Do Research and Practice of Code Smell Identification Walk Together? A Social Representations Analysis

Rafael de Mello*, Anderson Uchôa*, Roberto Oliveira*‡, Willian Oizumi*,
Jairo Souza†, Kleyson Mendes‡, Daniel Oliveira*, Baldoino Fonseca†, Alessandro Garcia*
*Informatics Department, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil
Email: {rmaiani, auchoa, woizumi, doliveira, afgarcia}@inf.puc-rio.br
†Computing Institute, Federal University of Alagoas (UFAL), Brazil
Email: {jrmcs, baldoino}@ic.ufal.br
‡ State University of Goiás (UEG), Goiás, Brazil
Email: roberto.oliveira@ueg.br, kleyson.lucas@gmail.com

***Abstract*—Context: It is frequently claimed the need for bridging the gap between software engineering research and practice. In this sense, the theory of social representations may be useful to characterize the actual concerns of software developers. It comprises the system of values, behaviors, and practices of communities regarding a particular *social object*, such as the task of smell identification. Aim: To characterize the social representations of smell identification by software developers. Method: Based on the answers given to a questionnaire, we analyzed the associations made by the developers about smell identification, i.e., what immediately comes to their minds when they think about this task. Results: We found that developers strongly associate smell identification with the practice of *smell removal* and with the incidence of *bugs*. They also frequently associate the task with the practice of *inspection* and with the need of having *individual skills*. Besides, we verified that the current state of the art on smell identification partially address the social representations of the software developers. Conclusion: There is a considerable gap between the research of smell identification and its practice. We propose directions to mitigating this gap.**

***Index Terms*—social representations; associations; code smells**

I. Introduction

There is a recurrent claim in the Software Engineering field regarding the need for bridging the gap between research and practice (e.g., [1], [2]). Despite the availability of several technologies, there are important software engineering tasks that remain challenging for software developers. One of these tasks is the identification of code smells. Code smells are symptoms of poor design observed in the low-level structure of a system [3], [4]. The code smell incidence hampers the readability of the source code and its maintenance.

Technical literature provides several catalogs (e.g., [5], [6]) and detection tools [7], [8] to support the identification of code smells. While catalogs try to organize knowledge regarding smelly code, detection tools aim at automating the detection of code smells. However, these technologies have been shown insufficient to capture the complexity surrounding the process of identifying code smells. The identification of code smells is a non-deterministic and subjective task [9], [10], which means that different interpretations may emerge from the same input. One possible explanation regards the fact that smell identification is influenced by several context factors, including human [11], [12], and technical ones [7].

In this sense, the findings of a recent preliminary study suggest that the community of software developers has different concerns regarding smell identification than those typically addressed by postgraduate students [13]. This scenario suggests the need for investigating in depth what developers have in mind regarding the task and to which extent their concerns are addressed by software engineering research. For this purpose, this paper presents an empirical study of the social representations of smell identification by software developers. Social representations mean the collective elaboration of a social object by a particular community for behaving and communicating [14]. A *social object* may be any task performed by the members of this community. Therefore, investigating social representations can be useful to identify opportunities for improving the practice surrounding a particular task.

In our study, we performed a free association task with the subjects [15]. This task works as a trigger to developers to evoke what comes to mind when they think about the code smell identification. The evoked terms resulted in the set of associations of the community. We collected 125 associations from 32 developers actively working in industry. They work in different software companies from Brazil. After performing the analysis of the social representations, we found evidence that the investigated community believes that smell identification is essentially an inspection task addressing the analysis of source code metrics, structure, and semantics. Developers also strongly associate the identification of code smells with their removal and with the incidence of bugs.

The findings of our study indicate gaps in the state of the art that should be addressed for supporting software development.

978-1-7281-2968-6/19/$31.00

For instance, we did not find technologies for supporting both the manual verification of code smells and the decision process involved in the removal of code smells. Considering these and other gaps, we discuss practical implications and directions for smell identification. These implications include, for instance, the need of developing a recommendation system for supporting the whole process of smell identification and removal. The experience reported in this paper indicates that the theory of social representations is a useful resource for investigating the actual concerns of the software industry on performing software engineering tasks.

