Investigating the Social Representations of the Identification of Code Smells by Practitioners and Students from Brazil

Rafael de Mello
PUC-Rio, Rio de Janeiro, Brazil
rmaiani@inf.puc-rio.br

Anderson Uchôa
PUC-Rio, Rio de Janeiro, Brazil
auchoa@inf.puc-rio.br

Roberto Oliveira
PUC-Rio, Rio de Janeiro, Brazil
UEG, Posse-GO, Brazil
roberto.oliveira@ueg.br

Daniel Oliveira
PUC-Rio, Rio de Janeiro, Brazil
doliveira@inf.puc-rio.br

Willian Oizumi
PUC-Rio, Rio de Janeiro, Brazil
woizumi@inf.puc-rio.br

Jairo Souza
UFAL, Alagoas, Brazil
jrmcs@ic.ufal.br

Baldoino Fonseca
UFAL, Alagoas, Brazil
baldoino@ic.ufal.br

Alessandro Garcia
PUC-Rio, Rio de Janeiro, Brazil
afgarcia@inf.puc-rio.br

ABSTRACT

Context: The identification of code smells is one of the most subjective tasks in software engineering. A key reason is the influence of collective aspects of communities working on this task, such as their beliefs regarding the relevance of certain smells. However, collective aspects are often neglected in the context of smell identification. For this purpose, we can use the social representations theory. Social representations comprise the set of values, behaviors, and practices of communities associated with a *social object*, such as the task of identifying smells. **Aim:** To characterize the social representations behind smell identification. **Method:** We conducted an empirical study on the social representations of smell identification by two communities. One community is composed of postgraduate students from different Brazilian universities. The other community is composed of practitioners located in Brazilian companies, having different levels of experience in code reviews. We analyzed the associations made by the study participants about smell identification, i.e., what immediately comes to their minds when they think about this task. **Results:** One of the key findings is that the community of students and practitioners have stronger associations with different types of code smells. Students share a strong belief that smell identification is a matter of measurement, while practitioners focus on the structure of the source code and its semantics. Besides, we found that only practitioners frequently associate the task with individual skills. This finding suggests research directions on code smells may be revisited. **Conclusion:** We found evidence that social representations theory allows identifying research gaps and opportunities by looking beyond the borders of formal knowledge and individual opinions. Therefore, this theory can be considered an important resource for conducting qualitative studies in software engineering.

CCS CONCEPTS

• **Social and professional topics** → **Software maintenance**; • **General and Reference** → **Empirical studies**;

KEYWORDS

Social representations, code smells, qualitative research

ACM Reference format:
Rafael de Mello, Anderson Uchôa, Roberto Oliveira, Daniel Oliveira, Willian Oizumi, Jairo Souza, Baldoino Fonseca, and Alessandro Garcia. 2019. Investigating the Social Representations of the Identification of Code Smells by Practitioners and Students from Brazil. In *Proceedings of Brazilian Symposium on Software Engineering, Salvador, Brazil, 23–27 September, 2019 (SBES'19),* 10 pages.
https://doi.org/10.475/123_4

SBES'19, 23–27 September, 2019, Salvador, Brazil

ACM ISBN 978-1-4503-7651-8…$15.00
https://doi.org/10.475/123_4

1 INTRODUCTION

For the past two decades, software engineering researchers have been adopting qualitative research to study several topics [8]. Qualitative research aimed at exploring the overall context to understand a phenomenon, a situation or an event [33]. Case study [32], focus group [16], grounded theory [1], and action research [7] are examples of qualitative methods emerged from social sciences and used in software engineering research. In general, these methods help to study the complexities of human behavior, a key issue in software engineering [8]. However, the aforementioned methods do not enable structuring the common sense of communities, i.e., the system of beliefs, values, and behaviors in a particular community.

Social representations emerge from common sense, influencing how individuals in this community will behave and communicate [23]. Consequently, investigating social representations can be useful to identify opportunities for improving the practices surrounding the social object investigated. The theory of social representations have been used to investigate several social objects in different fields [12], such as education [22, 30] and health [14, 21].

The social representations theory aims at understanding how a certain community elaborates a social object. Among others, a *social object* may be any task performed by the members of this community. Considering the sociotechnical nature of software engineering, we can identify several social objects in the field. Examples of social objects in Software Engineering range from developers' tasks such as code reviews and requirements elicitation to specific methods and techniques, such as object-oriented design and refactoring.

In this paper, we explore the use of the social representation theory in the context of code smells identification. Code smells are symptoms of poor design observed in the low-level structure of a software system [2, 15, 29]. A code smell typically affects a code

element, such as a class or a method [36]. The code smell incidence hampers the readability and maintainability of the source code.

We use the term *identification* to distinguish the manual verification of code smells from their automated *detection* [28]. In other words, a code smell may be detected by a tool, but it may be identified or discarded by a reviewer. Moreover, automatic smell detection usually leads to several false negatives, which need to be manually identified by reviewers. In this sense, different studies investigated how smell identification is influenced by human aspects [4], such as the knowledge and the professional background of the reviewer [6, 28, 38]. Besides the importance of investigating these aspects, we claim it is also important to investigate the beliefs, values, and behaviors shared by the members of particular software engineering communities such as the academia and the industry.

