




Community smells in software development communities: systematic literature review

Olores comunitarios en comunidades de desarrollo de software: Revisión Sistemática de la Literatura

Eydy del Carmen Suárez Brieval²   Jorge Enrique Bejarano Guar¹  Nicole Daniela Urbano Palechor¹  César Jesús Pardo Calvache¹ 

¹ GTI Research Group, Universidad del Cauca, Popayán, Colombia.

² GISICO Research Group, Universidad Popular del Cesar, Valledupar, Colombia.

Abstract

Objective: This study presents the initial findings of a Systematic Literature Review aimed at identifying potential anti-patterns, known as community smells, which, if not effectively addressed, can generate underlying social issues within software development teams.

Methodology: The study was conducted through a Systematic Literature Review that provided a detailed and structured analysis of research related to community smells in software development teams. The article focuses on three main aspects: (i) demographic trends within the topic, (ii) types of research, and (iii) definitions of community smells as provided by different authors. The review followed the protocol proposed by Kitchenham, enabling the identification, analysis, and rigorous evaluation of the available literature.

Results: A total of 44 primary studies were selected, analyzed, and evaluated, leading to the identification of 35 distinct community smells. Among the most prominent are organizational silo, black cloud, lone wolf, and bottleneck, along with their respective causes and adverse effects.

Conclusions: The Systematic Literature Review allowed for the identification of the current state of research on community smells and the main anti-patterns associated with their emergence. These issues can hinder interaction and teamwork among members, resulting in failures, delays, incomplete or defective software artifacts, and the emergence of psychosocial risks.

Keywords: Anti-patterns; Communication; Cooperation; Coordination; Software communities; Social debt; Community smells.

Resumen

Objetivo: Este estudio presenta los hallazgos iniciales de una revisión sistemática de la literatura, cuyo objetivo fue identificar posibles anti-patrones, conocidos como olores comunitarios, que, si no se abordan eficazmente, pueden generar problemas sociales subyacentes en los equipos de desarrollo de software.

Metodología: El estudio se realizó mediante una revisión sistemática de la literatura que proporcionó un análisis detallado y estructurado de la investigación relacionada con los olores comunitarios en equipos de desarrollo de software. El artículo se centra en tres aspectos principales: (i) tendencias demográficas en el tema, (ii) tipos de investigación y (iii) definiciones de olores comunitarios proporcionadas por diferentes autores. La revisión siguió el protocolo propuesto por Kitchenham, lo que permitió la identificación, el análisis y la evaluación rigurosa de la literatura disponible.

Resultados: Se seleccionaron, analizaron y evaluaron un total de 44 estudios primarios, lo que permitió la identificación de 35 olores comunitarios distintos. Entre los más destacados se encuentran el silo organizacional, la nube negra, el lobo solitario y el cuello de botella, junto con sus respectivas causas y efectos adversos.

Conclusiones: La revisión sistemática de la literatura permitió identificar el estado actual de la investigación sobre olores comunitarios y los principales antipatrones asociados a su aparición. Estos problemas pueden dificultar la interacción y el trabajo en equipo entre los miembros, lo que resulta en fallos, retrasos, software incompleto o defectuoso, y la aparición de riesgos psicosociales.

Palabras clave: Antipatrones; Comunicación; Cooperación; Coordinación; Comunidades de software; Deuda social; Olores comunitarios..

How to cite?

Suárez EC, Bejarano JE, Urbano ND, Pardo CJ. Community smells in software development communities: systematic literature review. Ingeniería y Competitividad, 2025, 27;(2):e-30414826

<https://doi.org/10.25100/iyv.v27i2.14826>

Received: 27-03-25

Evaluated: 5-05-25

Accepted: 24-02-25

Online: 23-04-25

Correspondence

eydysuarez@unicesar.edu.co



Spanish version



Contribution to the literature

Why was it conducted?

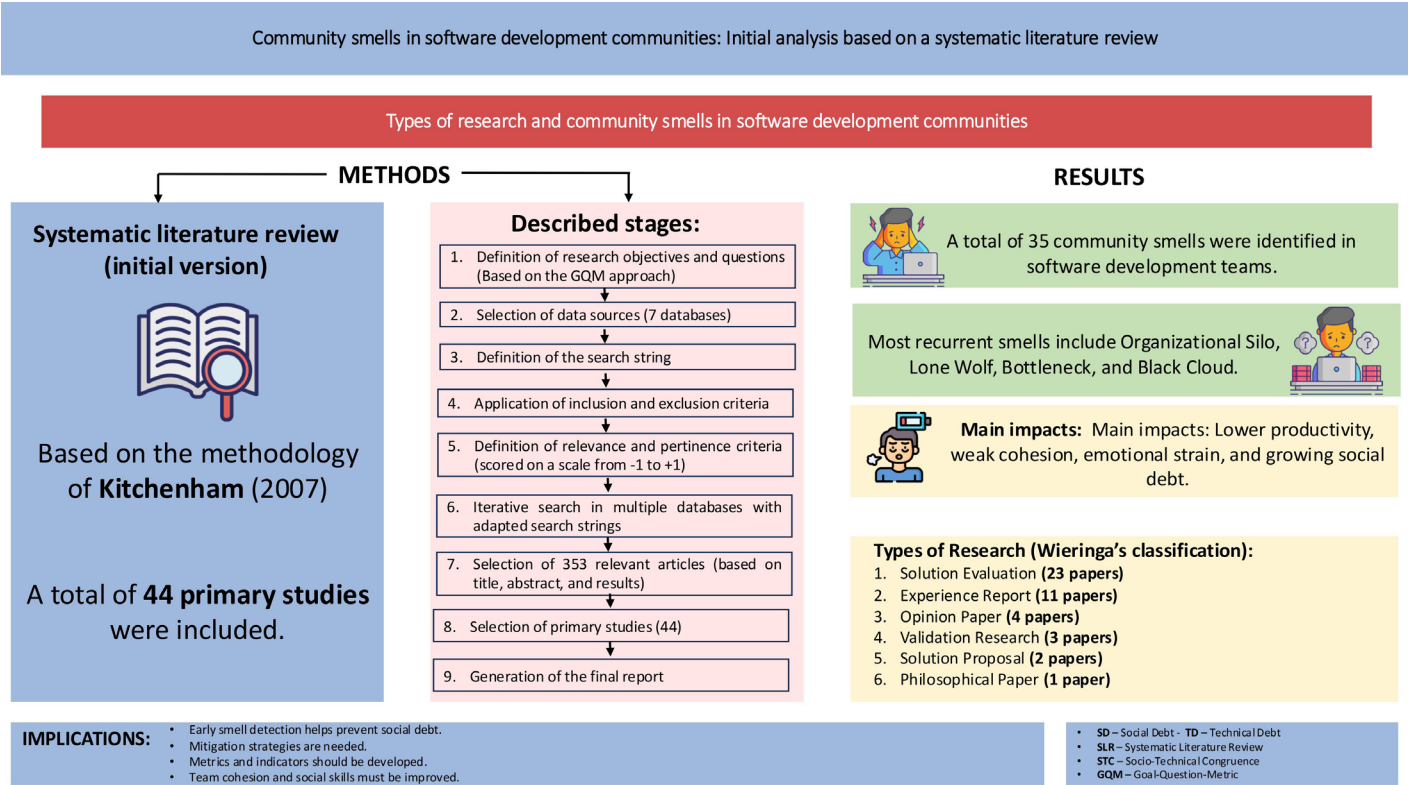
Software development extends beyond technical aspects; it is a socio-technical activity that demands a balance between social and technical skills. In this context, the human factor is critical to project success. The primary motivation for this article is to identify and analyze the factors that negatively affect the emotional and social well-being of members of software development communities, focusing on the phenomenon known as “community smells.” Furthermore, this study seeks to assess the importance placed on these issues within the industry, to identify the most significant research in the field, and to map the countries that have led the study of this topic.

What were the most relevant results?

The most significant findings of the study reveal the existence of 35 community smells within the context of software development. The research demonstrates that inadequate management of these smells negatively impacts not only individuals and teams but also affects organizational performance and the technical quality of developed products. The analysis further indicates that if the underlying causes of these smells are not effectively addressed, the accumulation of social debt may occur, with adverse effects on productivity, efficiency, and the overall well-being of team members.

What do these results contribute?

In addition, the study identified and classified both general and specific causes, associated effects, and strategies for managing and mitigating these community smells. These findings provide a solid conceptual and practical foundation for understanding and preventing psychosocial problems in software development communities, thereby contributing to enhanced team management and improved quality in software engineering processes.



Introduction

The software development process comprises several stages, tasks, and activities that must be addressed throughout a software project, from its initial development to its eventual retirement. Consequently, team collaboration is essential, as constant communication is required for scheduling and organizing tasks, solving problems, and monitoring the project's progress within the planned timeline [\(1\)](#). It is also critical that team members apply their technical knowledge to ensure the delivery of a quality product—software that is functional, efficient, secure, and maintainable. To this end, it is necessary to integrate social and technical skills that enable the team to adapt to new challenges and technologies while promoting continuous improvement within projects [\(2\)](#).

It is important to emphasize that a balance between social and technical skills contributes to project success, as reflected in the quality of the final product and the cultivation of a productive and organized work environment, especially one grounded in effective communication [\(3\)](#). Conversely, the lack of such balance may result in (i) disagreements or misinterpretations regarding assigned tasks, (ii) changes in roles after the project has commenced, and (iii) modifications in activities without prior clarification or feedback. These issues can engender distrust, stress, irritation, and a communication breakdown, thereby increasing psychosocial risk [\(4,5\)](#).

For this reason, if an organization does not invest in managing these human aspects, a phenomenon known as social debt may arise. Social debt can be described as the accumulation of negative consequences stemming from poor communication within the work team, leading to misinterpretation of ideas, errors, project delays, and insufficient collaboration [\(6–10\)](#). Social debt (SD) occurs when communication and support are not encouraged in the work environment, resulting in departments, areas, or even individual team members working in isolation without engaging with other members of the organization. This leads to a failure to effectively share resources, information, and objectives. It also manifests in individuals who prefer to work alone and in isolation, making decisions without consulting others or executing assigned activities without considering the viewpoints or opinions of others.

All these factors are reflected in team productivity, resistance to change and challenges, obstacles to the smooth and accessible flow of information, and a lack of encouragement for mutual learning [\(6–9\)](#). Such conditions manifest as community smells—socio-technical anti-patterns and a source of SD—that affect software development communities (SDCs), their processes, outcomes, and the broader organization. Community smells can be understood as symptoms or indicators of problematic situations or negative effects affecting development team members, processes, and/or artifacts, such as unclear or missing documentation, noisy code, incomplete manuals, and improper tool usage. These issues require extra effort to understand or maintain affected artifacts, reducing software quality and increasing production costs.

Moreover, community smells have been associated with various negative phenomena, such as high employee turnover and poor architectural decisions that often extend beyond technical problems. These community smells can also lead to organizational turmoil or even decrease the stability of software projects and/or precipitate project failure [\(11–16\)](#). Furthermore, it is pertinent to note

that predicting community smells within a community before they manifest during the software project timeline can be critical for anticipating suboptimal organizational and socio-technical conditions before they become unmanageable (15). The failure to address community smells fosters the formation of dysfunctional communities, which produces consequences not only at the organizational level but also in the creation of unfavorable work environments. Such conditions may adversely affect the mental health of personnel, triggering negative emotions such as irritability, fear, discouragement, burnout, and decreased motivation. They also contribute to social isolation and diminished emotional competence—factors that impair cooperation and team performance, ultimately reducing productivity and efficiency. Taken together, these effects have a detrimental impact on the overall well-being and quality of life of employees (12,17–21).

It is important to highlight that this situation may result in the establishment of an unfavorable and poorly managed work environment, generating consequences such as: (i) an increase in mental health issues, including depression, anxiety, and burnout, which undermine team well-being; (ii) physical conditions such as hypertension, cardiovascular diseases, and musculoskeletal disorders, as evidenced by increased medical costs and absenteeism; (iii) poor job performance, reflected in increased errors and lower productivity and quality in task execution; (iv) job dissatisfaction, as indicated by low commitment and high turnover; (v) diminished innovation and creativity due to stress and demotivation, leading to stagnation among team members and in project development; and (vi) heightened interpersonal conflicts, manifesting as poor emotional and professional communication among team members, resulting in disagreements (5,22–24). Given the above and to provide a structured and critical evaluation of social debt and community smell issues in software development teams, a literature analysis was conducted. The results identified 35 community smells, which are detailed in the annex, Table 10.

The preliminary Systematic Literature Review (SLR) is organized as follows: Section II presents an analysis of the studies related to the topic; Section III outlines the research methodology used for the study; Section IV details the results about the research questions; Section V presents the conclusions and future work; and finally, Section VI provides the bibliographical references.