II. Background and Related Work

At the beginning of the 20th century, Emelie Durkheim depicted the collective representations theory. For her, collective representations are constructed by a wide range of knowledge acquired and reproduced in society, unconsciously. Some decades after, the social psychologist Serge Moscovici develops the social representations theory influenced by Durkheim's theory [14]. Moscovici claims that collective representations do not contemplate contemporary individuality, once the current social phenomena are much more related to the daily life of the individual. Thus, the social representations theory considers the individual as part of the construction of representations. It aims to explain the humans' phenomena from a collective perspective without losing sight of the individuality.

Social representations mean the collective elaboration of a social object by a particular community for behaving and communicating [14]. A social object corresponds to an object socialized by two or more individuals from a group. It can be, e.g., a software engineering task such as the identification code smells. Social representation emerges from common sense. They comprise a system of values, ideas, and practices. One of the goals of the social representations is to establish an order that will enable the members of a community to guide themselves in their material and social world. Another goal is enabling the communication among the members of a community, establishing a code for social exchanging and a code for naming and classifying the different aspects of their world [14].

The data used for analyzing social representations are typically grounded gathering the free associations of the subjects. Free association is a technique used in psychoanalysis and frequently applied through individual semi-structured interviews [15]. In free association tasks, individuals are asked to quote what first comes to mind when they think about the social object investigated. The question should be immediately answered, and the quotations provided should be noted the same way they were uttered, i.e., the order they came to mind. Several associations with different frequencies may emerge from the free association task. In this sense, it is possible to identify and group terms with similar meanings in the context of the study. Therefore, we point out that working on free association tasks is quite different from opinion gathering, obtained from typical survey questions [16]. The characteristics of the free association task stimulate subjects to do not censor their thoughts. On the other hand, opinions are given by reflection regarding an issue, which may be strongly influenced by formal knowledge, external pressures, and moral codes.

To best of our knowledge, the social representations theory was used only once – and recently – in the field [13]. In this work, we investigated the social representations of the code smell identification by practitioners and postgraduate students. This so-called preliminary study, published in a workshop, involved software developers working in a particular city and students from a post-graduation course. We found differences between the social representations of the communities investigated. However, the restricted representativeness and the low frequency of the associations limited the discussions to first impressions and lessons learned. Different from [13], the study presented in this paper focus on characterizing the social representations of the identification of code smells by the software developers actively working in software industry located in a particular country (Brazil). We also performed an analysis regarding in which extent code smell research has been addressing the social representations identified.

III. Study Settings

The study aim at characterizing the social representations of software developers regarding the identification of code smells. In this sense, we aim to answer the following research question: *What are the social representations of the identification of code smells by software developers?* Social representations emerge from the community as a whole. Therefore, our definition of software developers covers individuals with development knowledge and working in the software industry.

Communities may share cultural aspects, such as language and values. Studies in social representations frequently investigate communities composed of groups from particular countries [17], [18]. We opted by listening software professionals actively working for companies located in different regions of Brazil. Due to the control needed to apply the study questionnaire, the recruitment was performed *in loco*.

A. Instrumentation

We applied the same questionnaire developed for the preliminary study [13]. The questionnaire includes a free association task for supporting the analysis of the social representations. The other items of the questionnaire address the characterization of the community (see table I). The questions were intentionally grouped in blocks to avoid biasing the participants in the free association task. Each block should be presented only after answering the previous one.

After answering the familiarity with the term *code smells* (B2), the respondent should perform the free association task (B3). In this task, we asked to list at least five words that *immediately* come to mind when thinking about the identification of code smells, following the sequence in which the words are evoked. Only after performing this task, we asked the opinion (B4) and knowledge (B5) regarding the object of study.

We designed the questionnaire to be applied in a controlled environment without external access. At least one researcher

should supervise this environment, assuring the subjects attend to the requirements to perform the free association task. Before the subjects started filling out the questionnaire, they should be instructed to follow the sequence of questions and not change their answers.

