and filters for participants screening, we decided to conduct a preliminary selection survey on the platform.

First of all, we used the statistic tool **G*Power** [24]. The tool allowed us to obtain the ideal sample size for our statistical investigation by specifying the desired effect size, statistical power, and number of hypotheses. Since our investigation regards 5 path models, we selected as a number of hypotheses to insert in the tool to obtain the minimum value of sample to satisfy, the number of hypotheses of the structural model with the most hypotheses, i.e., the Organizational Skirmish

[3]Prolific website: https://www.prolific.co/.

model with 9 hypotheses. To obtain our sample size, we set G*Power using an F-test with multiple linear regression, using an a priori test to calculate the required sample size with an effect size (0.15), a significance level (α) of 0.05, and a default value for power (1 - β) of 0.95. We adopted those settings based on the value indicated by Hair Jr et al. and adopted in other studies [31, 61]. Considering that we relied on the model with the largest number of phenomena to observe (Organizational skirmish), the number of predictors was set equal to 6. We thus obtained a total sample size of 146 items. In the end, we obtained an ideal sample size of 146 participants. For this reason, we administered the selection survey to a large population (391 participants) and selected the 150 participants that we identified as more adherent to our ideal selection criteria.

Before conducting the questionnaire, we conducted a selection survey to be shared on Prolific basis to obtain a representative sample of the population. This selective survey consisted of 18 questions and a focus question. The questions were divided into four sections: (1) personal information (e.g., role and background), (2) information about the team and company, (3) information about collaboration and communication, and (4) information about team culture and deployment.

We administered the questionnaire to developers on Prolific, obtaining 391 responses. Among them, we focused on selecting possible candidates who represented our sample of the ideal population. The selection criteria adopted were:

- Participants who work in the roles of software/data engineer, data scientist and similar roles;
- Participants who has an experience of more than one year;
- Participants who collaborated with a software engineer, data engineer, or data scientist.

We selected developers with at least one year of experience because we specifically wanted to include newcomers to the field, while also recognizing the value of their academic background as experience. Moreover, research has shown that new developers can significantly contribute to team dynamics and encounter key socio-technical challenges, including communication and coordination issues, within their first year of professional or academic work [9].

Applying these selection criteria, we obtained about 160 possible participants in our study.

7.1.2 Main Survey Structure. After figuring out the target population for the study, we moved on to design the main survey. The ultimate goal of this survey was to get the data we needed to work out latent variables and run the PLS-SEM algorithm. So, keeping in line with what other similar studies did, our survey mostly had questions that matched up with the indicators we found when we were building our measurement model. As mentioned before, we opted for Likert Scale questions using scales from 1 (strongly disagree) to 5 (strongly agree). Moreover, to ascertain the answers' correctness, we included some "attention checks" in the survey; we removed all incorrect answers and obtained 150 valid responses. The survey comprises 50 questions, incorporating 3 attention-check queries, designed to ensure participants' sustained engagement. The questions are categorized into four sections: (1) Communication and infrastructure; (2) Skills and background; (3) Collaboration and behaviors; and (4) Organizational strategies.

The first three sections provide a scenario in which the participant is asked to imagine themselves within a heterogeneous team with two main teams. The first works on the data (i.e., the team members are mainly Data Scientists or Data Engineers), while the second works on the product (i.e., the team members are mainly Software Engineers). Based on their experience, participants are asked to rate their level of agreement with phrases related to communication and collaboration activities within their work team and among different sub-teams. The fourth section provides a scenario describing a problem of communication, misunderstanding, and collaboration between the product team (i.e., the team members are mainly Software Engineers) and the data team (i.e., the team members are mainly Data Scientists or Data Engineers). Based on their experience, participants are

asked to rate their level of agreement with phrases related to solutions to be implemented in such cases, to apply to solve that problem in the team.

An example of answers are: In the following, rate the level of agreement about some team members' collaboration and communication behaviors. Give a value from "Strongly Disagree" to "Strongly Agree" of each situation you experienced where *(1) Some team members who were unwilling to welcome the change of legacy products and support from other teammates, (2) Some team members who tended to work in isolation* .

7.2 Survey Preliminary Analysis

The survey described in the previous paragraph was administered to 160 participants, representing our sample. Once the survey results were collected, we filtered the responses in accordance with the attention questions, removing the responses of participants who answered incorrectly, resulting in 150 responses. We conducted a demographic analysis of the participants to extract the information needed to answer our second research question, which aims to understand if the relationship between community smells and socio-technical anti-pattern may change according to the gender, role and development model adopted. In particular, we can observe that the distribution of participants reflects a gender imbalance. However, this distribution is aligned with the broader trends observed in the computer science field, where males tend to be over-represented. The gender gap in this field has been well-documented, with men being a larger proportion. Therefore, the sample's gender distribution can be considered representative of the typical workforce in this sector, ensuring the validity of the survey results despite the gender imbalance [16]. Focusing on the role of the developers, the distribution between Software Engineers and Data Scientist is balanced. Slightly lower but still frequent is the role of Data Engineers. A very small part is covered by developers, who take different roles such as data analyst, project manager, technical consultant, etc. [59]. Since that we focused on Data Scientists, Software Engineers and Data Engineers, as they cover most of the sample, and we merged all the other roles in the category *Others*. Finally, we focused on the type of development model adopted. The majority of participants stated that they use Agile, while a fraction stated that they use hybrid models or different types. Since all other development models represented a very small percentage compared to Agile, they were all grouped into the Others category [45] (summary in Table 7).

Table 7. Demographics of Participants

Attribute	Size	Percetangle
Gender		
Male	106	70.6%
Female	44	29.4%
Role		
Software Engineer	48	32%
Data Scientist	41	27.4%
Data Engineer	36	24%
Others	25	16.6%
Development Model		
Agile Model	93	62%
Others	57	38%

8 PLS-SEM STATISTICAL ANALYSIS—MEASUREMENT AND STRUCTURAL MODELS EVALUATION

In the previous sections, we collected data by distributing a survey on Prolific. Each survey question was associated with a measurable item of the measurement models. In this section, we describe the analysis of the structural and measurement models and explain the results.

8.1 Survey Data Analysis

After collecting the data by using a questionnaire on Prolific, we aimed to analyze the correctness of the measurement and structural models. The measurement models evaluation aims to assess the quality of the measurement framework developed and identify problems in the indexes used to measure the constructs. The structural models evaluations is the evaluation of the identified hypotheses. To perform the evaluation, we used **SmartPLS 4**,[4] a software tool used for partial least squares structural equation modeling (PLS-SEM). It assesses measurement and structural models and evaluates complex relationships among latent variables. We used SmartPLS 4 to conduct the Statistical Analysis of the PLS-SEM method.

To evaluate the significance of our models and answer the research questions of the study, we followed the evaluation protocol by Hair Jr et al. [31]. The analysis consisted of three steps:

(1) First, to ensure that the data set is adequate—sufficient correlations among variables, indicating that the dataset is structured in a way that allows for latent factors or constructs to emerge from the data—, we conducted **Bartlett's test** on all constructs, where the p-value threshold is less than 0.05, to indicate that factor analysis is valid [8]. We obtained a p-value less than 0.01112, respecting the recommended threshold. Afterward, we conducted the **Kaiser-Meyer-Olkin (KMO)** measure of sampling adequacy, to evaluate how well-suited the data is for factor analysis, where the recommendation threshold is 0.60 [28]. Our results for each model are higher than 0.7, better than the recommended threshold. Bartlett's test and KMO test are valuable for assessing the adequacy of the dataset before moving into the modeling phase of PLS-SEM. Different studies adopted these combination of tests [5, 61]. We can say that the **data set and the selected population sample are adequate for our analysis**; the results are shown in Table 8.

(2) We evaluated the *Measurement Model* to ensure the variables' correctness and the reliability and validity of the measurement instruments used in a study (Section 8.2). To assess the measurement model's robustness, reliability, and validity, we calculated the **index Reliability**, which ensures each index's precision as a reliable measure of its latent variable. **Internal Consistency Reliability** assessment guarantees that a set of indexes within the same latent construct demonstrates consistency, enhancing overall reliability. **Convergent Validity** evaluation verifies that diverse indexes measuring the same latent construct converge appropriately, reinforcing the accuracy of the measured concept. **Discriminant Validity** assessment ensures independence among distinct latent constructs, adding credibility to the model's ability to differentiate between unrelated concepts.

(3) In the next step, we evaluated the hypotheses composing our structural model to understand the relations between the different variables (Section 8.3.1). This process allowed us to answer the first research question of the study. Structural model evaluation allows empirically testing the hypotheses derived from theory or literature. To achieve a more robust and trustworthy understanding of the underlying relationships among latent variables, we calculated the **Collinearity**, and assessing the significance of relationships in structural model evaluation is essential for ensuring the reliability and interpretability of

[4]SmartPLS 4 https://www.smartpls.com

the model. Simultaneously, we analyzed the **Significance of Relationships** for confirming hypothesized associations, providing statistical evidence supporting the structural model's validity.