For instance, let us consider a team of developers sharing the following *belief* [5]: smell detection tools are useless due to the waste of time and potential rework resulting from their massive amount of false warnings. Thus, this team has the *behavior* of discarding these tools and remaining with the manual smell identification only. At the same time, there is a research group sharing the *belief* that optimizing smell detection rules is sufficient for improving the resulting accuracy. For this group, an eventual overhead of false warnings is a natural consequence of technological issues. These conflicting beliefs and behaviors exemplify how identifying potential gaps among the social representations can be useful. Not revealing these gaps may hamper the transference of technology from academic to industry circles, a prevailing challenge in software engineering.

We recently conducted a preliminary study [5] on the social representations of smell identification. In this study, two local software engineering communities composed the samples. One of this communities was composed of post-graduation students. Another community was composed of practitioners. The potential contributions identified in the findings of this preliminary study led us replicating the protocol over a more representative sample, in order to perform more accurate analyses. For this propose, we resignified the scope of each community to cover a higher ethnographic diversity. We enlarged our sample to include students and practitioners located in four different regions of Brazil. As a result, we surveyed 27 postgraduation students and 23 practitioners from different Brazilian institutions.

The results of our study led us to identify more clear differences between the social representations of post-graduation students and practitioners regarding smell identification. For instance, we found that only the community of practitioners strongly associates smell identification with structural smells and bugs. Moreover, only the practitioners frequently associate smell identification to semantic smells and bugs. On the other hand, research-driven students strongly associate smell identification with measurable smells, i.e., those smells typically detected with specific metrics and thresholds.

The replication of the preliminary study also led us to identify clear similarities among the social representations of the communities. Both students and practitioners do not seem to disassociate smell identification from refactoring and the identification of design problems. They seem to see these tasks as a single intertwined task in practice. These findings seem to be misaligned with recurring characteristics of smell detection tools [9].

This paper reports our investigation on the social representation of code smells identification by students and practitioners. Despite the existence of key challenges for effectively applying the theory of social representations in the field [5], our experience conducting this investigation suggests that the social representations theory may be an useful tool to support qualitative studies in software engineering. The remainder of this paper is organized as follows. Section 2 presents the Theory of Social Representations. Section 3 describes our study settings. Section 4 characterizes each community investigated. Section 5 presents social representation analysis. Section 6 discusses threats to validity. Section 7 present an overview of the related work. Finally, Section 8 concludes the paper and suggests future work.

2 THE THEORY OF SOCIAL REPRESENTATIONS

At the beginning of the 20th century, the sociologist Emelie Durkheim have presented the theory of collective representations. For her, the human being is a sociable being due to group living. This way of living let individuals learning habits, customs and reproducing myths. For Durkheim, 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 theory of social representations influenced by Durkheim's theory. Moscovici affirms 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 theory of social representations 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 [23]. A social object corresponds to an object socialized by two or more individuals from a group. It can be, for instance, 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 with the purpose of behaving and communicating. 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 [23].

Howarth [12] argue that social representations is not a quiet thing, embodying and defining the experience of reality. Different social representations both extend and limit possibilities, being converted continuously into a social reality while continuously being reinterpreted. The social representations theory has been used to support research in many fields, including health and education [14, 21, 25, 30, 31]. Moreover, characterizing social representations of the same object by different communities may help to identify commonalities and gaps between them, which can be useful to enhance the communication among their members [5].

The study of social representations in different fields is grounded in analyzing free associations made by individuals from communities regarding the social objects investigated [14, 21, 25, 30, 31]. Free association is a technique used in psychoanalysis and frequently applied through individual semi-structured interviews [3]. 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, it is important to note that working on free association tasks is quite different from opinion gathering, obtained from typical survey questions [34]. 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.

3 STUDY SETTINGS

This section describes the settings of the study. Except by the population and sample, we made efforts to follow the same protocol applied in [5]. Section 3.1 presents both goal and research question. Section 3.2 describes the population and sample. Section 3.3 overviews our study instrumentation. Finally, Section 3.5 reports execution of the study.

3.1 Goal and Research Question

Our study aims at *characterizing the social representations of different software engineering communities regarding the identification of code smells*. We aim to answer the following research question (RQ):

RQ. *What are the social representations of the identification of code smells by communities of postgraduate students and practitioners?*

We opted by investigating postgraduate students once they: (i) conduct state-of-the-art research on the topic, and (ii) are frequently involved in conceiving and developing software engineering tools and technologies, including those for supporting the identification of code smells [9]. On the other hand, those tools are candidates, at some point, to be eventually used by practitioners, i.e., software engineering professionals playing different roles. Therefore, by answering our RQ, we aim at identifying opportunities for conceiving new technologies and improving current ones to better support the identification of code smells in practical settings.

3.2 Population and Sample

The social representations theory is grounded in the common sense emerged from particular communities or groups. The members of a community share sociocultural aspects, having the opportunity of exchanging knowledge at some level. A community may be, e.g., the employees of a particular organization or a specific type of professional from a particular city or country. Thus, a member of a community may not be able to directly and frequently interact with the other member. However, all of them are influenced by common sense emerged from this community. Besides, it is important to note that individuals may be members of one or more communities.