TABLE I
QUESTIONNAIRE ITEMS

Block	Question
$\mathbf{B}_1$	Q1. What is your highest academic degree in computer science or related fields? ( ) Graduate degree ( ) Master degree ( ) Doctorate degree
	Q2. Are you currently working in the industry? If so, what is your current role?
	Q3. In the following lines, briefly summarize your experience with software development.
$\mathbf{B}_2$	Q4. Are you familiar with the term code smells? ( ) Yes ( ) No
	Q5. For you, what is a code smell?
$\mathbf{B}_3$	Q6. What immediately comes to mind when you think about the identification of code smells? Please provide up to five words in the order they came to your mind.
	Q7. Considering the words you have mentioned, which one do you consider the most relevant to express your opinion on the task of identifying code smells?
$\mathbf{B}_4$	Q8. On the following statement: "Identifying and removing code smells are essential tasks for understanding the source code," you: ( ) Totally disagree ( ) Partially disagree ( ) Partially agree ( ) Totally agree
	Q9. On the following statement: "Identifying and removing code smells are essential tasks for promoting the structural quality of the source code," you: ( ) Totally disagree ( ) Partially disagree ( ) Partially agree ( ) Totally agree
$\mathbf{B}_5$	Q10. Please provide all types of code smell that you remember how to identify. If you do not remember the name of the type of code smell, you may describe the general situation associated with it.

B. Data Analysis

We followed the same procedures used in [13] to analyze the social representations. The associations evoked by each community will be analyzed by calculating their frequencies and their average orders of evocation (AOE). The frequency of each association is calculated by counting how many times the association was quoted. AOE is the sum of the positions in which each subject had quoted the term divided by the frequency of the association. For instance, suppose the association "easy" was quoted four times in the first place and quoted twice in the fourth place. Therefore, the frequency of the association is 6, and its AOE is 3, i.e., $(((4*1)+(2*4))/4)$.

The lower the AOE of an association is, the more promptly the community is to associate the social object investigated to the corresponding term and vice-versa. We will measure the strength of the associations in a community through the four-quadrant analysis [15]. The following thresholds are calculated for identifying in which quadrant each association should be included: the mean frequency of the associations, and the ratio between the sum of the AOE and the sum of the frequencies of the associations. The strongest quadrant is the central system, composed by the core elements, i.e., the associations which were more frequently and more promptly evoked.

C. Execution

The study questionnaire was applied predominantly in the developers' workplace. In a few cases, it was needed to run the questionnaire at the researcher' university. We also experienced conducting individual and structured interviews for gathering the answers. Apart from the differences in the environments, we made efforts to preserve the control required to apply the free association task. The free association method requires providing prompt and objective answers to the stimulating questions [15]. In other words, the free association task should not be influenced by any external factor or even by any "time to think". The researchers involved in the execution verified that all participants followed the given instructions for answering the questionnaire.

IV. THE COMMUNITY

Our study count with the participation of 32 software developers. Most of these developers (22) currently work in four private companies. Two companies are small, having approximately ten employees each. They develop web and mobile applications for different domains. The other two companies are large. The first one have been delivering software solutions from scratch and customizing enterprise solutions in the last 18 years. The second one is an automotive company having more than 1,000 employees. This company delivers specialized software solutions for its needs. The other ten developers work in public companies. One company has approximately 30 employees and delivering software for innovation. This company is part of a university. The other company is another university in which their developers work maintaining academic systems.

After analyzing the answers given to the characterization questions (see Table I), we found considerable diversity in the sample. It includes not only individuals currently playing the role of software developers. It also include project managers, technical leaders, IT analysts, among others. Most of the developers hold bachelor degrees (72%), while others are undergraduate students (16%) or hold master degrees (12%). They have different levels of experience in software development, ranging from less than one year of experience to 17 years (6 years in the average). Regarding code reviews, we found that approximately half of them (53%) has professional experience in performing the task. The rest of this section covers the characterization of the community investigated in terms of knowledge and opinions about the identification of code smells. The results of the free association task are used in Section V to perform the Social Representation analysis.

What is a code smell? 63% of the developers declared familiarity with the term code smell. In spite of that, half the participants who were not familiar with code smells provided acceptable definitions. The majority of developers provided definitions of the term somehow compatible with the definition available in the literature [3], [4]. Most of developers provided definitions quoting negative consequences of code smells and concerns with the understanding of the source code. Besides, some developers also mentioned root causes of code smells.

Opinion on identifying and removing code smells. We asked the developers' opinion regarding the relevance of identifying and removing code smells for promoting the structural quality of the source code. We observe that the developers tend to agree (90%, 60% totally agree) that identifying and removing code smells is a relevant resource to this purpose. We also asked the perception of developers on the relevance of identifying and removing code smells for understanding the source code. We found that the majority of the software developers agree (71%, 26% totally agree) that identifying and removing code smells is relevant to this purpose.