(4) As the final step, we conducted a multi-group analysis to observe the heterogeneity of the developed structural model (Section 8.3.2). Such a multi-group analysis consisted of the comparison of the same model evaluated on different datasets differing for a moderator variable. To answer our second research question, we considered moderator variables such as the biological sex of the participants, the role currently held, and the development model currently adopted. Specifically, we diveded the answers from the participants of the study in two groups for the gender,[5] four groups for the role, and two groups for the development model adopted. For each set of groups, we conducted the evaluation of the model with the different portions of the dataset and compared the results to identify insight regarding the heterogeneity of our model.

All material about the analysis of the results is available in the online appendix [4].

Table 8. Bartlett and Kaiser-Meyer-Olkin (KMO) Tests

Model	Bartlett	KMO
Organizational Skirmish	3.281e-07	0.75
Organizational Silo	3.807e-13	0.71
Lone Wolf	0.004233	0.7
Prima Donnas	0.01112	0.73
Black Cloud	2.582e-09	0.74

8.2 Evaluation of the Measurement Model—Results

The first step is to examine the 5 models by evaluating each model's variables and the associated indexes; we wanted to understand their reliability, whether they fully represent the phenomenon, and whether multiple variables represent the same phenomenon [31].

Index Reliability. Our goal was to examine the relevance of each index associated with a latent variable, therefore, calculating it by the size of the **outer loading**, also called index Reliability [31]. This value thus indicates the relationship between the index and the construct it represents. A high outer loading suggests a strong association between the index and the construct. To ensure we had relevant indexes in all models, they must be not lower than the threshold of 0.7. However, in cases where the index value was between 0.4 and 0.7, Internal Consistency Reliability and Validity were analyzed, with and without that index. If the model without a specific index is more statistically significant than the model with it, that index would be permanently removed from the measurement model; otherwise, the index could remain within the model [30].

This procedure was performed for all latent variables in the 5 models. We removed from them all the items with an outer loading of less than 0.4. All items between 0.4 and 0.7 were taken into the analysis. They did not vary the Internal Consistency Reliability and Validity values, so it was chosen not to remove these items from the model. According to the results obtained in Model 1 (Organizational Skirmisk), we removed the items OS2 and OS5 from the respective latent variables.

[5]In our study, the classification of gender was intended to reflect biological sex and was constrained by the capabilities of the data collection tool (PROLIFIC) we used, which only permitted the specification of two sexes: male and female. The choice to use only two variables for representing gender was not intended to overlook or diminish the complexity of gender identity but was a practical decision based on the specific objectives of our study and the technological constraints we encountered. Our approach aimed to ensure consistency and clarity in data collection, allowing us to focus on the biological aspects relevant to our research questions within the boundaries set by the available tools.

Moreover, in Model 5 (Black Cloud), we removed the items SWI1 and BC1 by the respective latent variables. All outer loading results are shown in Figures 6, 7, 8, 9, and 10, represented as *values* between items (squares) and latent variables (circles).

Internal Consistency Reliability. We aimed to see how well the indexes are consistent with each other and how they can reliably and consistently measure constructs, in other words, the Internal Consistency Reliability of the variables. To calculate it, we used the **Cronbach's alpha** measure, which estimates the reliability based on the inter-correlation of the observed index variables. However, Cronbach's alpha is sensitive to the number of items in the scale and tends to underestimate the internal Consistency Reliability. A more appropriate measure to apply is **Composite Reliability**, which is based on the different outer loadings of the index variables [30]. Both Cronbach's alpha and Composite Reliability vary between 0 and 1. with higher values indicating higher reliability. The results show that all models have an acceptable value of Cronbach's alpha, around 0.55 and above, and a value of Composite Reliability, around 0.7 and above [6]. For more details, see Table 9.

Convergent Validity. We aimed to obtain the Convergent validity, in other words, the measure's degree of positive correlation with alternative measures of the same constructs. A common measure to establish this convergent validity is with the **Average Variance Extracted (AVE)**, which uses the sum of the squared loadings and divides it by the numbers of the indexes [30]. A recommended threshold of 0.50 indicates that the construct expresses more than half of the variance of its indexes. The results show that for all 5 models, we had a value of AVE up to 0.5 [6]. More details in Table 9.

Table 9. Internal Consistency Reliability with Cronbach's alpha and Composite reliabilimty (CR) & Convergent Validity with Average Variance Extracted (AVE) Results.

	Cronbach's alpha	CR	AVE
Organizational Skirmish			
OS	0.599	0.784	0.549
TBT	0.733	0.833	0.558
Organizational Silo			
OSE	0.774	0.844	0.523
TBT	0.742	0.834	0.558
Lone Wolf			
LW	0.734	0.844	0.643
ETBMD	0.631	0.842	0.728
Prima Donna			
PD	0.714	0.839	0.636
ETBDM	0.631	0.841	0.727
Black Cloud			
BC	0.647	0.806	0.588
SWI	0.595	0.784	0.550

Discriminant Validity. We aimed to calculate the extent to which each construct is distinct from others, to be sure that each construct captures a phenomenon not represented by others. One of the most used methods to calculate it is the **Heterotrait-monotrait ratio (HTMT)**, which is the ratio between-trait correlations to the within-trait correlation. A higher value means that the constructs are similar or observe similar phenomena; otherwise, with a lower value, we have that the constructs are independent of each other [35].

The results show that for all variables in the 5 models, there is a Discriminant Validity of less than 0.45. Therefore, all variables represent unique phenomena. The only exception is for the variables

Shadow IT (SWI) and Black Cloud (BC), which have a Discriminant Validity of 0.827; which is very close to 1. This indicates that the two variables represent concepts that, although different, turn out to be similar to each other. The results are shown in Tables 10, 11, 12, 13, and 14.

Table 10. Discriminant Validity - Organizational Skirmish.

	CoCT	ETBT	LRC	MEC	OS	PP	RTCF	SIS	TBT
ETBT	0.292								
LRC	0.299	0.122							
MEC	0.153	0.146	0.011						
OS	0.090	0.185	0.063	0.657					
PP	0.050	0.056	0.113	0.183	0.271				
RTCF	0.225	0.163	0.160	0.157	0.168	0.423			
SIS	0.029	0.068	0.241	0.028	0.210	0.075	0.011		
TBT	0.334	0.174	0.440	0.246	0.575	0.129	0.114	0.329	
TF	0.010	0.154	0.170	0.180	0.346	0.255	0.301	0.064	0.195

Table 11. Discriminant Validity - Organizational Silo.

	ETBT	MIBT	OSE	PP	RTCF	TBT
MIBT	0.146					
OSE	0.191	0.414				
PP	0.056	0.183	0.300			
RTCF	0.163	0.157	0.164	0.423		
TBT	0.174	0.246	0.310	0.129	0.114	
TF	0.154	0.180	0.392	0.255	0.301	0.195

Table 12. Discriminant Validity - Lone Wolf.

	ETBMD	LW	RTCF
LW	0.440		
RTCF	0.187	0.094	
TBMD	0.428	0.418	0.021

Table 13. Discriminant Validity - Prima Donna.

	ETBMD	PD	RTCF
PD	0.499		
RTCF	0.187	0.128	
TBMD	0.428	0.434	0.021

» Summary of the Measurement Model Evaluation

The results of the measurement models tell us that (1) the reliability of the *indexes* is confirmed, so the items we chose represent the Latent Variables. (2) The *Latent Variables* are consistent; the items can reliably and consistently measure the constructs. (3) The variables in the models represent *different phenomena* and do not overlap with each other.

Table 14. Discriminant Validity - Black Cloud.

	BC	ENCI	ESWI	NCI
ENCI	0.296			
ESWI	0.180	0.429		
NCI	0.226	0.444	0.437	
SWI	0.827	0.421	0.251	0.258

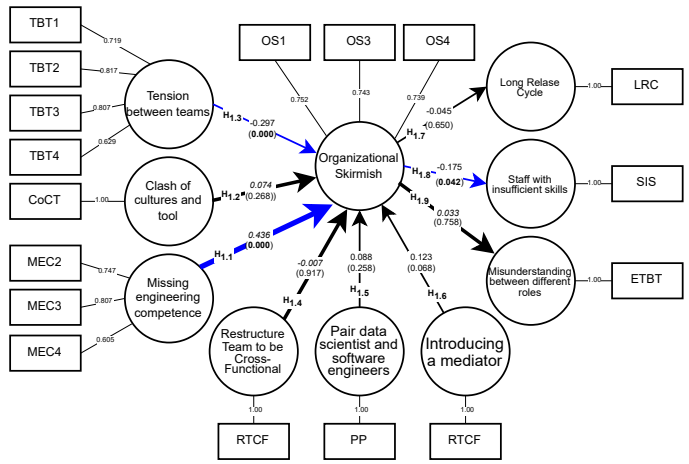


Fig. 6. Model 1 - Organizational Skirmish

8.3 Evaluation of the Structural Model—Results

In this section, we report the results of the structure model, which answers RQ_1, analyzing the reactions among the variables and studying the hypotheses, in section 8.3.1. Then, we examined the same hypotheses using a Multi-Groups Analysis to RQ_2—section 8.3.2.