The investigation performed in local communities in the preliminary study was an important step to identify the pertinence of investigating country-wide communities. In this study, we aim at listening to communities of research and practice of software engineering located in Brazil. In total, 50 individuals had participated in the study. These individuals represent a considerable ethnographic diversity from the perspective of our study. They have different levels of background and knowledge. Besides, they are located in different regions of Brazil. Figure 1 shows the geographic distribution of the participants by region.

Figure 1: Distribution of the study participants among the regions of Brazil

The research community is represented by postgraduate students. They are Master and Doctoral students from the Department of Informatics of three Brazilian universities. Each university is located in a different region of the country. All 27 subjects that represent this community investigate topics in software engineering. Among others, these topics include software maintenance ones, such code smells and design problems.

The population of practitioners is composed of software developers actively working in the Brazilian industry. We obtained a sample of 24 practitioners from four Brazilian software companies located in different regions of the country. Two of these companies deliver software solutions to the Brazilian government, typically web applications. A third one is a large-scale company from the automotive sector, having more than 1,000 employees. This company owns the development of several IT solutions for supporting their processes, including vehicle assembling. A fourth company is a software startup having predominately private clients. This company typically delivers web and mobile applications.

3.3 Instrumentation

We designed a questionnaire with two categories of questions, intended to gather quantitative and qualitative data from the subjects. The first category comprises questions for gathering subjects' background and specific opinions. The second category consists of questions for supporting the free association task [3] and, therefore, our analyses of the social representations (RQ). The former category will help us to characterize the communities investigated. Their questions will also support our broader investigation of human aspects affecting smell identification (Section 3.1).

Table 1 presents the questions (translated from Portuguese) in their respective blocks. The first category of questions is part of the B_1, B_2, B_4 and B_5 blocks, while the free association questions are grouped in the B_3 block. We intentionally organized and grouped the questions to avoid biasing participants' responses along the free association task. Each block of questions was presented to the subject only after answering the previous one.

After identifying the familiarity of the respondent with the term *code smells* (B_2), the participants should perform the free association task (B_3). 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 about the opinion (B_4) and knowledge (B_5) regarding the social object of study, i.e., the *identification of code smells*.

We designed the questionnaire to be applied in a controlled environment, without access to external sources. At least one researcher should supervise this environment, assuring the subjects comply with the requirements to the proper realization of the free association task. For instance, the subjects should be also instructed and monitored to answer the free association task immediately. Before the subjects started filling out the questionnaire, they were enforced to follow the sequence of questions and not change their previous answers. The study was carried out on five different executions from June 2017 to March 2018. The environments were controlled as planned in all executions. Besides, the researchers identified that all subjects followed the instructions.

Table 1: Questionnaire Items

Block	Question
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.
B_2	Q4. Are you familiar with the term code smells? ( ) Yes ( ) No
	Q5. For you, what is a code smell?
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?
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
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.

3.4 Data Analysis

From the associations defined by the students and practitioners, we evaluate the frequency and the average orders of evocation (AOE) related to each association. We calculate the frequency by counting how many times the association was quoted in the community. AOE is the sum of the positions in which each respondent had quoted an association divided by the frequency of the association. The lower AOE of an association is, more promptly the community is to associate the social object investigated to the corresponding term and vice-versa. For instance, suppose that three participants quoted the term *complex* in a certain free association task. However, two participants quoted it in the first place, and a third participant quoted it in the fourth place. Therefore, the frequency of the association *complex* will be 3, and its AOE is 2 $((1 + 1 + 4)/3)$.

The strength of each association indicates in which extent it is more or less relevant to the social representations of a community. It is measured through the four quadrant analysis [3]. For identifying in which quadrant each association should be included, the following thresholds are calculated: (i) the mean frequency of the associations, and (ii) the ratio between the sum of the AOE and the sum of the frequencies of the associations. Figure 2 shows the distribution of the four quadrants and the importance of each one to social representation. The strongest quadrant is the central system, composed by the core elements, i.e., the associations which were more frequently and more promptly evoked.

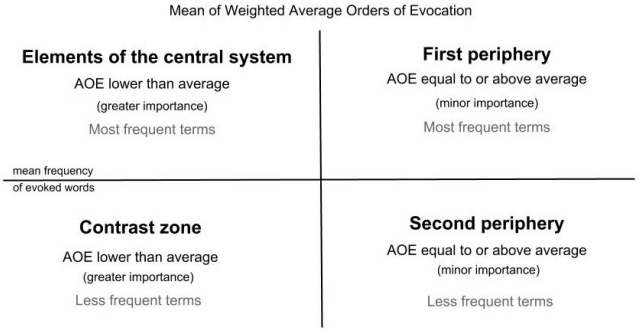

Figure 2: Structure of the four-quadrant analysis

3.5 Executions

The study was conducted in two major periods of executions. The first one was conducted from June 2017 to March 2018, involving the participation of 30 individuals. The second period was conducted from March 2019 to April 2019. In this period, we count with the participation of more 20 individuals.