Knowledge of code smell types. We also asked developers which types of code smell they remember how to identify. The respondents were allowed to describe the actual smelly structure even if they did not remember the formal name associated with the type representing that smelly structure. We carefully associated each descriptive answer to the definitions available in the catalogs [5], [6]. Then, we classified the code smells by granularity – inter-class or intra-class [19], and by the predominant scope in which they are applied – measurable, structural or semantic [13]. Table II lists the most frequent types of code smell mentioned by the developers (above the median). We found that developers mentioned more frequently knowledge about the identification of measurable smells than the structural ones. In addition, a single semantic smell- bad variable name- was mentioned twice.

The results reported indicate that the investigated community is aware of code smells and their consequences for the source code. They also share positive opinions regarding the benefits of smell identification and removal. Besides, the knowledge of software developers about code smell types is concentrated in measurable and structural ones. Such characterization is important to understand the context in which the social representations presented in Section V emerge.

TABLE II
MORE FREQUENTLY CODE SMELL TYPES MENTIONED BY DEVELOPERS

Code smell types	Granularity	Scope	Frequency
God Class	Inter-class	Measurable	9
Switch Statement	Intra-class	Structural	7
Long Method	Intra-class	Measurable	7
Feature Envy	Inter-class	Measurable	6
Duplicated Code	Inter-class	Structural	5

V. SOCIAL REPRESENTATIONS OF THE IDENTIFICATION OF CODE SMELLS

This section describes and discusses the analysis of the social representations of smell identification based on the free association task (see Section III-A) performed by the developers. In this task, each subject should answer the following question: *What comes to mind when you think about the identification of code smells?*. Three researchers performed separated analyses of the terms evoked by the members of each community. The goal of these analyses was identifying and clustering similar terms into a single association.

Table III exemplifies how evoked terms were performed in a single association. In this analysis, the researchers eventually used the respondents' answers to other items of the questionnaire, such the explanation regarding the more relevant association (see Section III-A). This verification was important to reach the actual meaning of the terms quoted. The researchers also grouped the different types of code smells quoted based in types of properties on which they apply (see Section IV). After concluding the grouping activity, one researcher composed the final set of associations based on the agreement of groupings made by the other researchers. In the case of no agreement for grouping a term, this researcher should make the final decision. Then, the final sets of associations were submitted to the four-quadrant analysis (Section III-B).

TABLE III
EXAMPLE OF TERMS CLUSTERED INTO A SINGLE ASSOCIATION

Evoked Term (frequency)	Association
refactoring (8) rewrite the code (1)	removing smells
design problems (1) fail on following a pattern (1) jerry-rig (1) problems (1)	design problem

Most of the 32 developers quoted five terms, resulting in 125 terms evoked, including repetitions. The set of terms evoked by the developers can be found at https://anderson-uchoa.github.io/ESEM2019/. After analyses, we consolidated a final set composed of 27 distinct associations. From these, nine associations were quoted only once. Therefore, they were excluded from the analysis. Table IV presents the analysis of the four quadrants. Associations with the frequency equal or higher than 5 and with AOE lower than 2.49 compose the central system of the social representations.

TABLE IV
FOUR-QUADRANT ANALYSIS OF THE SOCIAL REPRESENTATIONS OF SMELL IDENTIFICATION BY DEVELOPERS

Central system			**First periphery**		
Frequency $\geq$ 5	**AOE $<$ 2.49**		**Frequency $\geq$ 5**	**AOE $\geq$ 2.49**	
	Frequency	**AOE**		**Frequency**	**AOE**
Removing smells	9	2.00	Inspection	12	2.58
Bugs	8	2.25	Structural Smells	9	2.67
Measurable smells	7	1.71	Individual Skills	9	2.78
			Code complexity	7	3.71
			Semantics	5	2.60
Contrast zone			**Second periphery**		
Frequency $<$ 5	**AOE $<$ 2.49**		**Frequency $<$ 5**	**AOE $\geq$ 2.49**	
	Frequency	**AOE**		**Frequency**	**AOE**
Design Problem	4	1.50	Testing	3	2.67
Detection	4	1.75	Performance problems	3	3.00
Source code	3	2.33	Problems	3	4.00
Fowler	2	0.50	Design	2	3.00
Maintenance	2	2.50	Tools	2	3.00