Literature review

The SLR provides a synthesis of relevant studies on the topics of SD and community smells, including methodology, results, future work, and conclusions. The following are some of the research studies identified that enrich and contribute significantly to the topic and its conceptual foundation.

In a study by Espinosa et al. (17), the authors indicate that SD arises when shortcuts are taken in software development, which can generate long-term problems. These shortcuts, though adopted to save time or money, may ultimately result in higher future costs. In this context, SD describes the adverse consequences of policies and procedures that affect individuals and communities; these impacts can range from loss of income to disruptions in the mental equilibrium of professionals within SDCs. Communities affected by SD are likely to generate additional project costs and are referred to as “suboptimal” communities (7). On the other hand, Tamburri et al. (25) define SD as the absence of essential actions, conditions, or services that affect the well-being of community

members. Similarly, several authors (26–30) define SD as the unforeseen additional cost that can be incurred in a software project. These additional costs occur due to deficient practices in communication, cooperation, coordination, and collaboration among members of an SDC, or due to strict policies, obsolete processes, or tools established by the organization that do not promote the development of soft skills among members; these conditions generate negative causes and/or effects in the community. Dreesen et al. (29) also indicate that SD refers to the future consequences of decisions related to people and their interrelationships. Furthermore, a study conducted by (31) underlines that software engineering is, by nature, a social activity involving developers, managers, and stakeholders from different parts of the world. This brings several challenges, especially in terms of collaboration and communication, such as personality conflicts and language and cultural barriers, as indicated in (6–9).

Additionally, it is relevant to highlight what is mentioned in (32), where it is identified that, because of SD, inadequate and deficient working conditions can occur, manifesting as psychosocial risk factors. These factors have the potential to adversely impact workers' emotional, social, cognitive, and physical health, generating stress, frustration, feelings of defeat, sleep disorders, work overload, and, therefore, the need or obligation to work overtime (5,23,24) to achieve the goals defined in the software project. It has been shown that STC can be used as an early indicator for the identification of SD and, consequently, community smells. Adequate alignment between technical dependencies and social interactions, particularly in communication, coordination, and cooperation processes, helps to mitigate the effects derived from suboptimal decisions that compromise organizational dynamics. These decisions can generate interpersonal tensions, loss of cohesion, and accumulation of SD (10,14,28,33).

Another study, conducted by Almarimi et al. (34), explains the term community smells as suboptimal social and organizational conditions within a development community. These conditions are likely to lead to cost overruns in a software project and, possibly, to the emergence of SD, which will directly affect professionals in SDC and lead to the appearance of new community smells.

In another study, Palomba et al. (35) state that community smells represent suboptimal conditions that arise in SDC, such as a lack of communication between sub-teams, which can lead to the presence of SD and increase overall project costs. To anticipate the manifestation of these community smells, metrics associated with STC are used.

Additionally, the analysis by Catolino et al. (36), defines community smells as suboptimal organizational structures capable of generating SD. These smells are frequent in both open-source and proprietary code projects and are perceived as detrimental by practitioners; they can even lead to the introduction of TD in the source code. It is important to highlight the contribution by Palomba et al. (37), who state that identifying community smells within an SDC before their manifestation during the development of a software project implies recognizing early signs of organizational or socio-technical problems that could affect project success. By identifying these signs, preventive actions can be implemented to keep obstacles from becoming unmanageable as the project progresses.

Based on the above, and from the evaluation of the results of the primary studies identified in the SLR, it became evident that there are proposals related to the management, detection, and reduction of community smells in SDC. Some of these studies suggest that to eliminate or reduce community smells, an effort is required from the members of the work team through the recommendation of individual support measures for each member, according to their motivations, working styles, and status (16). In addition, several community smells were identified within SDC. The results show that the most frequent smells are organizational silo, black cloud, lone wolf, and radio silence. From these findings, it was observed that no research evaluates the documentation, causes, and/or negative effects that community smells can generate in a development community. Once the concepts of SD and community smells have been defined, the most relevant SLR on the topic are presented below.

Espinosa et al. (17) conducted an SLR in which they identified 25 primary studies, based on searches in recognized databases such as Scopus, IEEE Xplore, Google Scholar, Science Direct, ACM, Springer, and Web of Science. The findings of this review highlighted the detrimental effects of SD on the performance of development teams, negatively impacting collaborative decision-making, interpretation of information, ability to recognize challenges, and formulation of coping strategies. The study also showed that SD can foster interpersonal tensions, decrease individual commitment, generate ambiguity in role assignment, and increase frustration among team members—these effects being the consequence of poor communication and leadership. As part of the results, 30 community smells linked to the causes of SD were identified, which allowed a deeper understanding of their nature and organizational implications.

In a complementary approach, Saeeda et al. (21) conducted a multivariate systematic review focusing on recent research on NTD. For this purpose, they analyzed 110 sources, both scientific and gray literature, and selected 40 primary studies using the PICO strategy. The study highlights that many failures in software development have a social or human origin, emphasizing a significant gap in NTD research, despite growing interest in TD. The review identifies five types of NTD: process, social, people-related, organizational, and cultural, each associated with components of the socio-technical approach. Among the most common causes were poor communication, poor leadership, inefficient processes, and toxic organizational environments. Finally, the study proposes tools, mitigation strategies, and a conceptual framework to facilitate their understanding and management in different organizational contexts.

On the other hand, Raza et al. (38) developed an SLR focused on the in-depth analysis of the STC concept and its relevance in the context of software development. STC focuses on the alignment of three key factors: communication, coordination, and cooperation, which are essential for social interactions within development communities. The review considered 46 primary studies published between 2008 and 2019, selected by searching databases such as IEEE, ACM, Scopus, and Web of Science. The study sought to structure and synthesize the available knowledge on STC, covering its components, measurement techniques, influencing factors, its impact on team performance and product quality, as well as the consequences derived from its absence. Among the results, two main approaches to measuring STC are highlighted: one based on matrices and the other on social network analysis. Greater congruence is related to better results, while its lack can lead to delays,

errors, and failures. In addition, gaps in the literature were detected, such as the limited application of these metrics in phases other than development and the limited attention to contextual factors.

Likewise, Tahsin et al. (39) addressed the study of community smells in software engineering using an SLR. These smells represent dysfunctional social interaction patterns that, if not managed promptly, can negatively affect communication, collaboration, and team performance. The review was based on searches in four scientific databases: IEEE Xplore, ACM Digital Library, Scopus, and Springer Link, which identified 44 primary studies published up to 2022. The analysis allowed the classification of 14 types of community smells, the most common being organizational silos, lone wolf, bottleneck, black cloud, and missing link. The most common causes included communication deficiencies, inadequate organizational structures, lack of trust among team members, and ambiguity in role assignment. In terms of effects, the decline in collective productivity, the emergence of internal conflicts, and the accumulation of SD and TD were highlighted. For their detection, approaches such as socio-technical metrics, SNA, and machine learning techniques were used. However, there is a lack of standardized and widely adopted tools in industrial contexts, highlighting the need to integrate automatic detection mechanisms with organizational interventions aimed at strengthening culture, communication, and team structure, as well as deepening the relationship between community smells and other forms of NTD.

Finally, Raza et al. (40), conducted a scientometric analysis to examine the evolution, knowledge structure, and emerging trends of the STC concept in software development. The work was based on 306 academic articles collected between 2000 and 2020 through the Scopus and Web of Science databases. The main objective was to map the existing knowledge on STC, identify its main research lines, influential authors and institutions, and classify the most relevant topics addressed in the literature. For this purpose, four scientometric techniques were used: co-word analysis, co-authorship, co-citation, and document clustering with timeline analysis. As a result, six thematic axes were identified: community structure, STC, current research, product structure, key developers, and the last two decades, its growing relevance for addressing coordination issues in distributed teams, and the need to move towards integrative models that combine technical and social metrics were evidenced. This research provides a comprehensive view of the state of the art and establishes a solid foundation for future research, while mitigating the subjectivity of previous studies by combining quantitative analysis and structured critical review.

Table 1 below presents the comparative analysis of the related studies. The studies reviewed address community smells from complementary perspectives and with varying levels of theoretical and methodological depth. The work of Espinosa et al. (17) conducts an exhaustive systematic review, identifying 30 community smells and analyzing their properties, causes, effects, and mitigation strategies. Their approach is broad, formal, and structured, supported by models such as the Community Smell Stages Framework and graphical representations (e.g., Sankey diagrams). In contrast, the present article offers an initial exploration with the identification of 35 community smells, but does not incorporate a demographic analysis, focusing instead on a systematic and conceptual treatment of the phenomenon. For its part, the study by Saeeda et al. (21) addresses various forms of NTD, including social, organizational, cultural, process, and people dimensions, based on both scientific and gray literature. Although it considers the concept of SD, its treatment

is more general and diffuse, without focusing specifically on community smells. Unlike the present work, it does not provide a detailed classification of community smells or a targeted analysis of their effects, but rather a panoramic view of the sources and strategies associated with NTD as a whole.

Table 1. Comparative analysis of primary studies

Aspects	Espinosa et al. (17)	Saeeda et al. (21)	Raza et al. (38)	Tahsin et al. (39)	Raza et al. (40)	The present study
Year	2024	2024	2022	2023	2021	2024
Period	2013-2022	2014-2022	2008-2019	2015-2022	2000-2020	2017-2024
# primary studies	25	40	46	21	306	48
Target	Define the concept of community smells, describe their properties, and identify their relationship as a source of SD.	Analyze NTD by identifying its causes, effects, and mitigation strategies to reduce its impact on software development.	Identify studies on the evolution of STC and its impact on software development.	Systematically review community smells in software engineering, their classification, causes, effects, and mitigation.	To analyze the historical and conceptual development of STC through scientometric techniques and a critical review of the most significant studies in the area.	A literature review was conducted to provide a structured and critical evaluation of SD and community smells in software development teams.
Analysis	The properties, causes, effects, and mitigation approaches of community smells are discussed.	Identifies the most common causes of NTD, including poor communication, ineffective leadership, inadequate processes, and toxic organizational dynamics.	Techniques for measuring STC, such as the matrix approach and social network analysis, are identified and their positive impacts on team coordination, software quality, and development performance are highlighted.	Community smells are related to coordination, communication, and collaboration problems, which can generate SD. In addition, techniques for their detection, such as social network analysis and repository mining, are mentioned.	Identifies six major research topics, including community structure, coordination, GSD, and OSS. Shows the most influential authors, countries, and institutions.	Thirty-five community smells were identified, the most frequent being organizational silo, black cloud, lone wolf, bottleneck, and radio silence.

Research questions (RQ)	<p>Eight RQ were formulated focusing on community smells, addressing their definition, causes, effects, and mitigation strategies.</p>	<p>Three RQ are proposed to analyze the causes, effects, and mitigation strategies of NTD in the context of software development.</p>	<p>Seven RQs were formulated focusing on STC, addressing its components, evolution, the impact of its absence, data sources, measurement techniques, and factors influencing its implementation.</p>	<p>The study formulates five RQs that guide a qualitative analysis of the taxonomic classification, causes, consequences, mitigation strategies, and detection techniques of community smells.</p>	<p>The article does not formulate explicit RQs or structure its analysis around them.</p>	<p>In this first review, the study answers four of six RQs, focusing on the evolution of the concept, its definition, and the identification of community smells.</p>
Domains	<p>Thirty community smells originating from suboptimal sociotechnical decisions are identified.</p>	<p>NTD is classified according to the socio-technical hexagonal framework, which encompasses people, processes, culture, organization, and social aspects.</p>	<p>It addresses the STC domain by exploring its positive impact on team coordination and software quality.</p>	<p>This study directly relates community smells to the domains of communication, collaboration, coordination, social structure, and STC.</p>	<p>This article does not define or classify specific domains associated with SD. It focuses on the scientometric and thematic analysis of the concept of STC.</p>	<p>Communication, coordination, collaboration, social structure, and socio-technical congruence.</p>

Source: Author’s work

In comparison, the Raza et al. (38) focus on STC as a mechanism to improve coordination and prevent social problems. However, it does not directly address community smells or SD, and its analysis is limited to the impact of STC on team dynamics, which marks an important difference from the approach of the present article. On the other hand, Tahsin et al. (39) present a rigorous systematic review that explicitly links community smells to SD, classifying causes and effects from a global perspective. Unlike the present study, their approach is more theoretical and general, without delving into organizational contexts or the psychosocial consequences of community smells. Finally, Raza et al. (40) offer a scientometric and conceptual review focused on the evolution of the concept of STC. Unlike the present work, it does not identify domains or analyze community smells; its objective is oriented toward tracing the research trajectory of the concept, without considering its social effects or its relationship with SD.