All the results are shown in the figures 6, 7, 8, 9, and 10. In the spheres, there are all the variables of the models; in the rectangles, there are the measurable items through which each latent variable will be measured. The value of the outer loadings is written in the arcs between item and variable, representing the relevance of each index associated with a latent variable. The arrows represent the hypotheses and show the path coefficient (it is the above number, if it is congruent with our hypothesis, e.g., positive or negative, it is represented in *Italic* and by a bold arrow) and, the below number in parentheses, the p-value (if it is significant it is represented in **bold** and by a blue arrow). Figure 6 represents the Organizational Skirmish, Figure 7 represents the Organizational Silo; the Figure 8 represents the Lone Wolf; the Figure 9 represents the Prima Donna, and the Figure 10 represents the Black Cloud.

8.3.1 RQ_1—Analysis and Results.

Collinearity. To assess the Collinearity of the structural model, we examined each group of predictor constructs separately for each subpart of the structural model. Thus, calculating Collinearity with the **Variance Inflation Factor (VIF)** has a recommended threshold value that should be

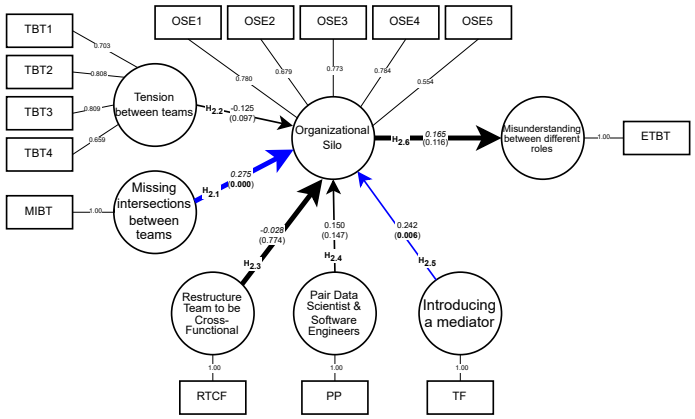


Fig. 7. Model 2 - Organizational Silo

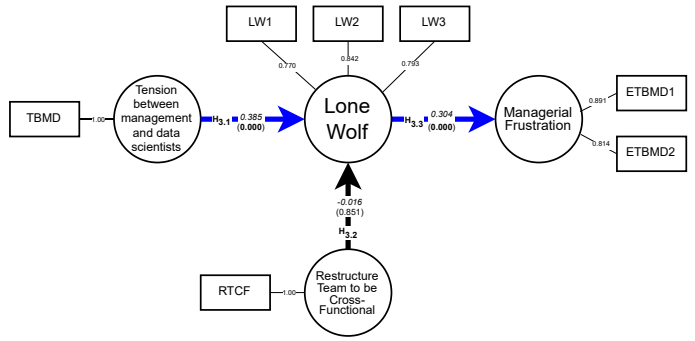


Fig. 8. Model 3 - Lone Wolf

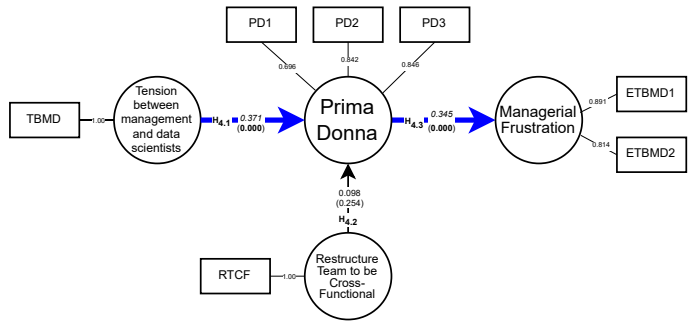


Fig. 9. Model 4 - Prima Donna

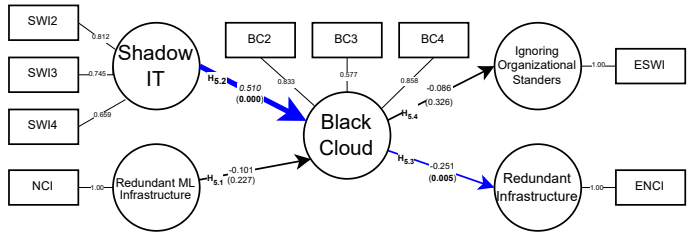


Fig. 10. Model 5 - Black Cloud

below 3 [30]. All variables in our model report have a VIF value of less than 1.5, meeting the recommended threshold. More details can be found in the online appendix [4].

Table 15. Collinearity assessment with VIF value.

Organizational Skirmish		Organizational Silo		Lone Wolf		Prima Donna		Black Cloud	
	VIF		VIF		VIF		VIF		VIF
CoCT	1.000	ETBT	1.000	ETBMD1	1.270	ETBMD1	1.270	BC2	1.525
ETBT	1.000	MEC	1.000	ETBMD2	1.270	ETBMD2	1.270	BC3	1.139
LRC	1.000	OSE1	1.669	LW1	1.556	PD1	1.268	BC4	1.435
MEC	1.000	OSE2	1.439	LW2	1.312	PD2	1.630	ENCI	1.000
OS1	1.218	OSE3	1.600	LW3	1.637	PD3	1.502	ESWI	1.000
OS3	1.192	OSE4	1.678	RTCF	1.000	RTCF	1.000	NCI	1.000
OS4	1.189	OSE5	1.149	TBMD	1.000	TBMD	1.000	SWI2	1.207
PP	1.000	RTCF	1.000					SWI3	1.250
RTCF	1.000	TBT1	1.343					SWI4	1.155
TF	1.000	TBT2	1.554						
SIS	1.000	TBT3	1.628						
TBT1	1.343	TBT4	1.265						
TBT2	1.554	TF	1.000						
TBT3	1.628	PP	1.000						
TBT4	1.265								

Significant Relationships. PLS does not make strong assumptions about the data distribution (e.g., a normal distribution), so parametric significance tests should not be used. However, a package in PLS uses a bootstrapping procedure to assess whether the path coefficients are statistically significant. It involves extracting a large number of random subsamples with replacement, usually around five thousand. Substitution is necessary to ensure that all subsamples have the same number of observations as the original data set. The PLS path model is estimated for each subsample. A standard error can be determined from the resulting bootstrap distribution, which can later be used to make statistical inferences.

The Path Coefficient represents the *hypothesized relationship* among constructs and has a standardized value approximately between -1 and +1; a value close to 1 represents a strong positive correlation, and vice versa, close to -1, a strong negative correlation. A coefficient closer to 0 means a weak correlation [31, 32]. Table 16 shows the results of the analysis. The table shows the Path Coefficient (PC), the standard deviation (SD), the p-value, and the Confidence Intervals (CI), which represent the range into which the population parameter will fall, assuming a certain level of confidence (e.g., 95%).

We explain how to interpret the results by detailing their reading for Hypothesis $H_{1.1}$, which sees Missing Engineering Competence (MEC) and Organizational Skirmish (OS) as related. The path coefficient of **0.436** indicates a **positive and moderately strong relationship** between MEC and OS. Missing Engineering Competence increases, there is a corresponding increase in the likelihood or intensity of Organizational Skirmish. In practical terms, the coefficient suggests that for every

unit increase in MEC, OS increases by approximately **43.6%**. The standard deviation (**SD**) of **0.069** reflects the variability of the path coefficient estimate. A low standard deviation suggests that the estimate is precise, and the relationship between MEC and OS is measured with consistency across the data. The **95% confidence interval (CI)** indicates that we can be 95% confident that the true value of the path coefficient lies between **0.288** and **0.559**. Since the entire interval is above zero, this further supports the conclusion that the relationship between MEC and OS is positive and statistically significant. Finally, the **p-value** of **0.000** means that the relationship between MEC and OS is statistically significant at any conventional significance level (e.g., 0.01, 0.05).

Looking at the results obtained, we can see that not all hypotheses turn out to be significant.

- For the community smell of **Organisational Skirmish**, we note a significance for hypotheses $H_{1.1}$, $H_{1.3}$, and $H_{1.7}$. This indicates that background disparity and misunderstanding of others' work may be causes of Organisational Skirmish smells, which can lead to a team that is not skilled enough for the tasks at hand. Very close to the significance value is hypothesis $H_{1.6}$, which indicates that the introduction of a team member with a common background mitigates the effects of the organizational skirmish smell.
- For the community smell **Organisational Silo**, we see that the significant hypotheses are $H_{2.1}$ and $H_{2.5}$, which show us that when software engineers or data scientists focus only on their tasks without engaging with the rest of the team, they tend to create sub-groups that are not very communicative and collaborative with each other. This situation could be mitigated by introducing developers into the team who are able to communicate, collaborate, and work in both roles.
- For the community smells **Lone Wolf** and **Prima Donna Effects** we can note that hypothesis $H_{3.1}$ and $H_{3.3}$ and assumptions $H_{4.1}$ and $H_{4.3}$ are significant. These hypotheses show us that Data Scientists tend not to communicate with the manager, work alore and independently from other roles, and/or adopt an attitude of superiority. This behavior causes problems and frustration for the manager.
- For the community smell **Black Cloud**, we note that the significant assumptions are $H_{5.2}$ and $H_{5.3}$. These hypotheses suggest that sometimes team members ignore the communication infrastructure provided by companies, creating their own private communication channels. This leads to redundancies in channels and communication and a loss of information.