The central system (top-left quadrant) reveals that the community of software developers frequently and promptly associate the identification of code smells with their *removal*. This community also share a strong belief that identifying code smells helps to avoid the incidence of *bugs*. Surprisingly, the identification of *design problems* and the automated *detection* of code smells are not frequent associations, although relevant (third quadrant). Both associations represent a considerable part of the research efforts on this topic. Regarding the types of code smells, we found that developers promptly associate the smell identification with *measurable smells*. Besides, they often associate the smell identification to *structural smells* and *semantic smells* (second quadrant). These results indicate that the community of software developers share a belief that smell identification is a matter of making the source code readable from different perspectives, which include its semantics. For them, the identification and removal of code smells are integrated tasks for avoiding future problems.

The other associations in the second quadrant somehow address the human capacity of analyzing the code: *inspections*, *individual skills*, and *code complexity*. Regarding the individ-

ual skills, the developers evoked non-technical ones such as *patience* and *attention*. These associations indicate that the community believe that it is worthwhile to perform an accurate analysis for identifying and removing code smells.

VI. Discussion

In this section, we discuss the main findings of the analysis of the social representations reported in Section V. We searched in the technical literature how the main associations of software developers have been supported by the research of code smells. Based on the gaps found, we propose a set of practical implications.

Code smell type. Readability aspects matter for software developers. They include the size, structure, and semantics of the code elements. Nevertheless, the available catalogs cover only measurable and structural smells [5], [6]. Despite that, the semantics of code elements plays a key role in software comprehension. For instance, a class may present high cohesion and low coupling. However, if the name of its methods does not correspond to their behaviors, future maintenance will be hampered. Similarly, a method may present an acceptable structure and metrics. Still, if this method performs tasks out of the scope of the class, it should be fixed.

The lack of semantic smells observed in the catalogs is reflected in the detection tools. There are several ones to support the semi-automated detection of measurable and structural smells [7], although none of them covers all the cataloged smells. In most cases, the detection tools diverge in their detection rules. Moreover, some of these tools allow developers to customize rules. Nevertheless, the effectiveness of the detection tools is limited in terms of precision and recall for multiple smell types. In addition, these tools often diverge on the incidence of code smells in the same code snippet [7]. In this way, the adoption of detection tools grounded in machine learning has emerged as an alternative for increasing effectiveness [20]–[22]. However, building adequate training sets may require considerable manual effort. In addition, the accuracy of machine learning detection tools varies according to the smell type [21].

Removal of code smells. Developers strongly associate the smell identification with the removal of code smells. More particularly, the practice of refactoring was frequently quoted. However, identifying code smells does not imply in its removal. The findings of our study indicate that developers believe that the decision on removing code smells should be properly evaluated, avoiding side effects for the maintenance of the source code. Indeed, the removal of a single code smell may affect various code elements. However, the currently available tools still have a number of limitations for both identification and removal. Moreover, there is a lack of resources for supporting the developers' decision on removing smells. Finally, these tools do not provide explanations regarding the detected smells.

Code inspection. The developers highly associated the smell identification with the practice of code inspection [23], [24], i.e., the manual analysis of the source code. Code inspection addresses the identification of semantic defects, such as incorrect instructions and the omission of code snippets. However, manual analysis can also be extended to the code smell identification. Indeed, informal reviews are commonly applied to verify the actual incidence of code smells reported by detection tools. Therefore, the code inspection is also important for verifying the incidence of structural and measurable smells, not only for semantics. Software inspection are typically supported by techniques, such as checklists [24], [25]. However, we could not find techniques for supporting the manual identification of code smells.

Code complexity and individual skills. Considering that smell identification requires manual analysis, developers also associate the identification of code smells with individual skills (e.g., professional background). Indeed, even the identification of apparently simple smells such as Duplicated Code may require expert judgment [26]. In this sense, technical literature provides evidence that some human aspects affect the effectiveness of smell identification tasks [4], [12]. However, it is important to note that the cost involved in reviewing complex source code may considerably increase. In this sense, technical literature presents the alternative of performing smell identification in pairs. Recent studies showed that this practice may increase the effectiveness of smell identification [27].