Metodology

The purpose of this article is to clarify the meaning of the terms “community smells” and “social debt,” as well as the causes that produce them, their characteristics, and their effects on CSD. This was achieved by reviewing and classifying 44 primary articles, which were identified through the application of inclusion criteria (IC) and exclusion criteria (EC). To carry out this study, an SLR was performed based on the procedure suggested by Kitchenham et al. (41).

In addition, the approach proposed by Basili and Caldiera (42), Goal-Question-Metric (GQM), was used to establish the objectives and research questions. This approach suggests three levels: (i) conceptual level, which establishes objectives to identify the purpose of the SLR, determining its scope and expected benefits; (ii) operational level, in which questions are designed based on the objectives of the conceptual level, these questions are fundamental to defining, organizing, and describing the main characteristics of the identified and selected items, ensuring they are related to the main research topic; and (iii) quantitative level, in which metrics are established to quantify the results, although in this study this level will not be applied. Figure 1 shows the actions carried out during the development of the SLR.

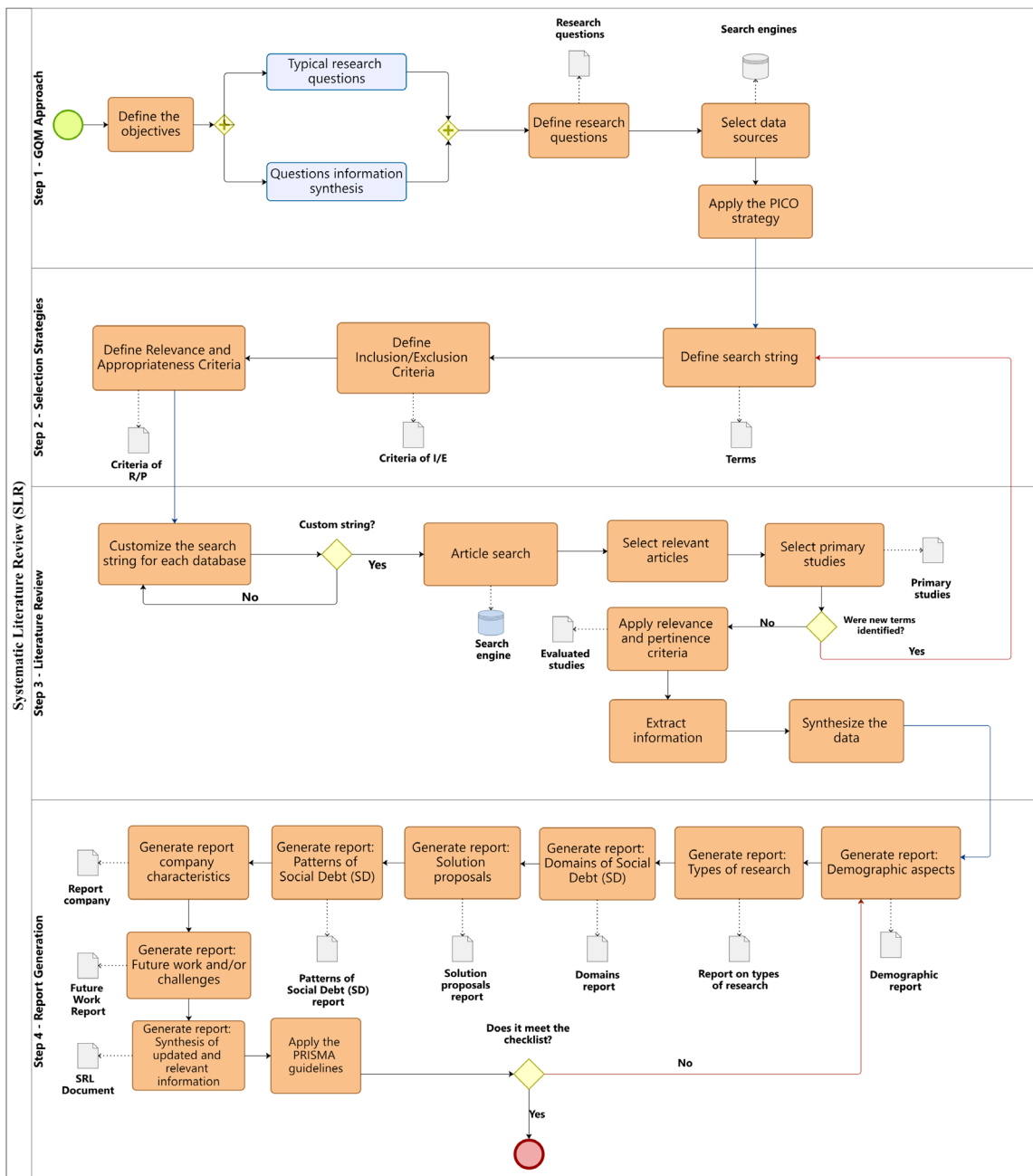


Figure 1. RSL protocol (41)

The following is a detailed explanation of the development of each of the stages listed in (Fig.1), and provides the structured framework used to carry out the SLR.

Stage 1 – GQM Approach: The activities developed in this stage are detailed below:

Establish the research objectives: These allow clarifying the specific aspects to be investigated, facilitating the focus on the topic. For this study, four research objectives were established: (i) to analyze studies on SD and community smells in CSD, (ii) to evaluate the quality and research methods to identify knowledge gaps, (iii) to systematize the current state of academic and professional proposals in the area, and (iv) to classify trends and future work.

Define research questions: This SLR addressed four of the eleven RQs that were originally formulated for the study, each linked to a research objective: two aimed at identifying demographic aspects (RQ1–RQ2), one to classify the type of research (RQ3), and one to synthesize up-to-date and relevant information (RQ4) on solutions related to the definition, causes, effects, and consequences of community smells in CSD. These questions, listed in Table 2, seek to: (a) identify conferences and scientific journals where the primary studies were published, (b) recognize leaders, countries, and institutions prominent in the study of community smells and SD, (c) organize the types of research according to the classification scheme established by Wieringa et al. (43), covering categories such as research, solution proposal, validation, philosophical articles, opinion papers, or personal experience articles, (d) clearly define the concept, types, and evolution of community smells, and (e) analyze their accumulation and effects on CSD.

Table 2. Research questions

Id question	Category	Question	Motivation
RQ1	Demographic aspects	What is the temporal distribution of scientific production on community smells in CSD?	This question allows the identification of trends and the evolution of academic interest in the subject, facilitating the analysis of the maturity of the field.
RQ2		What are the highest-impact journals and conferences that have published research on community smells in the context of software development?	Knowing the most relevant publication channels helps to delimit the key reference sources and recognize the academic legitimacy of the subject.
RQ3	Type of Research	What are the types of research that predominate in community smells studies in CSD?	Classify the different types of research evidenced according to the classification scheme proposed by Wieringa et al. (44,45), research, validation, solution proposal, philosophical articles, opinion papers, or personal experience articles.

RQ4	Summary of updated and relevant information	How do different authors conceptualize and define the term community smells in the context of CSD?	Defining the concept of community smells and its types in the context of software development allows understanding its evolution, the domains in which it occurs, and its manifestation within the development communities, thus facilitating greater clarity in its analysis and interpretation.
-----	---	--	---

Source: Author’s own work

Selecting data sources: The purpose of this phase is to ensure the inclusion of high-quality information, guaranteeing that the selected studies directly address the research questions. For this reason, the following databases were identified and selected: Google Scholar, ACM, Web of Science, Scopus, ScienceDirect, SpringerLink, and IEEE Xplore.

Stage 2 – Selection strategies. This stage includes:

Defining the search string, aimed at gathering relevant studies and avoiding bias. For this purpose, a combination of logical operators “AND” and “OR” was used, to control the inclusion of relevant terms in the results. Table 3 presents the basic search string executed to classify the primary articles, considering the period from 2017 to 2024, and adjusting the filters according to each search engine. The choice of this time window was based on the significant increase in publications on SD starting from 2017. It is important to note that the first article on this topic was published in 2013, entitled “What is social debt in software engineering?” (46) , This article focused on the concept of SD, some of its effects, and the technical aspects affected by its accumulation. In 2015, two additional articles were published focusing on the impact of SD in companies and the development of a framework called DAHLIA, under the titles “Social debt in software engineering: insights from industry” (25) y “When software architecture leads to social debt” (47). Subsequently, new results and updates to the framework were published in 2016 and considered as relevant articles in this paper. Therefore, based on the changes, publications, and updates, the selected time window was considered relevant.

Table 3. Search string

Search string.
(“social debt” OR “community smells”) AND (“agile software development” OR “agile approach” OR “agile development”).

Source: Author’s work

Define inclusion (IC) and exclusion (EC) criteria: During the execution stage, one of the investigators defined the criteria necessary to include or exclude studies (IC and EC), to narrow the search and ensure the relevance of the results. These results were subsequently quantified. The second

investigator then reviewed these criteria and provided comments for improvement. Thanks to these guidelines, it was possible to identify relevant studies that addressed the objectives and research questions. Table 4 details the criteria applied during the SLR. Their implementation optimized the evaluation process, allowing the selection of 353 studies considered relevant to the research. This selection was based on a careful review of the title, abstract, and results of each paper, prioritizing those that mainly addressed SD and community smells.

Table 4. Inclusion and exclusion criteria

Id IC	Inclusion criteria	EC Id	Exclusion criteria
IC1	Studies whose main topic is community smells or SD in software development.	EC1	Duplicate studies (considering the first study identified).
IC2	Studies whose subject matter is related to community smells or SD in software development.	EC2	Studies where the research topic is superficially approached.
IC3	Studies dealing with topics related to community smells or SD.	EC3	Studies outside the range in years between 2017 and 2024.
IC4	Studies published in high-impact peer-reviewed journals, congresses, or conferences.	EC4	Discussion-type studies or studies available only as presentations or abstracts.
IC5	Authors who have carried out the most studies in this field.	EC5	Studies that are books or book chapters.

Source: Author’s work

Define relevance and pertinence criteria: To assess the relevance and pertinence of primary studies related to community smells or SD in CSD, the approach of Kitchenham et al. (41), s was adapted, and 11 evaluation criteria were defined and applied using a scale of three values (-1, 0, +1): a score of +1 indicated that the study explicitly addressed the evaluated question; 0.5, that it partially addressed it; and 0, that it did not consider it. This evaluation allowed the classification of the articles according to their contribution, without implying their elimination from the analysis. The criteria applied are detailed in the Annex, Table 8.

Stage 3 – Literature review. The following steps were considered in this stage:

Article search: To identify relevant articles, 7 iterations were carried out, one for each database: Google Scholar, ACM, Web of Science, Scopus, ScienceDirect, SpringerLink, and IEEE Xplore. This process involved adapting the basic search string in each search engine. A total of 803 articles were obtained as search results, as presented in Table 5

Table 5. Search results

Id	Search source	Found articles	Relevant articles	Primary articles
1	Google Scholar	349	14	10
2	ACM Digital Library	31	7	7
3	Web of Science	183	182	1
4	Scopus	129	104	8
5	Science Direct	50	26	7
6	Springer Link	48	13	5
7	IEEEExplore	13	7	6
Total		803	353	44

Source: Author’s work

Select relevant articles: During this phase, the IC and EC were applied. The EC included the elimination of duplicate articles, books, or book chapters, as well as studies that superficially addressed SD and community smells. For this purpose, the title, abstract, and results of each article were reviewed. On the other hand, the inclusion criteria considered studies whose main topic was SD or community smells in the context of software development. As a result of this process, 353 relevant articles were identified, as presented in Table 5.

Selecting primary articles: In this stage, the review of relevant articles was carried out. During this process, the contribution of each article to the formulated research questions was evaluated; additionally, new keywords were identified, allowing for adjustments to the search string. After an exhaustive evaluation, 44 articles were selected as primary studies, which are highlighted in Table 6.

Table 6. Summary of the list of primary articles resulting from the search.