Additional details regarding the discussion on the supported and unsupported hypotheses are provided in Section 9.1.

» Summary of the Results of Structural Model - RQ_1

The results of the structural models tell us:

- Some hypotheses are unsupported, revealing a potential mismatch between findings in the literature and practitioners' perceptions. For instance, communication breakdowns—widely recognized in the literature as critical factors—may not be directly linked to background or team composition, as hypothesized. Instead, they could be influenced by individual traits, informal resolution practices, or the inherent dynamics within a team.
- Poor or misunderstood communication between managers and developers who manage the product tends to isolate the latter, who focus more on their interests than the company's, creating frustration with managers.
- The company's infrastructure and communicative standards tend to be ignored. Developers employ their informal communication, making it redundant.

Table 16. Significant Relationships.

	PC	SD	95% CI	p-value
Organizational Skirmish				
$\mathbf{H_{1.1}}$ Missing engineering competence → Organizational Skirmish	*0.436*	0.069	(0.288, 0.559)	**0.000**
$\mathbf{H_{1.2}}$ Clash of cultures and tools → Organizational Skirmish	*0.074*	0.067	(-0.059, 0.205)	0.268
$\mathbf{H_{1.3}}$ Tension Between Teams → Organizational Skirmish	-0.297	0.073	(-0.452, -0.162)	**0.000**
$\mathbf{H_{1.4}}$ Restructure team to be cross-functional → Organizational Skirmish	*-0.007*	0.071	(-0.144, 0.137)	0.917
$\mathbf{H_{1.5}}$ Pair data scientist and software engineers → Organizational Skirmish	0.088	0.077	(-0.065, 0.242)	0.258
$\mathbf{H_{1.6}}$ Introducing a mediator → Organizational Skirmish	0.123	0.068	(-0.007, 0.256)	0.068
$\mathbf{H_{1.7}}$ Organizational Skirmish → Long Release Cycle	-0.045	0.098	(-0.240. 0.143)	0.650
$\mathbf{H_{1.8}}$ Organizational Skirmish → Staff with insufficient skills	-0.175	0.086	(-0.342, -0.007)	**0.042**
$\mathbf{H_{1.9}}$ Organizational Skirmish → Misunderstanding between different roles	*0.033*	0.108	(-0.194, 0.232)	0.758
Organizational Silo				
$\mathbf{H_{2.1}}$ Missing intersections between teams → Organizational Silo	*0.275*	0.075	(0.124, 0.423)	**0.000**
$\mathbf{H_{2.2}}$ Tension Between Teams → Organizational Silo	-0.125	0.075	(-0.295, -0.004)	0.097
$\mathbf{H_{2.3}}$ Restructure team to be cross-functional → Organizational Silo	*-0.028*	0.098	(-0.226, 0.156)	0.774
$\mathbf{H_{2.4}}$ Pair data scientist and software engineers → Organizational Silo	0.150	0.103	(-0.051. 0.355)	0.147
$\mathbf{H_{2.5}}$ Introducing a mediator → Organizational Silo	0.242	0.087	(0.059, 0.403)	**0.006**
$\mathbf{H_{2.6}}$ Organizational Silo → Misunderstanding between different roles	*0.165*	0.105	(-0.050. 0.362)	0.116
Lone Wolf				
$\mathbf{H_{3.1}}$ Tension between management and data scientists → Lone Wolf	*0.385*	0.071	(0.245, 0.526)	**0.000**
$\mathbf{H_{3.2}}$ Restructure team to be cross-functional → Lone Wolf	*-0.016*	0.084	(-0.182, 0.148)	0.851
$\mathbf{H_{3.3}}$ Lone Wolf → Managerial frustration	*0.304*	0.081	(0.153, 0.469)	**0.000**
Prima Donna				
$\mathbf{H_{4.1}}$ Tension between management and data scientists → Prima Donna	*0.371*	0.072	(0.227, 0.508)	**0.000**
$\mathbf{H_{4.2}}$ Restructure team to be cross-functional → Prima Donna	0.098	0.086	(-0.071. 0.262)	0.254
$\mathbf{H_{4.3}}$ Prima Donna → Managerial frustration	*0.345*	0.077	(0.205, 0.501)	**0.000**
Black Cloud				
$\mathbf{H_{5.1}}$ Redundant ML infrastructure and tools → Black Cloud	-0.101	0.083	(-0.266, 0.061)	0.227
$\mathbf{H_{5.2}}$ Shadow IT → Black Cloud	*0.510*	0.070	(0.377, 0.653)	**0.000**
$\mathbf{H_{5.3}}$ Black Cloud → Redundant Infrastructure	-0.251	0.089	(-0.426, -0.080)	**0.005**
$\mathbf{H_{5.4}}$ Black Cloud → Ignoring Organizational Standards	-0.086	0.088	(-0.266, 0.076)	0.326

→ : positive association ↛ : reverse association
Supported Hypothesis - **Significant Hypothesis**

8.3.2 RQ_2—Analysis and Results. RQ_2 aims to determine whether the theorized relationship between community smells and socio-technical aspects in the ML-enabled context (as in RQ_1) varies when we considering the gender, the role of the developer, and the development model being used.

To answer this question, we used Multi-Group analysis, broken down our sample by gender, role, and type of developmental model to explore differences traced to observable characteristics. Multi-group analysis involves running the PLS path model multiple times for different groups, once for each group; groups are captured through categorical variables [32].

Groups Creation. We grouped participants to observe heterogeneity based on gender, roles, and development models. We used data from the Prolific profile of each participant to obtain the gender; We obtained the role and type of development model used in the selection survey. We converted the value in the dataset as follows: **gender**(males = 1 and females = 2), the **role currently held** (Data Scientist = 1. Software Engineer = 2, Data Engineer = 3, Others = 4), and **development model currently used** (Agile Models = 1. Other Models = 2).

Evaluation of Multi-Group Analysis (MGA). We want to assess whether the differences can be attributed to the theoretical constructs and not to how we measured those constructs. To do this, we compare the model reports specific to each group to detect significant differences using a **Multi-Group Analysis (MGA)** [56]. We use the MGA procedure on SmartPLS 4, which examines the correlation between the composite scores of all groups [36].

Groups Comparison and Analysis. We examined the variations in the coefficient paths for the groups and reported all the path coefficients in Table 17. Among the most significant results, we notice mixed opinions about the hypotheses that team restructuring may impact collaboration and communication among people with different backgrounds; moreover, it could involve more developers who tend to work alone.

Another result is that developers with role of Software Engineer have different opinions from developers in other roles. For example Hypothesis $H_{1.4}$, we note that Software Engineer believe that using Pair programs does not close gaps and differences in knowledge, as developers in other roles believe. Moreover, some results show similar opinions among the participants of different genders and also among participants who work with different developmental models. Between the few exceptions, we find perceptions of different backgrounds affecting mutual understanding of the work, causing similar roles to be isolated in independent teams.

Table 17. Multi-Group Analysis.