We made effort to apply the questionnaire in the subjects' environments. The students answered the questionnaire in the their classrooms or in their research laboratories. The practitioners answered the questionnaire at their respective companies. The environments were controlled as planned in both cases. Therefore, the study involved five researchers spread in the country. All of them certified that all 50 subjects followed the instructions for answering the questionnaire. The questionnaire was applied in Portuguese.

4 CHARACTERIZATION OF THE COMMUNITIES

We aim at characterizing the social representations of the identification of code smells by two different communities: Brazilian postgraduate students and Brazilian software engineering practitioners. We opted by studying Brazilian communities due to the observed interest of different Brazilian research groups in this topic. They have been publishing studies regarding code smells in different forums, including national (e.g., SBES and SBCARS) and international conferences (e.g., ICSE and ESEM). For this purpose, we made efforts to recruiting samples that would reflect the diversity of the communities investigated. In this sense, it is important to characterize the communities investigated for identifying in which extent members from these communities have commonalities and differences. This characterization will be useful to understand the different contexts in which the social representations emerge.

In this section, we characterize both communities by using the personal background and opinions given by the respondents (see Table 1). The results of the free association task are used in Section 5 to perform the Social Representation analysis. We characterize both communities as follows.

4.1 Academic and Professional Background

Both communities are composed of individuals sharing a similar distribution of higher academic degree. Most of the participants hold bachelor's degrees or similar degrees, whereas the rest hold master's degrees except by two practitioners. Although experienced in software development, these subjects did not have a bachelor degree in the field. Therefore, the main difference among the communities is that students are involved in a Master or a Doctoral program. Besides, a third of the students are conducting investigations related to code smells. These investigations include research topics addressing the detection of code smells, the identification of design problems and refactoring.

The majority of the individuals from both communities declared previous experience with software development (96%). In both communities, the experience varies a lot, ranging from a few academic projects until several projects in the industry. Particularly, we observed that a considerable part of the post-graduation students also works in the software industry. We also found a similar distribution of experience in conducting code reviews among both communities.

4.2 Definition of code smells

Most of the students (85.15%) declared familiarity with the term "code smell". On the other hand, only 56.22% of practitioners declared familiarity with this term. Despite that, most of the subjects who were not familiar with code smells provided valid definitions of the term. All students and great most of the practitioners (83.33%) provided definitions of the term "code smell" somehow compatible with the definition available in the technical literature [2, 15, 29].

Most of the students pointed out the negative consequences of the incidence of smells in the source code (59.26%). More specifically, they referred to possible structural problems and their impact on the software quality and its maintenance. For instance, one student reported that code smells are "(...) characteristics identified in the code that may represent or generate a maintenance problem", whereas another student defined code smell as "(...) a symptom of a problem in the source code". Two other students related their definitions to bad programming practices, providing examples of code smells. For instance, one of them provided "(...) a class with too many attributions" as an example.

As the students, most of the practitioners that provided valid answers quoted negative consequences of code smells (65%) in their definitions. These consequences include, e.g., the impact of code smells in the source code quality. However, they also frequently mentioned concerns with the understanding of the source code (4), which were not quoted among the students. For instance, one practitioner defined code smells as "(...) an implementation problem which hampers the reading and the analysis of certain code snippets". Moreover, about a half of the practitioners (45%) also mentioned root causes of code smells, i.e., the incidence of bad programming practices and anti-patterns. For instance, another practitioner defined code smells as "(...) some extraneous code snippet that does not follow the design patterns that should be used".

4.3 Relevance of code smell identification and removal

Figure 3 summarizes the perception of students and practitioners regarding the relevance of identifying and removing code smells to promote the structural quality of the source code. In general, we observe that the students and practitioners tend to agree that identifying and removing code smells is a matter of structural quality.

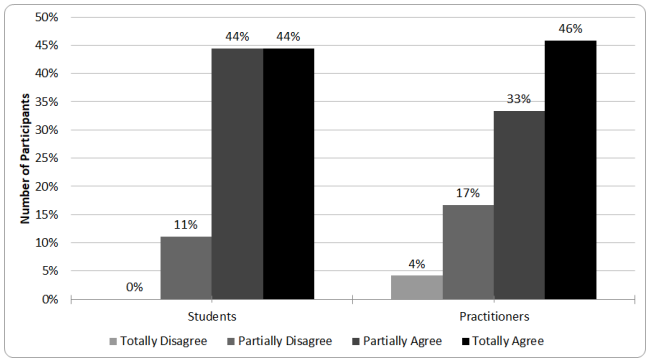

Figure 3: Perception about identifying and removing code smells to promote the structural quality of the source code

Figure 4 summarizes the perception of students and practitioners on the relevance of identifying and removing code smells for understanding the source code. We found that the vast majority of the members from both communities agree that identifying and removing smells promotes the understanding of the source code. Therefore, the results indicate an overall perception that it is worthwhile to identify and remove code smells. This perception addresses the characteristics of code smells observed in the technical literature [2, 15, 29]. However, the distributions of agreement levels also indicate that attributing a protagonist role to the identification and removal of code smells is questionable. Due to the small size of some distributions, the chi-square test could not be applied.