Id	Article	NC	Year	Ref
PS1	The influence of Technical Debt on software developer morale	61	2020	(48)
PS2	The use of incentives to promote technical debt management	14	2022	(49)
PS3	Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations	91	2018	(50)
PS4	Learning to detect community smells in open-source software projects	41	2020	(51)
PS5	Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach	42	2021	(52)
PS6	Impacts of software community patterns on process and product: An empirical study	14	2022	(53)
PS7	Refactoring Community Smells in the Wild: The Practitioner's Field Manual	39	2020	(54)
PS8	On the detection of community smells using genetic programming-based ensemble classifier chain	37	2020	(55)
PS9	Gender diversity and women in software teams: How do they affect community smells?	142	2019	(56)
PS10	Splicing Community Patterns and Smells: A Preliminary Study	23	2020	(57)
PS11	Breaking one barrier at a time: How women developers cope in a men-dominated industry	24	2021	(58)
PS12	Understanding Community Smells Variability: A Statistical Approach	25	2021	(59)
PS13	Exploring Community Smells in Open-Source: An Automated Approach	92	2021	(60)
PS14	Software Architecture Social Debt: Managing the Incommunicability Factor	34	2019	(61)
PS15	How Do Community Smells Influence Code Smells?	25	2018	(62)
PS16	Predicting Community Smells' Occurrence on Individual Developers by Sentiments	18	2021	(63)
PS17	Gender Diversity and Community Smells: Insights from the Trenches	43	2020	(64)
PS18	Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells?	132	2021	(65)
PS19	An empirical study on the effect of community smells on bug prediction	14	2021	(66)



PS20	Splicing Community and Software Architecture Smells in Agile Teams: An industrial Study	15	2019	(67)
PS21	Understanding Social Debt in Software Engineering	4	2021	(68)
PS22	"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	7	2021	(69)
PS23	Understanding the Relationship between Missing Link Community Smell and Fix-inducing Changes	4	2021	(70)
PS24	Gender Diversity and Community Smells: A Double-Replication Study on Brazilian Software Teams	10	2022	(71)
PS25	Good Fences Make Good Neighbours? On the Impact of Cultural and Geographical Dispersion on Community Smells	16	2022	(72)
PS26	In Search of Socio-Technical Congruence: A Large-Scale Longitudinal Study	23	2021	(73)
PS27	On the Relationship Between the Developer's Perceptible Race and Ethnicity and the Evaluation of Contributions in OSS	34	2022	(74)
PS28	Including Everyone, Everywhere: Understanding Opportunities and Challenges of Geographic Gender-Inclusion in OSS	50	2022	(75)
PS29	A Preliminary Study on the Assignment of GitHub Issues to Issue Commenters and the Relationship with Social Smells	2	2022	(76)
PS30	A study on correlations between architectural smells and design patterns	12	2021	(77)
PS31	Architectural design decisions that incur technical debt—An industrial case study	19	2021	(78)
PS32	Women in Agile: The Impact of Organizational Support for Womens Advancement on Teamwork Quality and Performance in Agile Software Development Teams	22	2021	(79)
PS33	Inter-team communication in large-scale co-located software engineering: a case study	17	2022	(80)
PS34	Technical-, Social- and Process Debt in Large-Scale Agile: An Exploratory Case-Study	34	2019	(81)
PS35	Discovering community patterns in open-source: a systematic approach and its evaluation	86	2019	(82)
PS36	Dimensions of Consistency in GSD: Social Factors, Structures and Interactions	6	2020	(83)

PS37	Refactoring Recommendations Based on the Optimization of Socio-Technical Congruence	4	2020	(84)
PS38	Understanding the involvement of developers in missing link community smell: An exploratory study on apache projects	7	2020	(85)
PS39	Are we working well with others? How the multi team systems impact software quality	6	2018	(86)
PS40	Revealing social debt with the CAFFEA framework: An antidote to architectural debt	13	2017	(87)
PS41	Emotional Labor of Software Engineers	27	2017	(88)
PS42	Social Debt Analytics for Improving the Management of Software Evolution Tasks	7	2017	(89)
PS43	Psychological Safety, Leadership and Non-Technical Debt in Large Scale Agile Software Development	1	2023	(90)
PS44	Locating community smells in software development processes using higher-order network centralities	35	2023	(91)

Acronyms. PS: Primary Study identifier. NC: Number of citations according to Google Scholar. Ref: Reference. PS: Primary Studies.

Source: Author’s work

Apply relevance and pertinence criteria: Once the criteria were defined, they were applied to the primary studies by adding up the individual values, as established in the Annex, Table 9, and to the studies listed in Table 6. Two percent (PS18) achieved the highest score (10 points), while 7 percent obtained 0 points, reflecting partial contributions with limitations in topic coverage. On the other hand, 4% (PS6, PS9) recorded the lowest score (-2 points). It is important to note that a low score does not imply low quality of the study, but only partial compliance with the criteria applied in this systematic review.

Extract information: This study compiled the most relevant elements of each article reviewed, including title, abstract, year, and type of publication, database, type of research, conclusions, results, and contributions. The objectives and research questions formulated were also considered.

Synthesize the data: In this phase, the information obtained from the reviewed studies was integrated, summarized, and presented in a coherent and structured manner. A summary was then prepared highlighting the key aspects and main conclusions derived from the literature analyzed.

Stage 4 – Report generation. In this stage, the research questions were answered through the preparation of reports, which were presented considering the three categories established for the research questions: demographic aspects, types of research, and synthesis of updated and relevant information (see Table 2). To facilitate the reading of this article in later sections and the identification of the primary studies, together with the rest of the references used, the identifier assigned to each primary study (PS) will be used, as presented in Table 6.

Results and discussion

After completing the search procedure and analyzing the identified primary studies, answers to the following research questions are provided.

Demographic aspects

In this section, the main objective is to provide relevant answers to research questions RQ1 – RQ2, listed in Table 1 and based on the analysis performed.

RQ1. What is the temporal distribution of scientific production on community smells in software development communities?

The classification and analysis of the selected primary studies made it possible to identify the frequency and type of publications related to the topics of community smells and SD. (Fig. 2) shows that the highest number of publications was recorded in 2021, representing 32% of the total, with 9 articles published in scientific journals and 5 articles in conferences in that year. In 2022, 18% of the publications were recorded, with 5 journal articles and 3 conference articles. For its part, 2020 contributed another 18%, distributed among 3 conference papers, 3 journal articles, and 2 workshop articles. In 2019, the production reached 11%, with 2 journal articles, 2 conference articles, and 1 workshop article. On the other hand, the year 2023 contributed 7%, distributed in 2 journal articles and 1 workshop article. The years 2017 and 2018 each contributed 7% of the publications: in 2017, there were 3 workshop publications, while in 2018, there were 2 journal articles and 1 workshop article. Finally, as of January 2024, no publications were identified.

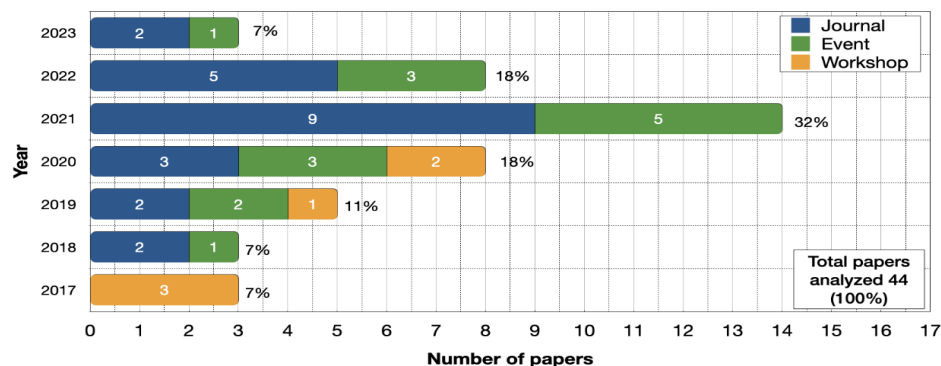


Figure 2. Type and number of publications by year

RQ2. What are the highest-impact journals and conferences that have published research on community smells in the context of software development?

The notable increase in the number of publications in 2021 indicates that these topics became more relevant among researchers and the scientific community, which motivated the dissemination of

findings, future work, and/or the identification of research gaps. Publishing in scientific journals and participating in academic events are key strategies for sharing knowledge, receiving feedback to improve studies, and increasing the visibility of research both within the scientific community and among the interested public.

A total of 23 articles were published in journals specialized in Software Engineering, covering a variety of topics related to this discipline, including the use of agile methods in globally distributed teams, technical debt management, artificial intelligence and machine learning in Software Engineering, and project management of software, as well as tools, environments, and development models. The remaining articles were presented at scientific events in Software Engineering, addressing topics of great importance such as the ethical responsibility of professionals and ethical dilemmas in the practice of this field, data privacy, equity and inclusion, cybersecurity, and their impact on employment and the economy. Research and projects that use Software Engineering to address social challenges and improve the quality of life were also discussed.

As shown in (Fig. 3), the journals “Journal of Systems and Software” account for 30.45% of the publications, which is equivalent to 7 articles, and “IEEE Transactions on Software Engineering” with 21.7%, corresponding to 5 publications, are the journals with the highest number of scientific articles published on SD and community smells. On the other hand, the event with the highest number of presentations is the “Proceedings of the Annual Hawaii International Conference on System Sciences” with 18.75%, corresponding to 3 published articles, as illustrated in Figure 4. The workshop with the highest number of publications is “CEUR Workshop Proceedings”, representing 50% of the total number of articles published in workshops, equivalent to 3 published articles, as shown in Figure 5.

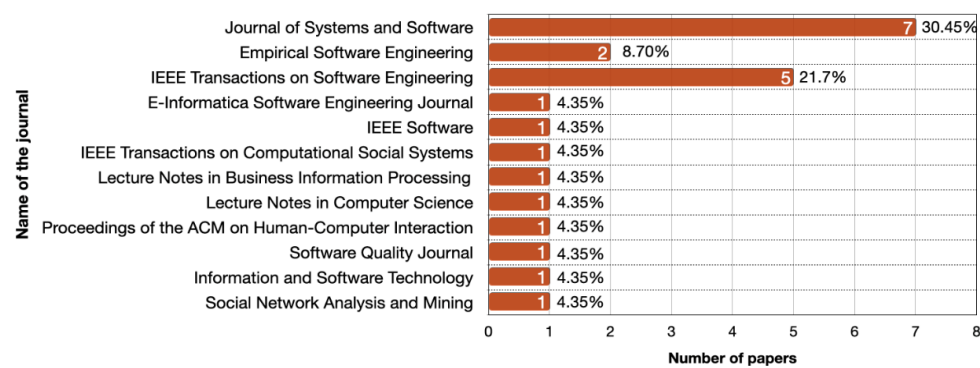


Figure 3. Number of papers per journal

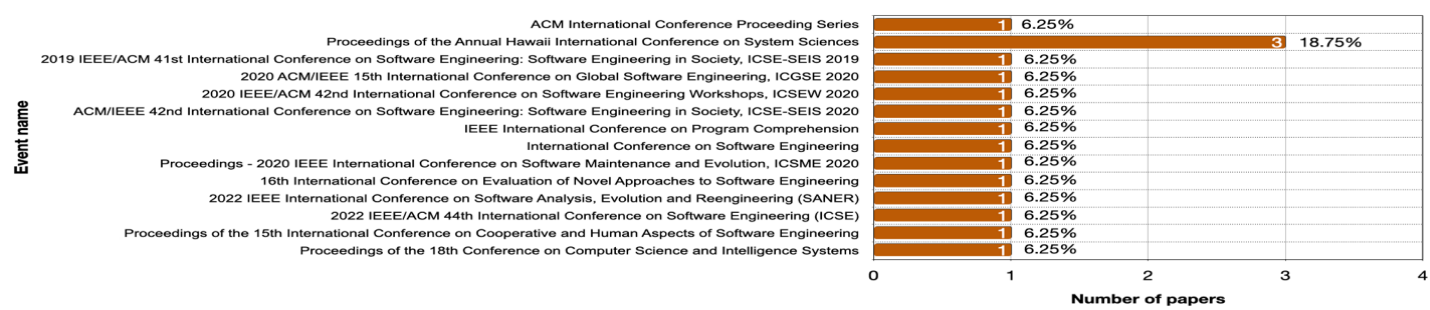


Figure 4. Number of papers per event



Figure 5. Number of papers per workshop

Types of research

This section answers the following research question:

RQ3. What are the types of research that predominate in studies on community smells in software development communities?

To classify the 44 primary studies identified, the six types of research proposed by Wieringa et al. (92) were used as a reference: (i) validation, (ii) evaluation, (iii) solution proposal, (iv) philosophical articles, (v) opinion articles, and (vi) personal experience articles. According to this classification and based on (Fig. 6), 2 studies (5%) of validation were identified. This category includes research that examines characteristics of a solution proposal not yet implemented. The topics addressed in these studies include different dimensions of Software Engineering, such as gender diversity, community smells, socio-technical congruence, and cultural and geographic dispersion.

Regarding the type of evaluation research, it was found that 45% (corresponding to 20 studies) belong to this category. These papers were classified as evaluation research because they generated new knowledge through detailed and novel research methodologies. These articles include case studies whose purpose is to collect quantitative and/or qualitative data on the factors and causes that drive and mitigate community smells in CSD. Likewise, studies were identified that rely on data collection techniques such as surveys, interviews, and observation to determine the occurrence of community smells in PS14–PS16 work teams.