Bugs. Previous work has observed the incidence of bugs as a consequence of smell co-occurrence, i.e., the co-occurrence of different types of smells on the same code element [28]. Therefore, the intensity of code smells in a code element is considered an important indicator for predicting the bugginess of smelly classes [29]. In this sense, developers should be aware that the introduction of a code smell may trigger the introduction of other smells. Moreover, developers should avoid introducing new code smells during removal activities.

A. Practical Implications

As observed the state of art in code smells partially addresses the social representations of software developers. Therefore, practical implications emerge from these findings. The following suggestions focus on improving the state of art on code smells, delivering technical resources for improving industrial practice. These suggestions do not intend to be exhaustive, neither aims at replacing the current technologies.

First, we suggest improving the catalogs of code smells used by software developers through the characterization of semantic smells. Second, we suggest the development and use of checklists for supporting the manual identification of code smells. Such checklists should take into account a large number of available smell types, including the semantic ones. Therefore, efforts should be made on composing verification items (questions) covering the different smelly scenarios. Alternatively, these verification items can be used to improve existing code inspection techniques. In this sense, it is important to identify approaches for organizations integrating smell identification in more comprehensive code review activities.

Finally, we believe that software developers would benefit from a recommendation system integrating the semi-automated detection of code smells and its removal. The recommendation system should report the root cause of each code smell, support the manual inspection, and offer alternatives to its removal. The recommendation system should also guide developers in their decision of removing code smells.

VII. Threats to Validity

Once cultural aspects may influence the building of social representations, the investigations should address to particular communities. However, it should be made efforts to reach representativeness. Although the limited sample size, it is important to note the study participants work in different software companies of the country. These developers have different levels of background and knowledge, playing different roles in software projects from considerably different domains.

The researchers' bias in interpreting the results is inherent to qualitative studies. In the analysis of the social representations, the grouping of terms for reaching the associations is particularly prone to bias. We mitigated this bias by grounding the associations on independent analyses from three researchers.

Free association tasks requires a level of control that hampers reaching large samples such as in opinion surveys. We trained several researchers to run the study *in loco* in different companies spread in the country. Even though the sample size is relatively small, it is important to note its diversity.

VIII. Conclusion and Future Work

The software engineering community should identify ways to bridging the gap between research and its practice. In this sense, we believe that it is worthwhile to make efforts on characterizing the actual needs and concerns of software developers regarding particular tasks. In this paper, we presented an investigation using the social representations theory to characterize what software developers have in mind regarding the identification of code smells. The social representation analysis allowed us to go beyond individual opinions, identifying the common sense of the community as a whole. The study findings suggest that academia should think the support for smell identification from a holistic perspective. As the next step, we intend to replicate the presented study with communities of software developers from other countries.

Acknowledgment

This work is funded by CAPES/Procad grant (175956) and CNPq grant (153363/2018-5, 141285/2019-2, 427787/2018-1).

References

[1] D. Lo, N. Nagappan, and T. Zimmermann, "How practitioners perceive the relevance of software engineering research," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 415–425.

[2] V. Ivanov, A. Rogers, G. Succi, J. Yi, and V. Zorin, "What do software engineers care about? gaps between research and practice," in *11th ESEC/FSE*. ACM, 2017, pp. 890–895.

[3] G. Bavota, A. De Lucia, M. Di Penta, R. Oliveto, and F. Palomba, "An experimental investigation on the innate relationship between quality and refactoring," *J. Syst. Softw. (JSS)*, vol. 107, pp. 1–14, 2015.

[4] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, and A. De Lucia, "Do they really smell bad?" in *30th ICSME*, 2014, pp. 101–110.

[5] M. Fowler, *Refactoring*. Addison-Wesley Professional, 1999.

[6] M. Lanza and R. Marinescu, *Object-oriented metrics in practice*. Springer Science & Business Media, 2006.

[7] E. Fernandes, J. Oliveira, G. Vale, T. Paiva, and E. Figueiredo, "A review-based comparative study of bad smell detection tools," in *20th EASE*, 2016, pp. 1–18.

[8] M. I. Azeem, F. Palomba, L. Shi, and Q. Wang, "Machine learning techniques for code smell detection: A systematic literature review and meta-analysis," *Inform. Softw. Tech. (IST)*, 2019.

[9] R. de Mello, R. Oliveira, L. Sousa, and A. Garcia, "Towards effective teams for the identification of code smells," in *10th CHASE*, 2017, pp. 62–65.