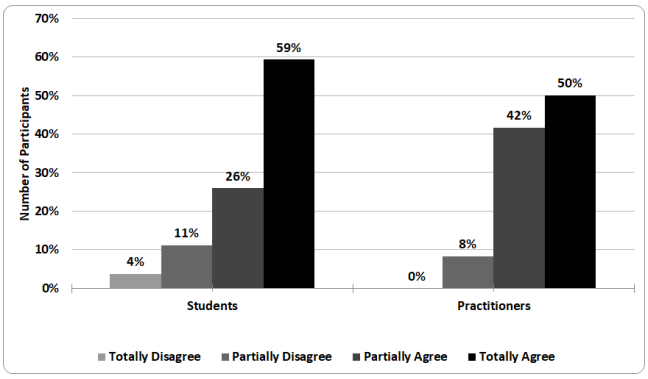

Figure 4: Perception about identifying and removing code smells to promote the understanding of the source code

4.4 Knowledge of code smell types

We also asked the subjects which code smell types they remember how to identify. The respondents are 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 [11, 17]. Then we classified the code smells by the predominant scope on which they apply – measurable, structural or semantic – adapting the classification proposed in [19]

Measurable smells have their detection based on measures of internal attributes of the program elements. *Structural smells* are based on the structural properties and relationships that define the program elements. *Semantic smells* concern the semantic problems associated with program elements, which includes, among others, the proper naming of code elements according to the requirements specifications or other terms of the system vocabulary. We also classified each code smell type by granularity – inter-class or intra-class. *Intra-class* smells are anomalous code structures which affect a particular class of the system. *Inter-class smells* are those which affect multiple classes together [5]. Finally, we calculated the frequency of each type of code smell by each community and analyzed the distributions using quartiles [20]. Table 2 presents the most frequent types of code smell mentioned by students.

Table 2: Code smell types mentioned by students

Code smell types	Granularity	Scope	Frequency
God Class	Inter-class	Measurable	15
Long Method	Intra-class	Measurable	13
Feature Envy	Inter-class	Measurable	6
Duplicated Code	Inter-class	Structural	6
Refused Bequest	Inter-class	Measurable	5
Long Parameter List	Intra-class	Measurable	4
Lazy Class	Inter-class	Structural	3
Switch Statement	Intra-class	Structural	3
Speculative Generality	Intra-class	Structural	2
Shotgun Surgery	Inter-class	Structural	2
Comments	Intra-class	Semantic	2
Data Clumps	Intra-class	Semantic	2

The most frequent code smell types mentioned by students (third quartile) are measurable smells: *God Class* (15), *Long Method* (13), *Feature Envy* (6), *Duplicated Code* (6), *Refused Bequest* (6), and *Long Parameter List* (4). Only these six types of code smells represent more than 60% of all the mentions. On the other hand, the knowledge about the identification of structural smells is less frequent, except by *Duplicated Code*, a type of code smell which detection is frequently automated. Regarding semantic smells, only *Comments* and *Data Clumps* were mentioned. Moreover, we can observe that there is a balanced distribution of mentions to intra-class and inter-class smells.

Table 3 presents the most frequent types of code smell mentioned by practitioners. By considering the third quartile, the most frequent types of code smell mentioned by practitioners were: *Feature Envy* (6), *God Class* (6), *Switch Statement* (5), *Long Method* (4), and *Duplicated Code* (3), i.e., the same measurable smells mentioned by the students. They represent 59% of the total amount of mentions. Thus, practitioners also reported less frequent knowledge about the identification of structural smells than measurable ones. Besides, no semantic smell was mentioned. Moreover, we can observe more mentions of inter-class smells than intra-class ones.

Table 3: Code smell types mentioned by practitioners

Code smell types	Granularity	Scope	Frequency
Feature Envy	Inter-class	Measurable	6
God Class	Inter-class	Measurable	6
Switch Statement	Intra-class	Structural	5
Long Method	Intra-class	Measurable	4
Duplicated Code	Inter-class	Structural	3
Dispersed Coupling	Inter-class	Measurable	2
Intensive Coupling	Inter-class	Measurable	2
Shotgun Surgery	Inter-class	Structural	2
Lazy Class	Inter-class	Structural	2
Middle Man	Inter-class	Structural	2

4.5 Discussion

Our results indicate that the community of students and the community of practitioners are not too different from a perspective of specific knowledge and professional experience. Most of the members from both communities are 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 both students and practitioners about code smell types is concentrated in measurable ones.

We emphasize that the investigated communities are dynamic, being continuously changed. Such dynamism includes the exchanging among its members. Former students may shift to industry, as well as experienced professionals may shift to academia. However, each community has its own world that will influence its members. For instance, even new postgraduate students are stimulated to see software engineering practice from a researcher perspective. Consequently, these individuals begin sharing new beliefs and values that will influence their behavior. Therefore, despite of the similarities found in the individual characteristics, we emphasize that each community has its collective background, from which emerges the social representations discussed in the next section.