	Gender		**Role**			**Development Model**			
	Male	Female	Software Engineer	Data Scientist	Data Engineer	Other	Agile Model	Others	All
Size	106	44	48	41	36	25	93	57	150
Organizational Skirmish									
$H_{1.1}$ Missing engineering competence → Organizational Skirmish	0.402	0.519	0.316	0.431	0.352	0.322	0.381	0.499	0.436
$H_{1.2}$ Clash of cultures and tools → Organizational Skirmish	0.010	0.197	0.160	0.069	0.222	-0.284	0.056	0.126	0.074
$H_{1.3}$ Tension Between Teams → Organizational Skirmish	-0.290	-0.364	-0.261	-0.304	-0.364	-0.341	-0.356	-0.173	-0.297
$H_{1.4}$ Restructure team to be cross-functional → Organizational Skirmish	*-0.020*	*0.119*	**-0.00**	**0.189**	**-0.080**	**-0.124**	*0.018*	*-0.104*	-0.007
$H_{1.5}$ Pair data scientist and software engineers → Organizational Skirmish	0.084	0.074	**-0.044**	**0.077**	**0.178**	**0.098**	0.142	0.047	0.088
$H_{1.6}$ Introducing a mediator → Organizational Skirmish	*0.174*	*-0.011*	0.228	0.039	0.121	0.322	0.056	0.276	0.123
$H_{1.7}$ Organizational Skirmish → Long Release Cycle	*-0.095*	*0.069*	**0.030**	**-0.061**	**0.067**	**-0.564**	*-0.154*	*0.179*	-0.045
$H_{1.8}$ Organizational Skirmish → Staff with insufficient skills	-0.152	-0.230	-0.203	-0.140	-0.248	-0.475	*-0.298*	*0.035*	-0.175
$H_{1.9}$ Organizational Skirmish → Misunderstanding between different roles	*0.193*	*-0.295*	**-0.097**	**0.074**	**0.236**	**-0.317**	*0.181*	*-0.136*	0.033
Organizational Silo									
$H_{2.1}$ Missing intersections between teams → Organizational Silo	0.246	0.272	0.208	0.317	0.300	-0.000	0.251	0.369	0.004
$H_{2.2}$ Tension Between Teams → Organizational Silo	-0.119	-0.184	-0.316	-0.204	-0.451	-0.484	-0.100	-0.102	-0.187
$H_{2.3}$ Restructure team to be cross-functional → Organizational Silo	*0.070*	*-0.256*	**-0.045**	**0.256**	**-0.025**	**-0.001**	*-0.120*	*0.065*	0.075
$H_{2.4}$ Pair data scientist and software engineers → Organizational Silo	0.085	0.349	**-0.042**	**0.148**	**0.411**	**-0.266**	0.281	0.031	0.300
$H_{2.5}$ Introducing a mediator → Organizational Silo	0.232	0.278	0.461	0.204	0.068	-0.320	0.270	0.219	0.300
$H_{2.6}$ Organizational Silo → Misunderstanding between different roles	0.145	0.145	0.070	0.172	0.281	0.405	0.186	0.138	0.171
Lone Wolf									
$H_{3.1}$ Tension between management and data scientists → Lone Wolf	0.334	0.437	0.530	0.262	0.341	0.345	0.346	0.441	0.385
$H_{3.2}$ Restructure team to be cross-functional → Lone Wolf	*0.223*	*-0.439*	**0.197**	**-0.116**	**-0.200**	**-0.160**	*-0.138*	*0.126*	-0.016
$H_{3.3}$ Lone Wolf → Managerial frustration	*0.451*	*0.086*	0.272	0.452	0.433	0.240	0.211	0.504	0.304
Prima Donna									
$H_{4.1}$ Tension between management and data scientists → Prima Donna	0.372	0.344	0.525	0.251	0.289	0.291	0.368	0.362	0.371
$H_{4.2}$ Restructure team to be cross-functional → Prima Donna	*0.314*	*-0.270*	**0.194**	**0.114**	**-0.153**	**0.117**	0.025	0.189	0.098
$H_{4.3}$ Prima Donna → Managerial frustration	0.350	0.365	0.301	0.517	0.332	0.444	0.337	0.366	0.345
Black Cloud									
$H_{5.1}$ Redundant ML infrastructure and tools → Black Cloud	-0.118	-0.170	-0.045	-0.138	-0.016	-0.259	-0.010	-0.200	-0.101
$H_{5.2}$ Shadow IT → Black Cloud	0.497	0.562	0.698	0.601	0.415	0.596	0.584	0.486	0.510
$H_{5.3}$ Black Cloud → Redundant Infrastructure	-0.265	-0.252	-0.124	-0.453	-0.231	-0.412	-0.224	-0.301	-0.251
$H_{5.4}$ Black Cloud → Ignoring Organizational Standards	-0.070	-0.170	**0.049**	**-0.254**	**-0.143**	**-0.211**	*-0.189*	*0.072*	-0.086

→ : positive association → : reverse association

Difference between Male and Female - **Different between the different roles** - *Difference between Agile and Others Development Model*

» Summary of the Results of Structural Model - RQ_2

The multi-group analysis tells us:

- There exist considerably different opinions related to the benefits that team restructuring can bring.
- Focusing on the role, software engineers show more sensibly different opinions than others.
- Perceptions of how different roles impact understanding of work and subgroup creation differ among people of different genders and also among those working with different developmental models.

9 DISCUSSION AND LIMITATIONS

Section discusses implications for future research and the threats to the validity of the study.

9.1 Discussion on the Hypotheses

In this section, we discuss the results of our study, describing the observations that warrant further discussion. In the following paragraphs, we report the observations and implications from our study organized by the research question investigated.

To identify relationships between community smells and socio-technical anti-patterns, we used PLS-SEM to create cause-and-effect graphs, where each relation is a hypothesis. The hypotheses, listed in Section 5.1, have been conjectured on the definitions of the various constructs in literature and the insights gained from the interviews conducted. However, some of our hypotheses were not supported by our analysis; this shows us a mismatch between the state of the art related to socio-technical aspects and practitioners' perceptions of them. Following, we discuss the various hypotheses for each community smell.

Organizational Skirmish [59]. The results show that significant hypotheses, with a p-value < 0.05, are $H_{1.1}$, $H_{1.3}$, and $H_{1.8}$. However, looking the coefficient path, we can observe that the supported hypotheses are $H_{1.1}$, $H_{1.2}$, $H_{1.4}$, and $H_{1.9}$. These hypotheses state that different backgrounds and roles lead to problems and difficulties in collaboration and communication, impacting team productivity, even because they tend to adopt different tools, therms, or languages ($H_{1.1}$ and $H_{1.2}$). In addition, the unsupported hypotheses ($H_{1.3}$) show that a misunderstanding in the communication between data scientists and software engineers may not depend on their background as we conjectured, but could depend about others inherent team dynamics. Looking at the assumptions $H_{1.4}$ and $H_{1.5}$ related to mitigation strategy, on one side, the results highlight that to facilitate knowledge and skill sharing to reduce the difference in developers' background a possible strategy may be restructuring teams; on the other side the results highlight that the adoption of pair programming does not mitigate that behavior. A different discussion can be made for $H_{1.6}$, although unsupported, the p-value is close to 0.05, and the path coefficient deviates from 0, suggesting minimal significance for the hypothesis that we conjecture. So, we should consider in future work the hypotheses that introducing a mediator able to understand responsibility and task of both the roles between different roles may impact the performance of developers with diverse backgrounds, to foster knowledge exchange. Moreover, it also reflects how developers are often assigned, in a reverse way, the fact that they are often assigned to tasks for which they do not have the proper knowledge. Other unsupported hypotheses suggest us that cultural differences do not affect developers' mutual understanding and developmental iterations, going to iterate release and continuous structure changes ($H_{1.7}$ and $H_{1.8}$). Meanwhile, the results of $H_{1.9}$ show that, for example, differences may affect the comprehension between the roles of the developers within teams, e.g., Data scientists may not fully understand the importance of software engineers' roles and the good practices to apply during the development phase, and vice versa. In addition to our results, P_4 said, *"I do not perceive major differences in skills and competencies between data engineers and software engineers, maybe because both come from an engineering root. However, I notice more pronounced differences between these roles and data scientists, which focus their studies on different aspects"*. Furthermore, our results shown a mismatch of idea for the Hypotheses $H_{1.4}$, $H_{1.5}$, $H_{1.7}$, and $H_{1.9}$. In particular, according to the results obtained in the multi-group analysis, the women believe that restructuring the team decreases the probability to perceive a difference in background and knowledge between different roles, thinking that such behavior can lead to a delay in releases. On the other hand, men do not find benefits in restructuring the team and believe that those knowledge difference lead to misunderstandings between roles, contrary to what women claim. Developers using an Agile development approach are more likely to

use pair programming and team restructuring to avoid to perceive a difference in background and knowledge between different roles, believing that this behaviour leads to skill gaps in teams and to delays in deliveries. Data scientists believe employing pair programming and team restructuring is advantageous to avoid to perceive a difference in background and knowledge between different roles. However, they do not think that this perceptions causes delivery delays, but they believe that it can cause misunderstandings between different roles; however, these thoughts are not shared by Software Engineers who believe that the difference of knowledge between the different roles does not create misunderstandings between them, but leads to delivery delays.

Organizational Silo [60]. The results of our study, specifically $H_{2.1}$ and $H_{2.6}$, show that software engineers and data scientists tend to work on different tasks based on their skills. Working on various tasks inherently leads the team to split into subgroups, where each is focused on a specific task without communicating or considering the other teams despite the need for integration with their own. A consequence of such behavior may be the misunderstanding of each other's work and problems during the integration phase. A possible solution to challenge could be to include a team member who has the knowledge and skills to perform the tasks assigned to software engineers and the assigned to data scientists. This individual could act as a mediator who can act as a mediator to improve communication between the different teams ($H_{2.3}$). $H_{2.2}$ and $H_{2.4}$ affirm that the lack of understanding among developers of different roles and the adoption of pair programming between data scientists and software engineers don't tends to reduce the creation of sub-teams among developers on the same development team. A further discussion can be made about $H_{2.5}$ since, although it is not supported, the p-value is close to 0.5. This situation suggests that it might be interesting to investigate this hypothesis, which states that the division of teams into subgroups causes a misunderstanding related to the expertise of other developers and the complexity of their work. Moreover, many interviewers support that they work in an environment where the manager themselves used to create subgroups, each composed of the same role, to make teams more productive and focused, e.g., a team of Data Scientists/Engineers focused on building the model and a team of Software Engineers focused on building the product. In addition, our results of the multigroup analysis show a mismatch of idea for the Hypotheses $H_{2.3}$, and $H_{2.4}$. According to the results obtained in the multi-group analysis, both women and developers who use Agile development methods believe that team restructuring does not reduce the probability that the team will be divided into independent, non-collaborative sub-teams, which men and developers using other development models support. Among developers, on the one hand, data scientists believe that adopting pair programming and team restructuring provide benefits by preventing splitting the team into independent and non-collaborative teams. On the other hand, software engineers do not claim that the adoption of pair programming avoids the occurrence of team splitting.