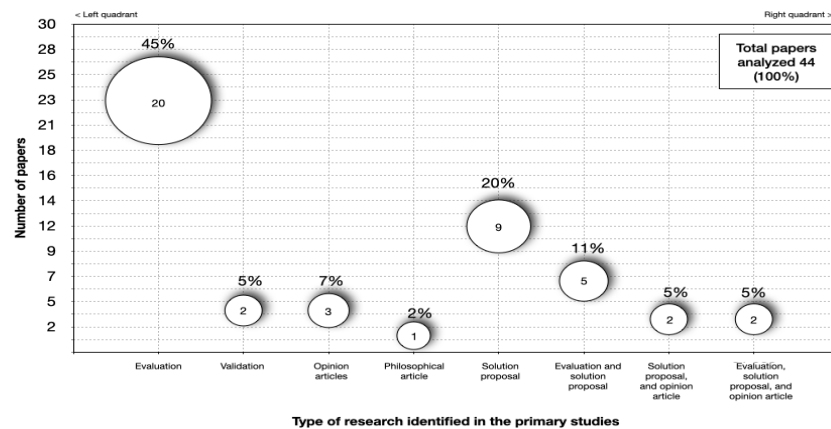


Figure 6. Types of Research

On the other hand, 20% of the reviewed studies, represented by 9 articles, were classified as solution proposals. This category is based on the originality of the proposals, their applicability as a solution, and the significant improvements they offer compared to previous approaches. Among the solution proposals identified in these articles, several addressed the use of natural language processing (NLP) techniques to detect the presence of community smells by identifying emotions or attitudes expressed by professionals in text, such as frustration, anger, distrust, stress, anxiety, distress, and irritability, among others. These solutions also include statistical methods to identify sentiments in the code, through comments or incident records from members of a PS16 development community. Additionally, the relationship between community smells and code smells is examined. In PS15, regression analysis is used to examine the relationship between community smells and the duration of code smells, while PS17 employs a regression model to analyze the relationship between community smells and the intensity of code smells. Furthermore, an automated tool called CodeFace4Smells was developed, which allows for the identification of 4 community smells: lone wolf, organizational silo, black cloud, and bottleneck. This tool reflects socio-technical problems that have proven detrimental to both the fields of Software Engineering and organizational research. The tool also uses a heuristic to identify coordination between two developers (PS4 – PS6, PS13, and PS23). In addition, 1 article (2%) was classified as philosophical, as it presents the author's perspective on the results of previous research, evaluating its validity or questioning it. Likewise, three articles (7%) were classified as opinion articles, based on the author's viewpoint, experience, and contributions to the debate on the topic. It is important to note that two articles (5%) were classified simultaneously as evaluation, solution proposal, and opinion articles. Another two articles (5%) were classified as both solution proposals and opinion articles, and 5 articles (11%) were included in both evaluation and solution proposal categories. Finally, no articles were identified in the primary studies that fit the category of personal experience research.

The classification of the 44 primary studies according to the six types of research proposed by Wieringa et al. (92) revealed remarkable diversity in the approaches used. Most studies were

organized in the categories of evaluation (45%) and solution proposal (20%), reflecting a marked interest in generating new knowledge and introducing innovations within Software Engineering. It is also evident that the scientific community has prioritized knowledge generation through the application of detailed and novel methodologies, which contributes to the validation and improvement of theories and practices relevant to the discipline, with a particular emphasis on software development and the importance of human talent over processes and tools. Furthermore, some studies adopt a multidimensional approach, addressing different aspects of the problem under investigation and proposing comprehensive solutions.

Summary of updated and relevant information

This section answers the following research question:

RQ4. How do different authors conceptualize and define the term “community smell” in the context of software development communities?

The SLR is a rigorous approach that facilitates the clarification of concepts related to the topic under study, in this case, focused on SD and community smells. The following is a synthesis of the concepts and contributions provided by different authors on the subject. According to PS11, PS12, PS13, and PS14, the concept of community smells can be considered as socio-technical anti-patterns and a source of social debt that affects the development processes, results, and organization of CSD. It is also important to highlight the definition of community smells established by Palomba and Tamburri in PS5, where it is suggested that “...community smells are related to all kinds of unpleasant phenomena, for example: employee turnover, poor architectural decisions that often go beyond causing technical problems in the code; they can also cause organizational confusion or even a decrease in the organizational stability of software projects and/or project failure...”. These smells are also associated with the behaviors of members of the software development community, as well as with policies and misalignment between organizational goals and/or processes.

Additionally, other authors associate community smells with the causes that lead to different types of debt, such as process debt in PS34 and technical debt in PS3 and PS31. On the other hand, SD manifests itself in technical problems reflected in software artifacts such as incomprehensible documentation and incomplete software components (PS14, PS43). Table 7 defines 35 community smells identified in the analyzed literature. If not properly managed, these smells can lead to the occurrence of SD in software development teams, as they are considered possible causes of SD. It was also observed that the community smells with the greatest presence in CSD are: organizational silo (mentioned in 20 primary studies), followed by black cloud and radio silence (each identified in 12 primary studies). Notably, some primary studies, such as PS7, PS18, PS26, and PS13, mention radio silence and bottleneck as the same definition; however, in this SLR, they are treated as two distinct community smells. Furthermore, it is important to highlight the observation by PS5 regarding “...that predicting community smells within a community before their manifestation over software project timelines can be critical to anticipate suboptimal organizational and sociotechnical conditions before they become unmanageable...”. Table 7 below lists 10 of the 35 identified community smells. The complete list can be found in Annex A – Table 10.

Table 7. Community Smells in Software Development

CS Id	CS Name	Definition of Community Smell	Reference
1	Architecture by Osmosis (AO)	Occurs when architectural decisions are made based on incomplete or poorly managed information. Some community members use poorly defined channels or different standards to document changes, resulting in wasted time, difficulty tracking key decisions, and an unstable architecture.	AP26
2	Architectural Hood (AH)	Occurs when architecture teams work remotely or with little interaction with developers. This lack of direct contact hinders cooperation and prevents the proper resolution of implementation problems arising from architectural decisions.	AP26
3	Organizational Silo (OS)	Refers to the existence of areas within the developer community that communicate only through one or two representatives, limiting the flow of information and the effective integration of the team.	AP4 - AP13, AP16- AP19, AP23, AP29, AP32, AP35
4	Black Cloud (BC)	Arises when the organization does not offer conditions that favor communication and interaction among community members. This hinders the exchange of knowledge and experiences during software development.	AP5- AP7, AP9 - AP10, AP12-AP13, AP17-AP18, AP24
5	Lone Wolf (LW)	Manifests when certain members act independently, without considering their peers or coordinating efforts. This behavior can lead to unauthorized technical decisions, code errors, and project delays.	AP7, AP9, AP10, AP12, AP16 - AP18, AP21
6	Bottleneck (BN)	Occurs when the same person intervenes in all interactions between two or more subcommunities, hindering smooth communication. This generates production delays and a decrease in team efficiency.	AP6 - AP7, AP10, AP13, AP16, AP18

7	Radial Silence (RS)	Occurs when procedures within the community are too formal and structured, causing delays in the implementation of changes and wasting time due to complex and not very agile processes.	AP1, AP4, AP7, AP8, AP8, AP9, AP12, AP13, AP17, AP24, AP25 - AP32
8	Prima-donnas (PD)	Appears when some members refuse to accept changes proposed by others, as a result of an inefficient collaboration structure. This resistance negatively affects workflow and team cohesion.	AP8, AP16
9	Share villainy (SV)	Describes environments where sharing information or experiences is not considered a priority. This is often due to a lack of space, incentives, or organizational support to foster collaborative learning.	AP1, AP8
10	Organizational Skirmish (OSk)	Occurs when differences between organizational cultures make it difficult to coordinate and manage projects. These differences can impact productivity and cause significant delays.	AP8, (17)

Acronyms. Id: Community smell identifier. CS: Community smell.

Source: Author’s elaboration.

Discussion

This section presents the results obtained from the SLR data collection, aiming to provide a comprehensive overview of the current state of SD and community smells in software development. The discussion is organized into three main sections: (A) Main observations, where key factors contributing to SD and community smells are examined, future research areas are identified, benefits and possible solutions are presented, and the current state of the literature is analyzed; and (B) limitations of the review and analysis of the results.

Main observations

The conducted SLR made it possible to gather, analyze, understand, and deepen the concepts of community smells and SD in the context of software development teams. The main findings are presented below:

In software development communities, professionals require a high level of interpersonal and soft skills. The absence or deficiency of these skills is a risk factor that favors the emergence of community smells, reflecting underlying problems in work dynamics. These community smells can negatively affect productivity, software quality, and organizational climate, directly impacting the health and sustainability of the development communities to which the professionals belong.

Among the fundamental soft skills for professionals in software development communities are communication, collaboration, and coordination. Deficiencies in these competencies can generate misunderstandings, affect the technical quality of software, and deteriorate team productivity and cohesion. In this context, community smells emerge as indicators of underlying problems in interpersonal dynamics. Identifying these community smells makes it possible to recognize situations or professionals who may be generating tensions, enabling the implementation of preventive strategies that support the social and emotional well-being of teams. Rather than being causes for exclusion, these findings should be seen as opportunities for improvement, recognizing that appropriate interventions can strengthen both individuals and the collective, fostering healthier and more sustainable work environments.

To address problems such as community smells, the concept of sociotechnical congruence is used, understood as the effective alignment between social and technical aspects within an organization. This congruence implies that processes, technologies, and labor relations must be designed in an integrated manner, promoting both organizational efficiency and employee well-being and satisfaction. It is essential to ensure that interactions among team members are fluid and conducive to achieving project objectives.

Additionally, gender diversity is a relevant factor in mitigating community smells. As demonstrated in the reviewed literature, development environments that integrate diverse teams while maintaining professional competence foster the creation of environments conducive to the effective use of soft skills, such as communication, cooperation, and coordination among community members.

Limitations of the Systematic Review

Each stage involved in conducting this SLR, from the formulation of objectives to the collection and analysis of the information obtained, may have been affected by sampling flaws, particularly regarding the accuracy of the collected information. Some of these limitations are presented below:

Bias due to conceptual ambiguity and late reformulation of research objectives and questions: During the development of the SLR, a methodological limitation was identified, related to the initial formulation of objectives and research questions, as well as the broad scope of SD, which can be applied to multiple domains. This breadth generated ambiguity in the conceptual delimitation and affected the clarity in the selection and extraction of relevant studies. In particular, the lack of a precise definition of the relationship between SD and community smells in agile software development contexts hinders the identification of relevant sources and consistent analysis of findings. As a corrective measure, objectives and research questions were reformulated, and the search string was adjusted to ground the research specifically in the software development domain. This refinement increased the precision and relevance of the results, strengthening the validity of the analysis and the consistency of the findings obtained.

A possible bias identified in this SLR is related to restrictions applied during the selection of articles, particularly in terms of language and publication period. The search was limited to studies in English, as most research on SD in software development is published in that language; however,

this decision may have excluded relevant studies in other languages, thus reducing the diversity of the analysis. Likewise, the inclusion of articles published from 2017 onwards was established as a temporal criterion, under the premise that previous studies were updated in more recent works. Although these restrictions aimed to ensure the timeliness and quality of the sources, they also imply a potential omission bias. To mitigate this risk, foundational papers were verified through their updated versions, and preference was given to highly cited, peer-reviewed literature from recognized databases.

A limitation was identified in the information search process that may have affected the coverage and clarity of the topic under analysis. Although several databases were consulted, each possesses distinct characteristics and not all provide the same type of content, resulting in inconsistent findings. Initially, a general search string was used with the terms “social debt” and “community smells”, which retrieved numerous articles not directly related to software engineering. This hindered the study’s focus. To improve the relevance and precision of the results, the search was refined by including the terms “software development” and “agile software development,” enabling more effective filtering of pertinent studies and more accurate information for analysis.

One of the potential biases identified relates to the classification process of the selected articles. Although certain criteria were applied, they may not have been sufficiently rigorous, potentially leading to ambiguity in the categorization and affecting the overall perspective of the subject. This limitation could have influenced the interpretation of the results, particularly when grouping articles by types of contribution, approach, or domain. To address this bias, the established categories were reviewed, and cross-validation was conducted based on explicit article descriptions.

Conclusions

Between 2017 and 2024, forty-four primary studies were identified, reflecting a growing interest in understanding community smells within software development communities. These studies, which were mainly exploratory, allowed for the identification of a wide variety of methodological approaches and publication channels relevant to this field. They focused on the root causes of these smells and their association with deficiencies in socio-technical factors, such as communication, coordination, and collaboration, which are key in agile and distributed environments.

A total of thirty-five community smells were identified and analyzed, highlighting the frequent occurrence of organizational silos, bottlenecks, black clouds, and lone wolves. These factors disrupt communication flow, create critical points of dependency, foster team member isolation, and impede equitable access to information. Such smells have been linked to increased social, technical, and process debt, with negative consequences for software quality and team well-being.