5 SOCIAL REPRESENTATIONS OF THE IDENTIFICATION OF CODE SMELLS

This section describes and discusses the analysis of the social representations of smell identification by each community investigated. The communities of students and practitioners, both characterized in Section 4. For this purpose, the participants of the study performed a free association task (see Section 3.3). 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. Thus, the researchers eventually used the respondents' answers to the other items of the questionnaire, such the explanation regarding the more relevant association (see Section 3.3). This verification aims to reach the actual meaning of the terms quoted. In cases of doubt regarding the meaning of a term, the association was not grouped. After concluding the grouping activity, one researcher composed the final set of associations for each community. Then, the final sets of associations were submitted to the four-quadrant analysis, following the methodology presented in Section 3.4.

Table 4 exemplifies how the groupings were performed. It presents the set of terms clustered for depicting the *design problem* association in each community. One can see that practitioners and students evoked a different set of terms. However, it was verified that each term was evoked addressing a similar meaning. The researchers also grouped the different types of code smells quoted based in types of properties on which they apply (see Section 4).

Table 4: Example of terms clustered into a single association

Students	Practitioners	Association
fixing the problem (1) solving the problem (1) refactoring (8)	refactoring (5) rewrite the code (1)	removing smells
anti-pattern (1) design problem (1) structural problem (1) bad structured code (2)	design problems (1) fail on following a pattern (1) jerry-rig (*gambiarra*)(1) problems (1)	design problem

5.1 Students

Most of the 27 students made five quotes, resulting in 105 associations, including repetitions. After performing analyses, we consolidated a set of 40 distinct associations. From these, 22 associations were quoted only once. Therefore, they were excluded from further analysis. Table 5 presents the four-quadrant analysis, distributing the associations – translated from the Portuguese – by frequency and the average order of evocation (AOE). In this community, the associations with the frequency equal or higher than 5 and with AOE lower than 2.80 composed the central system.

We can depict some interesting findings from Table 5. The central system (top-left quadrant) reveals that the community of students promptly associates the smell identification to *removing smells* and to the incidence of *design problems*. This community also has a strong belief that smell identification is a matter of *inspection*, recognizing the need for further manual analysis once smell suspects are automatically detected.

Table 5: Four-quadrant analysis of the social representations of smell identification by students

Central system			First periphery		
Frequency ≥ 5	AOE < 2.80		Frequency ≥ 5	AOE ≥ 2.80	
	Frequency	AOE		Frequency	AOE
Removing smells	10	2.40	Design	8	3.75
Measurable smells	10	2.10	Source code	6	3.33
Inspection	8	1.38	Structural smells	7	3.14
Design problem	5	2.20	Anomaly	4	3.00
Contrast zone			**Second periphery**		
Frequency < 5	AOE < 2.80		Frequency < 5	AOE ≥ 2.80	
	Frequency	AOE		Frequency	AOE
Bug	4	2.25	Problem	3	3.00
Detection	4	2.25	Quality	3	4.00
Individual Skills	4	1.50	Maintenance	2	3.00
Strategy	2	1.50	Cohesion	2	3.50
			Reuse	2	4.50
			Context knowledge	2	3.00

The associations made by the students included the evocation of several types of code smells. We observe that students promptly associate the smell identification with *measurable smells*. The detection of these smells is typically supported by a combination of metrics and their thresholds. In this sense, *detection* is an low frequent association but also relevant (third quadrant). This term comprises formal rules and automated support to detect different smell types. On the other hand, the association with structural smells is also frequent but less relevant (top-right quadrant). This finding suggests that the community of students believe that identifying code smells is more about measuring code rather than identifying design violations. Besides, no association with semantic smells was made by students.

In general, the associations made by the students are predominantly technical and partially reflects the concern of the research community in automating the process of smell identification. The detection of measurable smells is largely supported by existing solutions despite prevailing issues with precision and recall. Curiously, different from the solutions available, postgraduate students seem to think of smell identification and its removal as intermingled tasks that require inspections, i.e., manual analysis. However, this concern with inspections may not be extended to the identification of semantic issues.

5.2 Practitioners

Most of the 23 practitioners quoted five terms, resulting in 80 associations, including repetitions. After analyses, we consolidated a final set composed of 24 distinct associations. From these, eight associations were quoted only once. Therefore, they were excluded from the analysis of social representations. Table 6 presents the analysis of the four quadrants for the practitioners' community. Associations with the frequency equal or higher than 4 and with AOE lower than 2.55 compose the central system of the social representations.

The central system (top-left quadrant) reveals that the community of practitioners frequently and promptly associate the identification of code smells with the practice of *removing*. This community also share a strong belief that the prevalence of code smells is associated with the incidence of design problems and even *bugs*. The relationship between code smells, and design problems have been largely discussed in the technical literature. Although bug is a buzzword used in software engineering research and practice with

Table 6: Four-quadrant analysis of the social representations of smell identification by practitioners

Central system			First periphery		
Frequency≥4	AOE <2.55		Frequency≥4	AOE≥2.55	
	Frequency	AOE		Frequency	AOE
Removing smells	6	2.33	Inspection	12	2.58
Structural smells	5	2.00	Individual skills	8	2.88
Bugs	6	2.50	Semantics	5	2.60
Design problem	4	1.50	Code complexity	4	3.75
Contrast zone			**Second periphery**		
Frequency<4	AOE<2.55		Frequency<4	AOE≥2.55	
	Frequency	AOE		Frequency	AOE
Source code	3	2.33	Design	2	3.00
Measurable smells	3	1.00	Performance problems	2	3.50
Detection	3	2.00	Problems	2	5.00
Maintenance	2	2.50			
Testing	2	2.50			

several meanings [26, 27, 39], we found that the references to bug made by practitioners addresses problems in the execution of the source code and semantic issues. Therefore, we may interpret that practitioners believe that code smells would hamper maintenance activities by contributing to the incidence of bugs [10].