Lone Wolf [16, 60]. The results of $H_{3.1}$ and $H_{3.3}$ support that communication between the manager and the model's team is usually unclear, which pushes these figures to isolate themselves and work independently; the outcome is that these roles focus on their own goals, thus causing frustration on the part of the manager. Even if Hypothesis $H_{3.2}$ is supported, because the path coefficient is negative, it is not significant since the p-value is higher than 0.05. This hypothesis highlights that team restructuring could mitigate the isolation of the data scientists from the others. In addition, P_4 say that *"this behavior is not strictly related to the developer of the model team; but instead to the fact that the manager needs to fully understand the difficulties involved in developing the model and sometimes struggles to manage these issues"*. In addition, P_2 said "*Very often, the isolation factor depends more on a person's character than on their specific role.* ", making it clear that the manifestation of such community smells does not always depend on the role a programmer finds themself in, but also on personal and character factors. However, P_4 believes that a developer's role

greatly impacts their background and way of looking at things, specifically stated " *Data scientists usually go a little bit deeper into the logic of the problems. Otherwise, Software Engineering has a prediction background, and they were excellent, of course, in solving problems*". In conclusion, role and background may influence it to some extent, but they do not turn out to be the main cause. Concerning the multigroup analysis, our results shown a mismatch of idea for the Hypothesis $H_{3.2}$. In particular, male but more in general software engineers believe that restructuring the team reduces the possibility of some team members isolating themselves and not feeling part of the team; such thinking is not supported by women or developers in other roles (i.e., data scientist, data engineers).

Prima Donna [59]. Similar to the Lone Wolf, we also note that the $H_{4.1}$ and $H_{4.3}$ assumptions are supported for the Prima Donna. They state that misunderstanding between managers and developers in the model team causes the latter to isolate themselves and work independently. They are also counterproductive to others, trying to prevail over them or oppose the changes that the manager intends to make. Such behavior tends to create frustration and management difficulties toward managers. Again, $H_{4.2}$ is not supported, indicating that team restructuring is ineffective in addressing the tendency of model teams to work independently. Furthermore, as observed with the Lone Wolf smell, P2 remarked, "Very often, the isolation factor depends more on a person's character than on their specific role," emphasizing that behaviors such as uncooperativeness and superiority are influenced not only by a programmer's role but also by individual personality and character traits. This suggests that structural changes alone may be insufficient to address these issues, as deeper interpersonal and behavioral factors play a significant role. Considering the multigroup analysis, our results shown a mismatch for the Hypothesis $H_{4.2}$. In particular, in the multi-group analysis, women and data engineers believe that restructuring the team does not reduce the possibility that some team members isolate themselves or override team decisions, taking credit for work that is not their own. Such thinking is not supported by men or developers in other roles (i.e., data scientist, software engineers), which believe that by restructuring the team, it is possible to mitigate such behaviors in some of the team members.

Black Cloud [49, 59]. Our results support the $H_{5.2}$. This affirms that although organizations provide an infrastructure, sometimes developers ignore or do not understand it, thus impacting the ability to communicate or share personal information. Some of our interviewers say that their team prefers to use relatively informal communication channels, such as Microsoft Teams or even WhatsApp, avoiding somewhat more formal ones like Skype or Slack. In addition, according to our unsupported Hypotheses $H_{5.1}$, $H_{5.3}$, and $H_{5.3}$, the fact that team member do not understand the presence of a unified infrastructure provided by the company does not induce the occurrence of a black cloud smell. Moreover, the issue of a black cloud smell is not caused by redundancy in the infrastructure. If the organization fails to provide a well-defined communication infrastructure, there naturally will not be any redundancy to address the problem. The multigroup analysis showed a mismatch of idea for the Hypothesis $H_{5.4}$. In particular, according to the results obtained in the multi-group analysis, software engineers believe that organizations do not provide helpful infrastructure for communication and collaboration, leading developers to ignore standards, such as security standards, used by the organization. However, this belief is not shared by other roles (i.e., data scientists, data engineers), who do not perceive the absence of infrastructure as the cause of the adoption of organizational standards.

9.2 Implications for the Practitioners

The goal of our work aims to provide a first step towards understanding community smells in the context of ML-enabled systems by studying whether the socio-technical anti-patterns identified in

this context may be causes, effects, or strategies for already known community smells. The results indicate that not all of the hypothesized conjectures are significant or supported. However, they provide valuable insights, allowing us to deduce different implications.

9.2.1 Implication for Project Managers. Identifying the causes, effects, and organizational strategies for addressing social antipatterns in the context of ML-enabled systems can serve as valuable guidelines for managers. These insights help mitigate team-related risks that may otherwise lead to technical issues within the product. Key insights identified include:

Facilitating collaboration and knowledge exchange- Data scientists and software engineers often come from different educational and professional backgrounds, and these differences can create barriers to collaboration, as Organizational Skirmish and Organizational Silo smells highlights. To overcome this, managers should actively promote collaboration by fostering regular communication, knowledge sharing, and cross-functional teamwork, thereby team alignment and enhancing the quality and efficiency of ML-enabled systems.

Team restructuring- Data scientists often work more independently than other developers, leading to uncooperative or dismissive behavior, such as Prima Donna. This attitude creates division within the team and poses challenges for managers. The findings suggest pairing data scientists with developers on joint tasks to encourage collaboration on projects. Additionally, fostering clear communication and promoting a culture of inclusivity and respect can help break down barriers and improve team cooperation.

Facilitating communication- The lack of structured communication infrastructure in software development leads team members to rely on informal communication, which often results in fragmented information, inconsistent documentation, and the loss of crucial project details, how the Black Cloud smells highlight. Managers should establish a formal communication system that all team members can access to ensure clear and consistent communication throughout the development process.

By following these suggestions, project managers can enhance collaboration and communication among developers, leading to better management of socio-technical challenges in ML-enabled software development. This fosters stronger team dynamics, reduces misalignment, and improves project outcomes.

9.2.2 Implications for Researchers. The results of our study indicate that several assumptions are not supported, exposing a mismatch between existing literature and developers' perceptions of socio-technical issues within teams. These findings highlight implications that may establish a basis for further research and investigation.

Investigate mitigation strategies- The findings suggest that strategies such as team restructuring and pair programming may not be sufficient or fully effective. This highlights the need for further exploration of alternative strategies to manage Data Scientists' behavior. Identifying new solutions could foster collaboration and reduce divisive behaviors.

Mismatch between Software Engineers and Data Scientists- Results revealed differences in how data scientists and software engineers perceive social issues, particularly regarding the effectiveness of management strategies. Researchers should investigate these differing perspectives more deeply to tailor strategies for Project Managers that better align with the needs of diverse team members.

Addressing these implications can enhance the management of heterogeneous teams in ML-enabled systems by improving collaboration, communication, and the integration of developers with different skill sets. It also equips managers with more effective strategies to handle socio-technical challenges, leading to better project outcomes and reduced risks.

9.3 Threats to Validity

Construct Validity. We used and customized pre-existing measurement instruments and created instruments based on earlier research for specific constructs. The measurement model analysis demonstrated that our constructs met the convergent and discriminant validity requirements and were internally consistent. Our qualitative analysis and the results of our work are based on the work and definitions collected by Caballero-Espinosa et al. [13]. Another potential problem is not knowing how long interviewers have been in that specific role and how long they have been using that development model. For this reason, we report the metrics as "currently used" development roles and models. Measuring the cultural background related to the role one plays and the methodologies one uses more precisely could be an interesting topic for future work. A potential threat to the validity of the study arises from the use of reflective measurement models based on the assumption that indicators reflect the latent constructs. However, respondents did not explicitly validate this assumption, which could misalign the model with the true nature of the constructs. Nonetheless, the decision was supported by existing literature and agreed upon by all authors, ensuring alignment with both theoretical foundations and practical realities of the field.

Internal Validity. Among the various limitations, one is related to the reliability of the answers. The questionnaire proposed to measure the constructs was very long, thus bringing a possible bias in the responses caused by boredom. To overcome such biases, interviews were conducted to validate the results. Another limitation is the size of our theoretical model, which, represents only 5 of the various community smells. This choice was made to avoid creating a model and a questionnaire too large to analyze and have more accurate results on specific community smells. However, this is separate from the fact that other community smells can play an equally interesting role, and our results represent a starting point for future studies. The main threat is the way we recruited participants. We relied on voluntary participation through an online tool such as Prolific for the survey. Specifically, this platform provides access to a sample of participants based on specific characteristics,e.g., computer science work as a developer. In our case, we looked for people with (1) a background in computer science, (2) working as Software/Data Engineer or Data Scientist, or (3) working in a team or collaborating with a team of Software/Data Engineer or Data Scientist. In addition, as much as possible, we have tried to balance the number of participants by gender. Prolific allows us to distribute the survey to specific participants, so we conducted a selective survey to understand which possible participants to include in our study; we sent the survey to those who met our acceptance criteria. To ascertain the adequacy of the data set and the chosen sample, the Bartlett and KMO tests were conducted before performing the analysis by PLS-SEM.