[10] M. Hozano, A. Garcia, N. Antunes, B. Fonseca, and E. Costa, "Smells are sensitive to developers!: on the efficiency of (un) guided customized detection," in *25th ICPC*, 2017, pp. 110–120.

[11] A. Yamashita and L. Moonen, "Do developers care about code smells?" in *20th WCRE*, 2013, pp. 242–251.

[12] R. M. de Mello, R. F. Oliveira, and A. F. Garcia, "On the influence of human factors for identifying code smells: A multi-trial empirical study," in *11th ESEM*, 2017, pp. 68–77.

[13] R. de Mello, A. Uchôa, R. Oliveira, D. Oliveira, B. Fonseca, A. Garcia, and F. de Mello, "Investigating the social representations of code smell identification: a preliminary study," in *12th CHASE*, 2019, pp. 53–60.

[14] S. Moscovici, "Notes towards a description of social representations," *Eur. J. Soc. Psychol. (EJSP)*, vol. 18, no. 3, pp. 211–250, 1988.

[15] L. Dany, I. Urdapilleta, and G. L. Monaco, "Free associations and social representations: some reflections on rank-frequency and importance-frequency methods," *Quality & Quantity*, vol. 49, no. 2, pp. 489–507, 2015.

[16] M. Torchiano, D. M. Fernández, G. H. Travassos, and R. M. de Mello, "Lessons learnt in conducting survey research," in *5th CESI*, 2017, pp. 33–39.

[17] H. Joffe and N. Bettega, "Social representation of aids among zambian adolescents," *J. Health Psychol. (JHP)*, vol. 8, no. 5, pp. 616–631, 2003.

[18] H. Räty, K. Komulainen, and L. Hirva, "Social representations of educability in finland: 20 years of continuity and change," *Social Psychology of Education*, vol. 15, no. 3, pp. 395–409, 2012.

[19] R. Oliveira, B. Estacio, A. Garcia, S. Marczak, R. Prikladnicki, M. Kalinowski, and C. Lucena, "Identifying code smells with collaborative practices: A controlled experiment," in *10th SBCARS*, 2016, pp. 61–70.

[20] G. Rasool and Z. Arshad, "A review of code smell mining techniques," *J. Softw.: Evol. Process*, vol. 27, no. 11, pp. 867–895, 2015.

[21] F. A. Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Emp. Softw. Eng. (ESE)*, vol. 21, no. 3, pp. 1143–1191, 2016.

[22] D. Di Nucci, F. Palomba, D. A. Tamburri, A. Serebrenik, and A. De Lucia, "Detecting code smells using machine learning techniques: are we there yet?" in *25th SANER*, 2018, pp. 612–621.

[23] A. Dunsmore, M. Roper, and M. Wood, "The development and evaluation of three diverse techniques for object-oriented code inspection," *IEEE Trans. Softw. Eng. (TSE)*, vol. 29, no. 8, pp. 677–686, 2003.

[24] B. Brykczynski, "A survey of software inspection checklists," *ACM SIGSOFT Softw. Eng. N.*, vol. 24, no. 1, p. 82, 1999.

[25] C. Parnin, C. Görg, and O. Nnadi, "A catalogue of lightweight visualizations to support code smell inspection," in *4th SOFTVIS*, 2008, pp. 77–86.

[26] F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, and A. De Lucia, "The scent of a smell: An extensive comparison between textual and structural smells," *IEEE Trans. Softw. Eng. (TSE)*, vol. 44, no. 10, pp. 977–1000, 2017.

[27] R. Oliveira, L. Sousa, R. de Mello, N. Valentim, A. Lopes, T. Conte, A. Garcia, E. Oliveira, and C. Lucena, "Collaborative identification of code smells: A multi-case study," in *39th ICSE, SEIP Track*, 2017, pp. 33–42.

[28] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, and A. De Lucia, "A large-scale empirical study on the lifecycle of code smell co-occurrences," *Inform. Softw. Tech. (IST)*, vol. 99, pp. 1–10, 2018.

[29] F. Palomba, M. Zanoni, F. A. Fontana, A. De Lucia, and R. Oliveto, "Smells like teen spirit: Improving bug prediction performance using the intensity of code smells," in *32nd ICSME*, 2016, pp. 244–255.