One key finding is that community smells often go unnoticed, as they tend to become normalized within the work culture and are not typically perceived as an immediate threat. This invisibility hinders early detection and timely intervention. Consequently, it is recommended to advance the development of metrics and indicators for their detection, as well as to design contextualized mitigation strategies.

It is recognized that the social dimension of debt cannot be disentangled from technical debt, as both are inherently interconnected in development processes. The emotional and social well-being of team members thus becomes a decisive factor in sustaining the balance between productivity and long-term sustainability. In this regard, fostering collaborative environments, recognizing outstanding performance, enhancing technical and interpersonal skills, and encouraging active participation from all members contribute significantly to mitigating the incidence of community smells.

As future work, it is proposed to further investigate the remaining thirty community smells, identifying their causes, effects, and potential metrics, and documenting their characteristics from various conceptual perspectives. Additionally, it is considered essential to reduce terminological heterogeneity by designing a taxonomy that systematizes community smells and their relationship with the social dimension of debt in software engineering. This taxonomy will constitute both a conceptual and practical contribution, advancing the management of these phenomena in distributed and collaborative software development teams.

Acknowledgments

Professors Eydy Suárez Brieva and César Pardo Calvache wish to thank the Universidad Popular del Cesar and the Universidad del Cauca, where they currently serve as full professors, respectively. Likewise, students Jorge Enrique Bejarano Guar and Nicolle Daniela Urbano express their gratitude to the Universidad del Cauca, where they are currently enrolled in the Systems Engineering program.

CrediT authorship contribution statement

Conceptualization - Ideas: Eydy Suárez Brieva, Jorge Enrique Bejarano Guar, Nicole Daniela Urbano Palechor y César Jesús Pardo Calvache. **Data Curation:** Eydy Suárez Brieva, Jorge Enrique Bejarano Guar y Nicole Daniela Urbano Palechor. **Formal analysis:** Eydy Suárez Brieva y Jorge Enrique Bejarano Guar. **Investigation:** Eydy Suárez Brieva, Jorge Enrique Bejarano Guar y Nicole Daniela Urbano Palechor. **Methodology:** Eydy Suárez Brieva y Jorge Enrique Bejarano Guar. **Project Management:** Eydy Suárez Brieva y César Jesús Pardo Calvache. **Resources:** Eydy Suárez Brieva, Jorge Enrique Bejarano Guar y Nicole Daniela Urbano Palechor. **Software:** Eydy Suárez Brieva, Jorge Enrique Bejarano Guar y Nicole Daniela Urbano Palechor. **Supervision:** Eydy Suárez Brieva y César Jesús Pardo Calvache. **Validation:** Eydy Suárez Brieva y César Jesús Pardo Calvache. **Writing - original draft - Preparation:** Eydy Suárez Brieva y Jorge Enrique Bejarano Guar. **Writing - revision and editing - Preparation:** Eydy Suárez Brieva, Jorge Enrique Bejarano Guar, Nicole Daniela Urbano Palechor y César Jesús Pardo Calvache.

Financing: does not declare. **Conflict of interest:** does not declare. **Ethical aspect:** does not declare.

References

1. Gómez Fuentes M del C, Cervantes Ojeda J, González Pérez P. Fundamentos de Ingeniería de Software 1st ed. UAM UC, editor. Ciudad de México: Universidad Autónoma Metropolitana; 2019, 1–277 p. Available from: <https://shorturl.at/gjGIL>
2. Yilmaz M, O'Connor R V., Clarke P. Effective Social Productivity Measurements during Software Development - An Empirical Study. International Journal of Software Engineering and Knowledge Engineering. 2016 Apr 1;26(3):457–90. <https://doi.org/10.1142/S0218194016500194>
3. Galster M, Tamburri DA, Kazman R. Towards Understanding the Social and Organizational Dimensions of Software Architecting. ACM SIGSOFT Software Engineering Notes. 2017;42(3):24–5. <https://doi.org/10.1145/3127360.3127374>
4. Sievi-Korte O, Fagerholm F, Systä K, Mikkonen T. Dimensions of Consistency in GSD: Social Factors, Structures and Interactions. Proceedings Lecture Notes in Computer Science. 2020;12562:315–30. https://doi.org/10.1007/978-3-030-64148-1_20
5. Brieva ES, Pardo C, Villarreal V. Deuda Social y Riesgos psicosociales en entornos de desarrollo de software ágil: Análisis preliminar de una Revisión Sistemática de la Literatura. In: 2024 IEEE VII Congreso Internacional en Inteligencia Ambiental, Ingeniería de Software y Salud Electrónica y Móvil (AmITIC). 2024. p. 1–8. <https://doi.org/10.1109/AmITIC62658.2024.10747594>
6. Vizcaíno A, García F, Piattini M. Visión General del Desarrollo Global de Software. International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC) 2014;1(1):8–22. www.ijisebc.com
7. Jiménez M, Piattini M, Vizcaíno A. Challenges and improvements in distributed software development: A systematic review. Advances in Software Engineering. 2009; 1–14. <https://doi.org/10.1155/2009/710971>
8. Vizcaíno A, Valencia D, Soto JP, García-Mundo L, Piattini M. ¿Qué desafíos presenta el desarrollo global del software? Aprende jugando. In: XXI Jornadas de Ingeniería del Software y Bases de Datos. México; 2016. p. 605–8. https://investigadores.unison.mx/ws/portalfiles/portal/8020882/CEDI_2016_paper_47.pdf
9. García GD, Pardo Calvache CJ, Rodríguez FJA. Society 5.0 and Soft Skills in Agile Global Software Development. Revista Iberoamericana de Tecnologías del Aprendizaje. 2022 May 1;17(2):197–207. <https://doi.org/10.1109/RITA.2022.3166966>
10. Mens T. An ecosystemic and socio-technical view on software maintenance and evolution. In: Proceedings - 2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016. Institute of Electrical and Electronics Engineers Inc.; 2017. p. 1–8. <https://doi.org/10.1109/ICSME.2016.19>
11. Gote C, Perri V, Zingg C, Casiraghi G, Arzig C, von Gernler A, et al. Locating community smells in software development processes using higher-order network centralities. Soc Netw Anal Min. 2023 Oct 1;13(1):1–28. <https://doi.org/10.1007/s13278-023-01120-w>
12. Olsson J, Risfelt E, Besker T, Martini A, Torkar R. Measuring affective states from technical debt: A psychoempirical software engineering experiment. Empir Softw Eng. 2021 Sep 1;26(5). <https://doi.org/10.1007/s10664-021-09998-w>
13. Tamburri DA, Milano D, Kazman R. The Architect's Role in Community Shepherdling. <https://doi.org/10.1109/MS.2016.144>
14. Tamburri DA, Palomba F, Kazman R. Exploring Community Smells in Open-Source: An automated approach. IEEE Transactions on Software Engineering. 2017;14(8):630–52. <https://doi.org/10.1109/TSE.2019.2901490>

15. Palomba F, Tamburri DA. Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. *Journal of Systems and Software*. 2021;171. <https://doi.org/10.1016/j.jss.2020.110847>
16. Huang Z, Shao Z, Fan G, Gao J, Zhou Z, Yang K, et al. Predicting Community Smells' Occurrence on individual developers by sentiments. In: ICPC, editor. 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC). Madrid : IEEE; 2021. p. 230–41. <https://doi.org/10.1109/ICPC52881.2021.00030>
17. Caballero-Espinosa E, Carver JC, Stowers K. Community smells—The sources of social debt: A systematic literature review. *Inf Softw Technol*. 2023;153. <https://doi.org/10.1016/j.infsof.2022.107078>
18. Catolino G, Palomba F, Tamburri DA, Serebrenik A. Understanding Community Smells Variability: A Statistical Approach. *Proceedings - International Conference on Software Engineering*. 2021; 77–86. <https://doi.org/10.1109/ICSE-SEIS52602.2021.00017>
19. Almarimi N, Ouni A, Chouchen M, Mkaouer MW. Improving the detection of community smells through socio-technical and sentiment analysis. *Journal of Software: Evolution and Process*. 2022;35 <https://doi.org/10.1002/smr.2505>
20. Saeeda H, Ovais Ahmad M, Gustavsson T. Navigating social debt and its link with technical debt in large-scale agile software development projects. *Software Quality Journal*. 2024;32:1581–1613 <https://doi.org/10.1007/s11219-024-09688-y>
21. Saeeda H, Ovais Ahamd M, Gustavsson T. A Multivocal Literature Review on Non-Technical Debt in Software Development: An Insight into Process, Social, People, Organizational, and Culture Debt. *e-Informatica Software Engineering Journal*. 2024;18(1):240101. <https://doi.org/10.37190/e-Inf240101>
22. Palomba F, Serebrenik A, Zaidman A. Social Debt Analytics for improving the management of Software evolution tasks. In: *CEUR Workshop Proceedings*, editor. Belgian Netherlands Software evolution Symposium. Belgium; 2023;19–21. <https://shorturl.at/ahoX7>
23. Moreno Jiménez, Bernardo. "Factores y riesgos laborales psicosociales: conceptualización, historia y cambios actuales." *Medicina y Seguridad del trabajo*. 2011; 57: 4-19.
24. Organización Internacional del Trabajo. La organización del trabajo y los riesgos psicosociales: una mirada de género. Organización Internacional del Trabajo. San José; 2013 https://scielo.isciii.es/scielo.php?pid=S0465-546X2011000500002&script=sci_arttext&tlng=en
25. Tamburri DA, Kruchten P, Lago P, Vliet H van. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*. 2015;6(1). <https://doi.org/10.1186/s13174-015-0024-6>
26. De Stefano M, Iannone E, Pecorelli F, Tamburri DA. Impacts of software community patterns on process and product: An empirical study. *Sci Comput Program*. 2022; 214:102731. <https://doi.org/10.1016/j.scico.2021.102731>
27. Lambiasi S, Catolino G, Tamburri DA, Serebrenik A, Palomba F, Ferrucci F. 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*. New York, NY, USA: Association for Computing Machinery; 2022. p. 67–78. <https://doi.org/10.1145/3510458.3513015>
28. Tamburri DA. Software Architecture Social Debt: Managing the Incommunicability Factor. *IEEE Trans Comput Soc Syst*. 2019;6(1):20–37. <https://doi.org/10.1109/TCSS.2018.2886433>

29. Dreesen T, Hennel P, Rosenkranz C, Kude T. 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. Proceedings of the Annual Hawaii International Conference on System Sciences. 2021; 6826–35. <https://doi.org/10.24251/HICSS.2021.818>
30. Mumtaz H, Paradis C, Palomba F, Tamburri DA, Kazman R, Blincoe K. A Preliminary Study on the Assignment of GitHub Issues to Issue Commenters and the Relationship with Social Smells. Vol. 1, Proceedings - 15th International Conference on Cooperative and Human Aspects of Software Engineering, CHASE 2022. Association for Computing Machinery. 2022; 61–65 <https://doi.org/10.1145/3528579.3529181>
31. Lambiase S, Catolino G, Tamburri DA, Serebrenik A, Palomba F, Ferrucci F. Good fences make good neighbours? 2022;67–78. <https://doi.org/10.1109/ICSE-SEIS55304.2022.9793992>
32. Pontificia Universidad Javeriana, Ministerio de la Protección Social. Batería de instrumentos para la evaluación de factores de riesgo psicosocial. 2010 <https://shorturl.at/kuE49>
33. R. S. Sangwan, K. W. Jablokow and J. F. DeFranco, "Asynchronous Collaboration: Bridging the Cognitive Distance in Global Software Development Projects," in IEEE Transactions on Professional Communication. 2020; 63(4):361-371 <https://doi.org/10.1109/TPC.2020.3029674>
34. Almarimi N, Ouni A, Mkaouer MW. Learning to detect community smells in open source software projects. Knowl Based Syst. 2020 Sep 27;204. <https://doi.org/10.1016/j.knosys.2020.106201>
35. Palomba F, Tamburri DA. Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. Journal of Systems and Software. 2021;171. <https://doi.org/10.1016/j.jss.2020.110847>
36. Catolino G, Palomba F, Tamburri DA, Serebrenik A, Ferrucci F. Refactoring Community Smells in the Wild: The Practitioner's Field Manual. In: Proceedings - 2020 ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS 2020. 2020; 25–34. <https://doi.org/10.1145/3377815.3381380>
37. Palomba F, Tamburri DA. Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. Journal of Systems and Software. 2021;171:110847. <https://doi.org/10.1016/j.jss.2020.110847>
38. Raza B, Ahmad R, Nasir MH, Fauzi SS, Raza MA. Assessing the impact of socio-technical congruence in software development: a systematic literature review. KJS 2021;49(1). <https://journalskuwait.org/kjs/index.php/KJS/article/view/9240>
39. Tahsin N, Rauf A, Manzoor A, Anwer MU. Community smells—The sources of social debt: A systematic literature review. Systematic Literature Review and Meta-analysis Research. 2023;3(4). Available from: <https://doi.org/10.54480/slr.m.v3i4.51>
40. Raza B, Ahmad R, Nasir MHNBM, Fauzi SSM. Socio-Technical Congruence as an Emerging Concept in Software Development: A Scientometric Analysis and Critical Literature Review. IEEE Access. 2021;9:129051–77. <https://doi.org/10.1109/ACCESS.2021.3113637>
41. Kitchenham B, Charters S. Guidelines for performing Systematic Literature Reviews in Software Engineering Version 2.3. Vol. 45, Durham University. 2007 <https://acortar.link/dzpbgk>
42. Van Solingen R, Basili V, Caldiera G, Rombach HD. Goal Question Metric (GQM) Approach. Encyclopedia of Software Engineering. 2002. <https://doi.org/10.1002/0471028959.sof142>
43. Wieringa R, Maiden N, Mead N, Rolland C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requir Eng. 2006;11(1):102–7. <https://doi.org/10.1007/s00766-005-0021-6>