External Validity. The questionnaires were conducted on Prolific, going to select people working in collaboration with figures such as Data Scientists, Data Engineers, and Software Engineers. The response rate numbers are aligned with the overall sample distribution calculated through G*Power, so we can consider our results generalized. The responses obtained were sufficiently consistent to find full or partial empirical support for the hypotheses.

Conclusion Validity. Potential challenges in interpreting the results could emerge due to the complexity of the findings. In order to ensure the credibility and validity of the results, the models were meticulously crafted by drawing insights from existing literature and information gathered

through interviews with seasoned developers in the ML-enabled context. This approach aimed to comprehensively understand the subject matter and create an adequate set of hypotheses. The results were obtained using the PLS-SEM approach, following the guidelines detailed in the book *A primer on partial least squares structural equation modeling (PLS-SEM)* [31]. This methodological choice was made to enhance the robustness and reliability of the analysis. The combination of theoretical underpinnings, empirical insights from interviews, and a well-established modeling approach strengthens the confidence in the results and their interpretation.

10 CONCLUSION

The scientific community is moving on to analyze socio-technical phenomena. This concept is also increasingly present in the software engineering world. Our work aims to extend knowledge related to socio-technical aspects, widely analyzed in software engineering [13], by understanding other causes, effects, and possible organizational strategy correlations arising from socio-technical aspects in the context of ML-enabled systems [44]. The results found report interesting statements. In particular, analyzing the relationships involving community smells Prima Donna and Lone Wolf and how they are accentuated when it comes to manager-developer interactions of the model team. Also very interesting is the perception given by the analysis of the Black Cloud, which makes us understand how often practitioners tend to ignore the communication standards provided by the company, creating their communication infrastructure, which then turns out to be redundant. Among the most noteworthy aspects is the discourse related to the differences in background given by different roles, which participants perceive as one of the reasons associated with the lack of understanding among them and a possible cause of subdivision of developers into subgroups according to their roles. For our study, we analyzed 5 of the most well-known community smells. However, this does not mean that the others may also provide as much interesting information to analyze. In future work, we aim to: (a) Investigate causes related to unsupported hypotheses; (b) Expand the work by analyzing additional community smells and socio-technical aspects in the ML-enabled systems context; (c) Investigate the link between Black Cloud and Shadow IT, as the observed phenomena appear to be very similar to each other; (d) Conduct further investigation into moderating factors that may influence survey results, considering aspects such as team size and experience of developers.

ACKNOWLEDGMENTS

This work has been partially supported by the *EMELIOT* national research project, which has been funded by the MUR under the PRIN 2020 program (Contract 2020W3A5FY), and *QUAL-AI* national research projects funded by the EU - NGEU and the MUR under the PRIN 2022 program (Contracts 2022B3BP5S). We sincerely thank the reviewers and the editor for their valuable comments and suggestions, which have significantly contributed to enhancing the quality of our paper.

REFERENCES

[1] Khaled Albusays, Pernille Bjorn, Laura Dabbish, Denae Ford, Emerson Murphy-Hill, Alexander Serebrenik, and Margaret-Anne Storey. 2021. The Diversity Crisis in Software Development. *IEEE Software* 38, 2 (2021), 19–25. https://doi.org/10.1109/MS.2020.3045817

[2] Nuri Almarimi, Ali Ouni, Moataz Chouchen, and Mohamed Wiem Mkaouer. 2021. csDetector: an open source tool for community smells detection. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1560–1564.

[3] Dorine Andrews, Blair Nonnecke, and Jennifer Preece. 2007. Conducting research on the internet:: Online survey design, development and implementation guidelines. (2007).

[4] Giusy Annunziata, Stefano Lambiase, Gemma Catolino, Damian A. Tamburri, Willem-Jan Van Den Heuvel, Fabio Palomba, Filomena Ferrucci, and Andrea De Lucia. 2024. *Online Appendix - Uncovering Community Smells in Machine*

Learning-Enabled Systems: Causes, Effects, and Mitigation Strategies. https://figshare.com/s/1449581f3fe126da76fb

[5] Claudia Binz Astrachan, Vijay K. Patel, and Gabrielle Wanzenried. 2014. A comparative study of CB-SEM and PLS-SEM for theory development in family firm research. *Journal of Family Business Strategy* 5, 1 (2014), 116–128. https://doi.org/10.1016/j.jfbs.2013.12.002 Innovative and Established Research Methods in Family Business.

[6] Richard P Bagozzi and Youjae Yi. 1988. On the evaluation of structural equation models. *Journal of the academy of marketing science* 16 (1988), 74–94.

[7] Sebastian Baltes and Paul Ralph. 2022. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering* 27, 4 (2022), 94.

[8] Maurice S Bartlett. 1950. Tests of significance in factor analysis. *British journal of psychology* (1950).

[9] Andrew Begel and Beth Simon. 2008. Novice software developers, all over again. In *Proceedings of the fourth international workshop on computing education research*. 3–14.

[10] Andrew Begel and Thomas Zimmermann. 2014. Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*. 12–23.

[11] Thomas M Brill, Laura Munoz, and Richard J Miller. 2022. Siri, Alexa, and other digital assistants: a study of customer satisfaction with artificial intelligence applications. In *The Role of Smart Technologies in Decision Making*. Routledge, 35–70.

[12] Gabriel Busquim, Hugo Villamizar, Maria Julia Lima, and Marcos Kalinowski. 2024. On the Interaction Between Software Engineers and Data Scientists When Building Machine Learning-Enabled Systems. In *Software Quality as a Foundation for Security*, Peter Bludau, Rudolf Ramler, Dietmar Winkler, and Johannes Bergsmann (Eds.). Springer Nature Switzerland, Cham, 55–75.

[13] Eduardo Caballero-Espinosa, Jeffrey C Carver, and Kimberly Stowers. 2023. Community smells—The sources of social debt: A systematic literature review. *Information and Software Technology* (2023), 107078.

[14] Marcelo Cataldo, Patrick A Wagstrom, James D Herbsleb, and Kathleen M Carley. 2006. Identification of coordination requirements: Implications for the design of collaboration and awareness tools. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*. 353–362.

[15] Gemma Catolino, Fabio Palomba, Damian Andrew Tamburri, and Alexander Serebrenik. 2021. Understanding community smells variability: A statistical approach. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 77–86.

[16] Gemma Catolino, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, and Filomena Ferrucci. 2019. Gender Diversity and Women in Software Teams: How Do They Affect Community Smells?. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. 11–20. https://doi.org/10.1109/ICSE-SEIS.2019.00010

[17] Gemma Catolino, Fabio Palomba, Damian Andrew Tamburri, Alexander Serebrenik, and Filomena Ferrucci. 2020. Refactoring community smells in the wild: the practitioner's field manual. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*. 25–34.

[18] Peter M Chisnall. 1996. Qualitative Interviewing: The Art of Hearing Data. *Journal of the Market Research Society* 38, 4 (1996), 553–555.

[19] Manuel De Stefano, Emanuele Iannone, Fabiano Pecorelli, and Damian Andrew Tamburri. 2022. Impacts of Software Community Patterns on Process and Product: An Empirical Study. *Science of Computer Programming* 214 (Feb. 2022), 102731. https://doi.org/10.1016/j.scico.2021.102731

[20] Tim Dreesen, Phil Hennel, Christoph Rosenkranz, and Thomas Kude. 2021. "The Second Vice is Lying, the First is Running into Debt." Antecedents and Mitigating Practices of Social Debt: An Exploratory Study in Distributed Software Development Teams. (2021).

[21] Tore Dybå and Torgeir Dingsøyr. 2008. Empirical studies of agile software development: A systematic review. *Information and software technology* 50, 9-10 (2008), 833–859.

[22] Felipe Ebert, Alexander Serebrenik, Christoph Treude, Nicole Novielli, and Fernando Castor. 2022. On Recruiting Experienced GitHub Contributors for Interviews and Surveys on Prolific. In *International Workshop on Recruiting Participants for Empirical Software Engineering*.

[23] Beyza Eken, Francis Palma, Başar Ayşe, and Tosun Ayşe. 2021. An empirical study on the effect of community smells on bug prediction. *Software Quality Journal* 29 (2021), 159–194.

[24] Franz Faul, Edgar Erdfelder, Axel Buchner, and Albert-Georg Lang. 2009. Statistical power analyses using G* Power 3.1: Tests for correlation and regression analyses. *Behavior research methods* 41, 4 (2009), 1149–1160.

[25] Timothy S Flanigan, Emily McFarlane, and Sarah Cook. 2008. Conducting survey research among physicians and other medical professionals: a review of current literature. In *Proceedings of the Survey Research Methods Section, American Statistical Association*, Vol. 1. 4136–47.