Regarding the types of code smells, we found that practitioners promptly associate the smell identification with *structural smells*. Besides, they often associate the smell identification to *semantic* issues (second quadrant). Although semantic smells are not frequently cataloged or supported by detection tools, the identification of semantic problems in the source code is typically supported by code inspections. On the other hand, *measurable smells* is a less frequent association. These results indicate that the community of practitioners share a belief that smell identification is a matter of understanding the source code and its design rather than detecting issues in metrics. Consequently, it indicates the concern of this community on avoiding or prioritizing fixes for smells that indicate relevant, major design problems rather than tiny code smells affecting only a program statement or the inner body of a design-irrelevant method.

The set of associations presented in the second quadrant strengths the aforementioned findings. The associations in this quadrant somehow address the human capacity of analyzing the code: *inspections*, *individual skills*, *semantics*, and *code complexity*. Moreover, one can see that the inspections and semantics have AOEs pretty close to the maximum AOE accepted in the first quadrant (2.55). Regarding the individual skills, the practitioners evoked non-technical ones such as patience and attention. These associations indicate that the community of practitioners believe that it is worthwhile to perform an accurate analysis for identifying code smells.

5.3 Discussion

In Section 4, we found that practitioners and students have similar technical knowledge and opinions about code smells and its identifications. However, the results of social representations analysis revealed that these communities have more differences than similarities in their associations. In a big picture, both communities have the belief that code smells deserves to be properly identified and removed, mitigating the incidence of design problems. Even removing a single code smell may be a costly activity, affecting several code elements. Moreover, it also brings the risk of rework due to the incidence of false positives. The unnecessary maintenance of the source code also increases the risk of introducing other code smells and even bugs. However, they diverge on the emphasis that should be given on performing the task.

The community of students strongly associates the smell identification with measurable smells, for which detection is grounded in metrics and thresholds largely applied in detection tools [24, 35, 37]. This association is compatible with the formal knowledge of code smells reported by students (see Section 4). On the other hand, the community of practitioners shows a high association with structural smells, for which detection rules are more grounded in expert judgment than metrics. Besides, only practitioners associate the smell identification with semantic smells, although they do not have showed formal knowledge about them (see Section 4). Interestingly, we note that the current catalogs and tools available for supporting the identification of code smells barely address semantic ones.

Therefore, the social representations indicate the need for rethinking certain directions on developing technologies for supporting the smell identification. For instance, let us consider the complexity involved in identifying semantic smells. The mere verification of a single method name requires checking not only its correctness and compliance with the system vocabulary but also if other classes are using the same name with different purposes (consistency). Besides, it should be verified whether the method name does not lead to ambiguous interpretations, i.e., the lack of meaningful names that correspond to the real behavior of the method. In this sense, one possible alternative for overcoming this challenge may be investing research efforts on guiding the code reviews performed by practitioners through recommendation systems. Recommendation systems may also provide customized guidelines for supporting the decision of the developers on fixing semantic smells. Such guidelines may include the support to impact analysis of the different decisions that could be taken.

Despite the difference of the communities investigated, we can perform some comparison among the social representations of communities of research and practice from this study and the preliminary study [5]. If we compare the local industry/academia (previous study) with the Brazilian industry/academia (this study), their four-quadrant analyses resulted in considerable differences. For instance, in the Brazilian community of practitioners, *bug* is a strong association, while in the local community of practitioners is not. Similarly, *design problem* and *removing smells* are strong associations among Brazilian students, which was not observed among the local community of students surveyed in the previous study. Besides, the replication presented in this paper resulted in higher mean frequencies and several new associations. Therefore, it was possible to identify more accurately the strongness of the associations.

The findings of this study indicate that investigating social representations is a useful resource to expand the possibilities of performing qualitative research in the field. Especially in the case of practitioners, one can see that gathering technical answers and opinions was not sufficient to identify the collective beliefs and values shared by the community. On the other hand, the free association tasks allowed individuals to expose actual concerns that go beyond the boundary of formal definitions of the research topic.

6 THREATS TO VALIDITY

We recruited postgraduate students to represent the Brazilian academic community, which may be considered a threat. However, our experience in interacting with different research groups suggest that investigations on technologies for supporting the identification of code smells are typically conducted in the context of Master and Doctoral Theses.

We made efforts to involve as much as possible all Brazilian research groups deeply investigating code smells. Unfortunately, we could not find research groups located in the North of Brazil concentrating their investigation on code smells. Thus, it also affected the composition of the sample of practitioners, once the researchers were only able to recruit practitioners located in their respective regions.

The nature of the free association task requires a level of control that hampers reaching large samples such as in opinion surveys. In this sense, it was needed to train several researchers to run the study in loco with students from different universities spread in the country. Even though the sample sizes of each community is relatively small, it is important to note that they are composed of individuals considerably diverse in background and experience, as presented in Section 4.