44. Kitchenham BA, Charters S. Guidelines for performing Systematic Literature Reviews in Software Engineering (Software Engineering Group, Department of Computer Science, Keele Technical Report EBSE 2007- 001 Keele University and Durham University Joint Report. 2007;(January).
45. Wieringa R, Maiden N, Mead N, Rolland C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir Eng*. 2006 11(1):102–7. <https://doi.org/10.1007/s00766-005-0021-6>
46. Tamburri DA, Kruchten P, Lago P, Van Vliet H. What is social debt in software engineering? In: 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013 - Proceedings. 2013; 93–6. <https://doi.org/10.1109/CHASE.2013.6614739>
47. Tamburri DA, Nitto E Di. When Software Architecting Leads to Social Debt.2005. <http://tinyurl.com/ljqgay9>
48. Besker T, Ghanbari H, Martini A, Bosch J. The influence of Technical Debt on software developer morale. *Journal of Systems and Software*. 2020;167. <https://doi.org/10.1016/j.jss.2020.110586>
49. Besker T, Martini A, Bosch J. The use of incentives to promote technical debt management. *Inf Softw Technol*. 2022;142: 106740 <https://doi.org/10.1016/j.infsof.2021.106740>
50. Martini A, Besker T, Bosch J. Technical Debt tracking: Current state of practice: A survey and multiple case study in 15 large organizations. *Sci Comput Program*. 2018;163:42–61. <https://doi.org/10.1016/j.scico.2018.03.007>
51. Almarimi N, Ouni A, Mkaouer MW. Learning to detect community smells in open source software projects. *Knowl Based Syst*. 2020 ;204. <https://doi.org/10.1016/j.knosys.2020.106201>
52. Palomba F, Tamburri DA. Predicting the emergence of community smells using socio-technical metrics: A machine-learning approach. *Journal of Systems and Software*. 2021;171:110847 <https://doi.org/10.1016/j.jss.2020.110847>
53. De Stefano M, Iannone E, Pecorelli F, Tamburri DA. Impacts of software community patterns on process and product: An empirical study. *Sci Comput Program*. 2022; 214: 102731 <https://doi.org/10.1016/j.scico.2021.102731>
54. Gemma Catolino, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, Filomena Ferrucci. Refactoring Community Smells in the Wild: The Practitioner’s Field Manual. *ICSE*. 2020;25–34. <https://doi.org/10.1145/3377815.3381380>
55. Almarimi N, Ouni A, Chouchen M, Saidani I, Mkaouer MW. On the detection of community smells using genetic programming-based ensemble classifier chain. In: *Proceedings - 2020 ACM/IEEE 15th International Conference on Global Software Engineering, ICGSE 2020*. Association for Computing Machinery, Inc. 2020; 43–54. <https://doi.org/10.1145/3372787.3390439>
56. Catolino G, Palomba F, Tamburri DA, Serebrenik A, Ferrucci F. Gender diversity and women in software teams: How do they affect community smells? 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS). IEEE, 2019. <https://doi.org/10.1109/ICSE-SEIS.2019.00010>
57. De Stefano M, Pecorelli F, Tamburri DA, Palomba F, De Lucia A. Splicing Community Patterns and Smells: A Preliminary Study. 42nd International Conference on Software Engineering Workshops. 2020: 703–10. <https://doi.org/10.1145/3387940.3392204>
58. Canedo ED, Mendes F, Cerqueira A, Okimoto M, Pinto G, Bonifacio R. Breaking one barrier at a time: How women developers cope in a men-dominated industry. In: *ACM International Conference Proceeding Series*. Association for Computing Machinery. 2021;378–87. <https://doi.org/10.1145/3474624.3474638>

59. Catolino G, Palomba F, Tamburri DA, Serebrenik A. Understanding Community Smells Variability: A Statistical Approach. In: Proceedings - International Conference on Software Engineering. IEEE Computer Society. 2021;77–86. <https://doi.org/10.1109/ICSE-SEIS52602.2021.00017>
60. Tamburri DA, Palomba F, Kazman R. Exploring Community Smells in Open-Source: An Automated Approach. IEEE Transactions on Software Engineering. 2019; 47(3): 630–652. <https://doi.org/10.1109/TSE.2019.2901490>
61. Tamburri DA. Software Architecture Social Debt: Managing the Incommunicability Factor. IEEE Trans Comput Soc Syst. 2019;6(1):20–37. <https://doi.org/10.1109/TCSS.2018.2886433>
62. Palomba F, Tamburri DA, Serebrenik A, Zaidman A, Fontana FA, Oliveto R. How do community smells influence code smells? In: Proceedings - International Conference on Software Engineering. 2018; 240–1. <https://doi.org/10.1145/3183440.3194950>
63. Huang Z, Shao Z, Fan G, Gao J, Zhou Z, Yang K, et al. Predicting Community Smells' Occurrence on Individual Developers by Sentiments. 2021;12. <https://doi.org/10.1109/ICPC52881.2021.00030>
64. Catolino G, Palomba F, Tamburri DA, Serebrenik A, Ferrucci F. Gender Diversity and Community Smells: Insights from the Trenches. IEEE Softw. 2020;37(1):10–6. <https://doi.org/10.1109/MS.2019.2944594>
65. Palomba F, Andrew Tamburri D, Arcelli Fontana F, Oliveto R, Zaidman A, Serebrenik A. Beyond Technical Aspects: How Do Community Smells Influence the Intensity of Code Smells? IEEE Transactions on Software Engineering. 2021;47(1):108–29. <https://doi.org/10.1109/TSE.2018.2883603>
66. Eken B, Palma F, Ay   B, Ay   T. An empirical study on the effect of community smells on bug prediction. Software Quality Journal. 2021;29(1):159–94. <https://doi.org/10.1007/s11219-020-09538-7>
67. Tamburri TU DA, datamburri -jads, Rick Kazman tuenl, Van den Heuvel Universiteit van Tilburg -JADS WJAMvdHeuvel WJ. Splicing Community and Software Architecture Smells in Agile Teams: An industrial Study. 2019. <https://doi.org/10.24251/HICSS.2019.843>
68. Tamburri DA, Kruchten P, Lago P, Vliet H van. Social debt in software engineering: insights from industry. Journal of Internet Services and Applications. 2015;6(1). <https://doi.org/10.1186/s13174-015-0024-6>
69. Dreesen T, Hennel P, Rosenkranz C, Kude T. "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;6826–35. <https://doi.org/10.24251/HICSS.2021.818>
70. Ahammed T, Asad M, Sakib K. Understanding the Relationship between Missing Link Community Smell and Fix-inducing Changes. 2021;469–75. <https://doi.org/10.5220/0010500604690475>
71. Sarmiento C, Massoni T, Serebrenik A, Catolino G, Tamburri D, Palomba F. Gender Diversity and Community Smells: A Double-Replication Study on Brazilian Software Teams. 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, 2022; 273–83. <https://doi.org/10.1109/SANER53432.2022.00043>
72. Lambiase S, Catolino G, Tamburri DA, Serebrenik A, Palomba F, Ferrucci F. Good fences make good neighbours? In Association for Computing Machinery (ACM); 2022; 67–78. <https://doi.org/10.1109/ICSE-SEIS55304.2022.9793992>
73. Mauerer W, Joblin M, Tamburri DA, Paradis C, Kazman R, Apel S. In Search of Socio-Technical Congruence: A Large-Scale Longitudinal Study. IEEE Transactions on Software Engineering. 2022;48(8):3159–84. <https://doi.org/10.1109/TSE.2021.3082074>

74. Nadri R, Rodriguez-Perez G, Nagappan M. On the Relationship Between the Developer's Perceptible Race and Ethnicity and the Evaluation of Contributions in OSS. *IEEE Transactions on Software Engineering*. 2022;48(8):2955–68. <https://doi.org/10.1109/TSE.2021.3073773>
75. Prana GAA, Ford D, Rastogi A, Lo D, Purandare R, Nagappan N. Including Everyone, Everywhere: Understanding Opportunities and Challenges of Geographic Gender-Inclusion in OSS. *IEEE Transactions on Software Engineering*. 2021; <https://doi.org/10.1109/TSE.2021.3092813>
76. Mumtaz H, Paradis C, Palomba F, Tamburri DA, Kazman R, Blincoe K. A preliminary study on the assignment of GitHub issues to issue commenters and the relationship with social smells. 15th International Conference on Cooperative and Human Aspects of Software Engineering. 2022; 61–5. <https://doi.org/10.1145/3528579.3529181>
77. Pigazzini I, Fontana FA, Walter B. A study on correlations between architectural smells and design patterns. *Journal of Systems and Software*. 2021;178:110984. <https://doi.org/10.1016/j.jss.2021.110984>
78. Soliman M, Avgeriou P, Li Y. Architectural design decisions that incur technical debt — An industrial case study. *Information-and-Software-Technology*. 2021;139. <https://doi.org/10.1016/j.infsof.2021.106669>
79. Aksekili AY, Stettina CJ. Women in Agile: The Impact of Organizational Support for Women's Advancement on Teamwork Quality and Performance in Agile Software Development Teams. *Lecture Notes in Business Information Processing*. 2021;408:3–23. https://doi.org/10.1007/978-3-030-67084-9_1
80. Bjarnason E, Gislason Bern B, Svedberg L. Inter-team communication in large-scale co-located software engineering: a case study. *Empir Softw Eng*. 2022;27(2). <https://doi.org/10.1007/s10664-021-10027-z>
81. Martini A, Stray V, Moe NB. Technical-, Social- and Process Debt in Large-Scale Agile: An Exploratory Case-Study. In: *Lecture Notes in Business Information Processing*. Springer Verlag. 2019; 112–9. https://doi.org/10.1007/978-3-030-30126-2_14
82. Tamburri DA, Palomba F, Serebrenik A, Zaidman A. Discovering community patterns in open-source: a systematic approach and its evaluation. Vol. 24, *Empirical Software Engineering*. Empirical Software Engineering; 2019;1369–1417 <https://doi.org/10.1007/s10664-018-9659-9>
83. Sievi-Korte O, Fagerholm F, Systä K, Mikkonen T. Dimensions of Consistency in GSD: Social Factors, Structures and Interactions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 2020;12562:315–30. https://doi.org/10.1007/978-3-030-64148-1_20
84. Stefano M De, Pecorelli F, Tamburri DA, Palomba F, Lucia A De. Refactoring Recommendations Based on the Optimization of Socio-Technical Congruence. In: 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME. 2020;794–6. <https://doi.org/10.1109/ICSME46990.2020.00094>
85. Ahammed T, Asad M, Sakib K. Understanding the Involvement of Developers in Missing Link Community Smell: An Exploratory Study on Apache Projects. *International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE*. 2020; 469–75. <https://doi.org/10.5220/0010500604690475>
86. Lavallée M, Robillard PN. Are we working well with others? How the multi team systems impact software quality. *E-Informatica Software Engineering Journal*. 2018;12(1):117–31. <https://doi.org/10.5277/e-inf180105>

87. Martini A, Bosch J. Revealing social debt with the CAFFEA framework: An antidote to architectural debt. Proceedings - 2017 IEEE International Conference on Software Architecture Workshops. 2017;179–81. <https://doi.org/10.1109/ICSAW.2017.42>
88. Serebrenik A. Emotional labor of software engineers. 2017;1-6. <https://tinyurl.com/46n66wmx>
89. Palomba F, Serebrenik A, Zaidman A. Social Debt Analytics for Improving the Management of Software Evolution Tasks. 16th Edition of the BElgian-NEtherlands Software EVOLution Symposium.2017; 18-21 <https://tinyurl.com/mr3edacy>
90. M. O. Ahmad. Psychological Safety, Leadership and Non-Technical Debt in Large-Scale Agile Software Development. In: Proceedings of the 18th Conference on Computer Science and Intelligence Systems. IEEE; 2023; 327–34. <https://doi.org/10.15439/2023F8595>
91. C. Gote. Locating community smells in software development processes using higher-order network centralities. Soc Netw Anal Min. 2023;13:1–28. <https://doi.org/10.1007/s13278-023-01120-w>
92. Wieringa R, Maiden N, Mead N, Rolland C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requir Eng. 2006;11(1):102–7. <https://doi.org/10.1007/s00766-005-0021-6>