[26] Harald Foidl, Michael Felderer, and Rudolf Ramler. 2022. Data smells: Categories, causes and consequences, and detection of suspicious data in ai-based systems. In *Proceedings of the 1st International Conference on AI Engineering:*

Software Engineering for AI. 229–239.

[27] Jill J Francis, Marie Johnston, Clare Robertson, Liz Glidewell, Vikki Entwistle, Martin P Eccles, and Jeremy M Grimshaw. 2010. What is an adequate sample size? Operationalising data saturation for theory-based interview studies. *Psychology and health* 25, 10 (2010), 1229–1245.

[28] Joseph F Hair. 1995. Multivariate data analysis with readings. (1995).

[29] Joseph F Hair, Arthur H Money, Philip Samouel, and Mike Page. 2007. Research methods for business. *Education+ Training* 49, 4 (2007), 336–337.

[30] Joseph F Hair, Jeffrey J Risher, Marko Sarstedt, and Christian M Ringle. 2019. When to use and how to report the results of PLS-SEM. *European business review* 31, 1 (2019), 2–24.

[31] Joe Hair Jr, Joseph F Hair Jr, G Tomas M Hult, Christian M Ringle, and Marko Sarstedt. 2021. *A primer on partial least squares structural equation modeling (PLS-SEM)*. Sage publications.

[32] Joseph F Hair Jr, Marko Sarstedt, Christian M Ringle, and Siegfried P Gudergan. 2023. *Advanced issues in partial least squares structural equation modeling*. saGe publications.

[33] Petra Heck. 2024. What About the Data? A Mapping Study on Data Engineering for AI Systems. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 43–52.

[34] James Heckman. 1990. Varieties of selection bias. *The American Economic Review* 80, 2 (1990), 313–318.

[35] Jörg Henseler, Christian M Ringle, and Marko Sarstedt. 2015. A new criterion for assessing discriminant validity in variance-based structural equation modeling. *Journal of the academy of marketing science* 43 (2015), 115–135.

[36] Jörg Henseler, Christian M Ringle, and Marko Sarstedt. 2016. Testing measurement invariance of composites using partial least squares. *International marketing review* 33, 3 (2016), 405–431.

[37] Siw Elisabeth Hove and Bente Anda. 2005. Experiences from conducting semi-structured interviews in empirical software engineering research. In *11th IEEE International Software Metrics Symposium (METRICS'05)*. IEEE, 10–pp.

[38] Zijie Huang, Zhiqing Shao, Guisheng Fan, Jianhua Gao, Ziyi Zhou, Kang Yang, and Xingguang Yang. 2021. Predicting community smells' occurrence on individual developers by sentiments. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*. IEEE, 230–241.

[39] Katherine J Hunt, Natalie Shlomo, and Julia Addington-Hall. 2013. Participant recruitment in sensitive surveys: a comparative trial of 'opt in'versus 'opt out'approaches. *BMC medical research methodology* 13 (2013), 1–8.

[40] Ankur Joshi, Saket Kale, Satish Chandel, and D Kumar Pal. 2015. Likert scale: Explored and explained. *British journal of applied science & technology* 7, 4 (2015), 396–403.

[41] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2017. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering* 44, 11 (2017), 1024–1038.

[42] Barbara A Kitchenham and Shari L Pfleeger. 2008. Personal Opinion Surveys. In *Guide to advanced empirical software engineering*. Springer, 63–92.

[43] Stefano Lambiase, Gemma Catolino, Damian A. Tamburri, Alexander Serebrenik, Fabio Palomba, and Filomena Ferrucci. 2022. Good Fences Make Good Neighbours? On the Impact of Cultural and Geographical Dispersion on Community Smells. In *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS '22)*. Association for Computing Machinery, New York, NY, USA, 67–78. https://doi.org/10.1145/3510458.3513015

[44] Alina Mailach and Norbert Siegmund. 2023. Socio-Technical Anti-Patterns in Building ML-Enabled Software: Insights from Leaders on the Forefront. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 690–702. https://doi.org/10.1109/ICSE48619.2023.00067

[45] Marcelo Morandini, Thiago Adriano Coleti, Edson Oliveira, and Pedro Luiz Pizzigatti Corrêa. 2021. Considerations about the efficiency and sufficiency of the utilization of the Scrum methodology: A survey for analyzing results for development teams. *Computer Science Review* 39 (2021), 100314. https://doi.org/10.1016/j.cosrev.2020.100314

[46] Nadia Nahar, Shurui Zhou, Grace Lewis, and Christian Kästner. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 413–425. https://doi.org/10.1145/3510003.3510209

[47] Ipek Ozkaya. 2020. What is really different in engineering AI-enabled systems? *IEEE software* 37, 4 (2020), 3–6.

[48] Fabio Palomba, Damian Andrew Tamburri, Francesca Arcelli Fontana, Rocco Oliveto, Andy Zaidman, and Alexander Serebrenik. 2021. Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells? *IEEE Transactions on Software Engineering* 47, 1 (Jan. 2021), 108–129. https://doi.org/10.1109/TSE.2018.2883603

[49] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Fausto Fasano, Rocco Oliveto, and Andrea De Lucia. 2018. A large-scale empirical study on the lifecycle of code smell co-occurrences. *Information and Software Technology* 99 (2018), 1–10. https://doi.org/10.1016/j.infsof.2018.02.004

[50] Fabio Palomba and Damian Andrew Tamburri. 2021. Predicting the Emergence of Community Smells Using Socio-Technical Metrics: A Machine-Learning Approach. *Journal of Systems and Software* 171 (Jan. 2021), 110847. https://doi.org/10.1016/j.jss.2020.110847

[51] David Piorkowski, Soya Park, April Yi Wang, Dakuo Wang, Michael Muller, and Felix Portnoy. 2021. How ai developers overcome communication challenges in a multidisciplinary team: A case study. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–25.

[52] Gilberto Recupito, Raimondo Rapacciuolo, Dario Di Nucci, and Fabio Palomba. 2024. Unmasking Data Secrets: An Empirical Investigation into Data Smells and Their Impact on Data Quality. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*. 53–63.

[53] Brittany Reid, Markus Wagner, Marcelo d'Amorim, and Christoph Treude. 2022. Software Engineering User Study Recruitment on Prolific: An Experience Report. *arXiv preprint arXiv:2201.05348* (2022).

[54] Daniel Russo and Klaas-Jan Stol. 2021. PLS-SEM for software engineering research: An introduction and survey. *ACM Computing Surveys (CSUR)* 54, 4 (2021), 1–38.

[55] Marko Sarstedt, Christian M Ringle, and Joseph F Hair. 2017. Treating unobserved heterogeneity in PLS-SEM: A multi-method approach. *Partial least squares path modeling: Basic concepts, methodological issues and applications* (2017), 197–217.

[56] Marko Sarstedt, Christian M Ringle, and Joseph F Hair. 2021. Partial least squares structural equation modeling. In *Handbook of market research*. Springer, 587–632.

[57] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. *Advances in neural information processing systems* 28 (2015).

[58] Damian Andrew Tamburri, Rick Kazman, and Hamed Fahimi. 2016. The architect's role in community shepherding. *IEEE Software* 33, 6 (2016), 70–79.

[59] Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. 2015. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications* 6 (2015), 1–17.

[60] Damian A. Tamburri, Fabio Palomba, and Rick Kazman. 2021. Exploring Community Smells in Open-Source: An Automated Approach. *IEEE Transactions on Software Engineering* 47, 3 (March 2021), 630–652. https://doi.org/10.1109/TSE.2019.2901490

[61] Bianca Trinkenreich, Klaas-Jan Stol, Anita Sarma, Daniel M German, Marco A Gerosa, and Igor Steinmacher. 2023. Do i belong? modeling sense of virtual community among linux kernel contributors. *arXiv preprint arXiv:2301.06437* (2023).

[62] Bianca Trinkenreich, Klaas-Jan Stol, Igor Steinmacher, Marco A. Gerosa, Anita Sarma, Marcelo Lara, Michael Feathers, Nicholas Ross, and Kevin Bishop. 2023. A Model for Understanding and Reducing Developer Burnout. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 48–60. https://doi.org/10.1109/ICSE-SEIP58684.2023.00010

[63] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. 2015. Gender and tenure diversity in GitHub teams. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 3789–3798.

[64] Gianmario Voria, Viviana Pentangelo, Antonio Della Porta, Stefano Lambiase, Gemma Catolino, Fabio Palomba, and Filomena Ferrucci. 2022. Community Smell Detection and Refactoring in SLACK: The CADOCS Project. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 469–473.

[65] David Wicks. 2017. The coding manual for qualitative researchers. *Qualitative research in organizations and management: an international journal* 12, 2 (2017), 169–170.

[66] Haiyin Zhang, Luís Cruz, and Arie Van Deursen. 2022. Code smells for machine learning applications. In *Proceedings of the 1st international conference on AI engineering: software engineering for AI*. 217–228.