Anexe

Table 8. Relevance and pertinence criteria

Id criterion	Criteria	Scoring for answers		
		+ 1	0	-1
C1	The main theme of the study is the investigation of community smells or the social dimension of debt in software development.	Yes	Partially	No
C2	The study presents a precise description of the research problem.	Yes	Partially	No
C3	The study follows a structured and transparent research procedure.	Yes	Partially	No
C4	The study provides a clear definition of the social dimension of debt.	Yes	Partially	No
C5	The study proposes a set of elements to be considered for evaluating community smells or the social dimension of debt in software development.	Yes	Partially	No
C6	The study presents the results in a clear and accessible manner.	Yes	Partially	No
C7	The study clearly outlines the contributions relevant to industry and/or academia.	Yes	Partially	No
C8	The study provides a comprehensive discussion of the limitations of the research process and analyzes the results obtained.	Yes	Partially	No
C9	The study presents future work and research gaps in a clear and explicit way.	Yes	Partially	No
C10	The study has been published in a relevant journal, conference, or congress. The quartile ranking proposed by Scimago (https://www.scimagojr.com/) was used for journal classification, and the Computing Research & Education (CORE, http://portal.core.edu.au/conf-ranks/) ranking for conferences and congresses.	Very relevant (quartile Q1 for journals and A+ for congresses and conferences)	Relevant (Q2 and Q3 quartiles for journals and A or B for congresses and conferences)	Not relevant (Q4 quartile for journals and C or unclassified for congresses and conferences)
C11	The study has been cited by other authors (according to the Google Scholar citation index).	It has been cited by more than ten authors	Between one and ten authors	Has not been cited to date

Source: Author’s work

Table 9. Score obtained by article relevance and pertinence criteria

Primary Studies	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	Total
PS1	+1	+1	0	0	-1	0	0	-1	0	0	+1	1
PS2	+1	+1	0	0	0	+1	+1	0	0	+1	-1	4
PS3	0	0	+1	0	-1	+1	+1	-1	+1	0	+1	3
PS4	0	0	+1	-1	0	0	0	+1	0	0	+1	2
PS5	+1	0	+1	0	0	+1	+1	0	+1	+1	+1	7
PS6	0	0	0	-1	-1	0	0	0	0	+1	-1	-2
PS7	0	1	0	0	0	1	0	0	0	-1	+1	2
PS8	1	1	1	0	0	1	0	-1	0	-1	+1	3
PS9	-1	0	0	-1	-1	+1	0	0	0	-1	+1	-2
PS10	+1	+1	0	+1	0	0	0	+1	+1	-1	+1	5
PS11	+1	+1	0	-1	-1	+1	+1	0	0	-1	-1	0
PS12	0	+1	+1	+1	0	+1	+1	+1	+1	0	0	7
PS13	+1	+1	0	+1	0	+1	+1	0	0	1	+1	7
PS14	+1	0	+1	+1	0	0	+1	0	+1	1	+1	7
PS15	+1	+1	0	+1	0	+1	0	+1	0	-1	+1	5
PS16	0	+1	0	0	0	+1	0	+1	0	-1	0	2
PS17	+1	0	0	-1	+1	+1	+1	+1	+1	1	+1	7
PS18	+1	+1	+1	+1	+1	+1	0	+1	+1	1	+1	10
PS19	0	+1	+1	+1	0	+1	+1	0	+1	+1	-1	6
PS20	+1	+1	+1	0	+1	+1	0	0	0	+1	0	6
PS21	0	+1	+1	0	+1	+1	+1	+1	+1	-1	-1	5
PS22	+1	+1	+1	+1	+1	+1	0	0	+1	-1	0	6
PS23	+1	+1	0	0	+1	+1	+1	+1	0	-1	0	5
PS24	+1	+1	+1	+1	+1	+1	+1	+1	0	0	0	8
PS25	+1	+1	+1	0	+1	+1	+1	+1	0	-1	-1	5
PS26	+1	+1	+1	0	+1	+1	+1	0	+1	+1	0	8
PS27	+1	+1	+1	+1	+1	+1	+1	+1	0	0	0	8
PS28	+1	+1	+1	0	-1	+1	+1	+1	0	+1	0	6
PS29	0	+1	+1	0	+1	+1	+1	+1	0	+1	0	7
PS30	+1	0	+1	+1	0	+1	+1	+1	+1	+1	-1	7
PS31	+1	+1	+1	-1	+1	+1	+1	+1	0	+1	0	7
PS32	0	+1	0	-1	-1	+1	+1	+1	-1	-1	-1	-1
PS33	-1	1	0	-1	-1	1	0	1	1	-1	0	0
PS34	-1	1	0	-1	-1	1	0	1	1	-1	0	0
PS35	0	1	0	-1	0	1	1	0	1	-1	1	3
PS36	-1	1	1	-1	0	1	1	1	1	-1	0	3
PS37	0	1	1	-1	0	1	1	1	1	-1	1	5
PS38	-1	1	1	-1	-1	1	1	1	1	-1	1	3
PS39	-1	1	1	-1	-1	1	1	1	1	-1	0	2
PS40	0	1	1	-1	1	1	1	1	1	-1	0	5
PS41	-1	1	1	-1	1	1	-1	1	1	-1	0	2
PS42	1	1	1	1	1	1	1	1	1	-1	0	8
PS43	1	1	1	1	1	1	1	1	1	-1	0	8
PS44	1	1	1	1	1	1	1	1	1	-1	1	9

Source: Author's work

Table 10. Community Smells Identified in Software

CS Id	Name of Community Smell	Definition of Community Smell	Ref.
1	Architecture by osmosis. (AO)	Occurs when architectural decisions are made based on incomplete or poorly managed information. Some community members use poorly defined channels or differing standards to document changes, resulting in wasted time, difficulty tracking key decisions, and an unstable architecture Development..	A26
2	Architectural hood (AH)	Occurs when architecture teams work remotely or have little interaction with developers. This lack of direct contact hinders collaboration and prevents the proper resolution of implementation issues arising from architectural decisions.	A26
3	Organizational silo (OS)	Refers to the existence of subgroups within the developer community that communicate only through one or two representatives, limiting information flow and the effective integration of the team.	A4, A5, A6, A7, A8, A9, A10, A12, A13, A16, A17, A18, A19, A23, A24, A25, A26, A29, A32, A35, A5, A6, A7, A9, A9,
4	Black cloud (BC)	Arises when the organization does not provide conditions that foster communication and interaction among community members, impeding knowledge and experience exchange during software development.	A10, A12, A13, A17, A18, A24, A25, A26

5	Lone Wolf (LW)	Manifests when certain members act independently, disregarding their peers or failing to coordinate efforts. Such behavior may result in unauthorized technical decisions, coding errors, and project delays.	A7, A9, A10, A12, A12, A16, A17, A18, A21, A26
6	(BN) Bottleneck	Occurs when the same individual mediates all interactions between two or more sub-communities, hampering communication flow. This causes production delays and reduced team efficiency.	A6, A7, A10, A13, A13, A16, A18 A1, A4, A7, A8, A9, A12, A13, A17, A17, A24, A25, A26, A32
7	Radial Silence (RS)	Arises when community procedures are overly formal and structured, leading to implementation delays and wasted time due to complex, inflexible processes.	A8, A16, A26
8	Prima-donnas (PD)	Emerges when some members refuse to accept changes proposed by others, as a result of an inefficient collaboration structure. This resistance negatively impacts workflow and team cohesion.	A1, A8, A26
9	Share villainy (SV)	Describes environments where sharing information or experiences is not prioritized, often due to a lack of space, incentives, or organizational support for collaborative learning.	A8, A14, A26
10	Organizational Scrimmage (OSCR)	Occurs when differences between organizational cultures hinder project coordination and management. These differences can impact productivity and cause significant delays.	A8, A9, A18, A25
11	Solution Defiance (SD)	Refers to tensions arising when groups within the same community, with similar values or technical backgrounds, defend opposing technical or strategic decisions, generating conflict in key meetings.	A16, A41
12	Truck factor (TF)	Evident when a significant portion of project-relevant information is concentrated in one or two developers, representing a risk for project continuity and limiting team autonomy.	
13	Smelly Developer (SDV)	Refers to developers affected by community smells such as Lone Wolf, Organizational Silo, and Bottleneck, which are characterized by poor communication, limited collaboration, and poorly integrated structures, reducing developer participation and team effectiveness.	

14	Smelly Abandoner (SAB)	Refers to developers with community smell presence who participated in earlier project phases and then left the community, potentially causing knowledge gaps and affecting development continuity.	A16
15	Missing Link (ML)	Occurs when collaborators modify source code in isolation, without communicating with colleagues, leading to inconsistencies, rework, and hindering change integration.	A1, A26, A32, A41
16	Classroom cognition (CC)	Refers to situations where refactoring complicates code understanding for both current and new developers, requiring extra effort to comprehend new structures and locations.	A18, A26
17	Code Red (CR)	This smell arises when classes are so complex that only one or two developers can understand or maintain them, generating critical dependencies and technical risk.	A26 A14, A20, A26, A36, A38, A38, A14, A20, A26, A36, A38
18	Cognitive distance (CD)	Occurs when team members perceive physical, technical, social, or cultural barriers with colleagues, hindering collaboration and potentially causing conflicts or coordination problems.	A20, A26
19	Cookbook development (CKD)	Occurs when developers cling to traditional approaches and resist adopting new methodologies, such as agile, thereby limiting innovation and producing products that fail to meet customer expectations.	A20, A26
20	DevOps Shock (DOS)	Describes clashes between distributed development and operations teams, especially when subject to differing contractual obligations, slowing down development and reducing operational efficiency.	A20, A26
21	Unhooking (UNH)	Happens when developers consider software ready and deliver it to operations even though it is incomplete, leading to unvalidated features and errors.	A20, A26
22	Dispersion (DISP)	Refers to solutions or refactoring that cause fragmentation of an established group or collaborative network. Rearranging functionality can result in disorganized work and increased maintenance difficulty.	A26
23	Dissent (DIS)	Occurs when teams fail to reach consensus on issue resolution, leaving identified code smells unaddressed.	A18, A26
24	Hypercommunity (HC)	In highly cohesive communities, the excessive influence of collective opinions may override critical thinking, spill over into sub-communities, and ultimately impact software quality.	A20, A26
25	Excess of informality (EI)	The absence of clear information management rules leads to data overload, disorganization, and lack of accountability among community members.	A20, A26

26	Institutional isomorphism (II)	Refers to similarities between processes or structures of different sub-communities, whether by imitation or operating under similar constraints, potentially leading to stagnation, lack of innovation, and communication issues.	A26, A38
27	Invisible architecture (IA)	Occurs when architectural decisions and meeting agreements are not documented clearly or consistently, leading to knowledge gaps, implementation errors, and intra-team conflicts.	A26
28	Solitary architecture (SA)	Arises when developers without architectural training make critical decisions due to limited architect availability, resulting in technical incompatibilities and misalignment during development.	A20, A26
29	Leveraging rookies (LR)	Newcomers do not receive proper orientation and must independently determine their roles and contacts, increasing pressure, disinterest among senior staff, and negatively affecting team psychological well-being.	A20, A26
30	Obfuscated architecture (OA)	Occurs when multiple sub-communities lack a shared organizational and technical vision, hindering the integration of new developers, especially in projects combining legacy and modern systems.	A26
31	Power distance (PDST)	Refers to how lower-ranking members perceive and accept leaders' behavior, decisions, and assignments. This perception can influence team cohesion, motivation, and trust.	A6, A20, A25, A26
32	Meek members (MM)	Some members behave with excessive caution and rigid adherence to ethical or technical standards, impeding risk-taking, limiting adaptability, and affecting project progress.	A20, A26
33	Unlearning (UL)	Variations in member experience levels hinder the assimilation of new training content. Updated knowledge risks being lost if not properly transmitted to senior members.	A20, A26
34	Time Tunnel (TT)	After structural changes, members may mistakenly assume communication will improve, overlooking the need for new explicit coordination mechanisms.	A20, A26
35	Leftovers from a Technology Enthusiast (LTE)	When there is no collaborative network between developers and technical staff (e.g., help desk, operations, maintenance), knowledge sharing is impeded, fragmenting the development process.	A26

Source: Author's work