In free association tasks, subjects should provide their quotations immediately, avoiding the bias of elaborating opinions and formal knowledge. For this propose, interviews are typically conducted. We opted by applying a questionnaire due to the opportunity of controlling in person the answering to each item, including the free association task. In this way, we observed that the subjects followed all instructions provided by the researchers 3.3. Besides, our option by questionnaire was also motivated by time constraints in industrial settings.

7 RELATED WORK

In this paper, we applied the social representation theory to characterize the beliefs, behaviours, and practices regarding the identification of code smells. The study presented in [5] is the only previous work we find investigating the social representations of a software engineering activity. This preliminary study, published in a workshop, involved 13 software developers working in a particular city and 17 students from a post-graduation course.

In this study, each subject was quoted to answer what immediately comes to his/her mind when thinking about smell identification. The authors found that both communities share a similar knowledge of code smells and similar opinions regarding the relevance of identifying and removing code smells. However, after analyzing the associations, the authors found considerable differences between the social representations of the communities. However, due to the small sample sizes and the low frequency of the associations, the discussion of the findings were restricted to first impressions.

An important contribution of [5] resides on discussing five lessons learnt by the authors on applying the social representations theory in software engineering. These lessons address challenges such as the diffuse terminology of the field and the predominance of technical associations. Therefore, this work can be considered a useful reference to perform investigations of social representations of social objects from software engineering. As presented in Section 2, the social representations from a collective emerge from three different spheres of pertinence: the intersubjective, the trans-subjective and the (individual) subjective sphere. In this sense, the individual subjectivity of the identification of code smells has been investigated in previous work. These investigations are predominantly grounded in controlled experiments and surveys.

A recent study [13] investigated the level of agreement of several software developers regarding the incidence of different code smells. For this propose, 40 developers were divided into four groups. All participants from each group analyzed the possible incidence of code smells in 15 different code snippets. In most cases, the level of agreement found was significantly low. Moreover, different arguments were provided by the subjects to take the same decision. The authors concluded that developers tend to have different perceptions about the incidence of code smells. Yamashita and Moonen [38] conducted an exploratory survey with developers from different countries regarding code smells. By listening to developers with different demographic characteristics, they found different interpretations of code smell and different perceptions about the impact of code smells to the software design and its overall quality.

Mäntylä et al. [18] investigated the effect of demographics on the evaluation of code smells previously detected in software modules of a Finnish software company. They found initial evidence that the conflicting perceptions about the incidence of code smells were associated with the professional experience and the professional role of the reviewers. Palomba et al. [29] investigated the developers' perception of poor design and implementation choices by investigating developers with previous experience in the software modules and outsiders. The researchers concluded that developer's experience and the previous knowledge of the modules to be reviewed influences their perceptions.

More recently, we conducted empirical studies to observe the influence of three human factors in the precision of the code smell identification: the professional background; the module knowledge; and the conduction of single or pair reviews [6, 28]. The executions of the study were conducted in industrial settings, having software professionals as subjects and real software projects as objects of study. We found evidence that setting and combining these factors may influence the precision of the identification tasks.

8 CONCLUSION AND FUTURE WORK

In the last decades, software engineering research had been benefited from the adoption of different qualitative research methods. The social representations theory proposes a new perspective on doing qualitative research, grounded in common sense. It comprises the system of beliefs, behaviours, and attitudes of communities regarding social objects, such as software engineering tasks. In this way, investigating social representations is a useful tool to guide the research agenda, identifying opportunities for bridging the gap between research and practice.

In this paper, we conducted a preliminary study using the social representations theory for investigating the identification of code smells from the perspective of two communities located in Brazil: post-graduation students and software engineering practitioners. We found the communities have different social representations

of the task, although they share similar knowledge of code smells and opinions regarding the task. Among others, the findings of the presented study may be used in the development of more effective technologies for supporting the identification of code smells. In this way, we intend to refine the current findings by increasing the representativeness of the samples, which includes cover all the regions of the country. In a second step, we intend to replicate the study in other countries.

Different from the observed in fields such as health and education, we observed that the social representations address more technical attributes than qualitative ones. At the same time, the diffuse terminology in the field demands a rigorous analysis for understanding the context of the associations made by the subjects. We also observed in practice the need for rigorously conducting the free association task. Based on these lessons, we intend to evolve the planning and execution of future investigations of social representations of other software engineering activities highly influenced by human aspects. We also plan to use the experience gathered in this study to investigate the social representations of other software engineering activities highly influenced by human aspects.

ACKNOWLEDGMENTS

This work is supported by CAPES/Procad (175956), and CNPq (153363/2018-5, 141285/2019-2). The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Steve Adolph, Wendy Hall, and Philippe Kruchten. 2011. Using grounded theory to study the experience of software development. *Emp. Softw. Eng. (ESE)* 16, 4 (2011), 487–513.
[2] Gabriele Bavota, Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, and Fabio Palomba. 2015. An experimental investigation on the innate relationship between quality and refactoring. *J. Syst. Softw. (JSS)* 107 (2015), 1–14.
[3] Lionel Dany, Isabel Urdapilleta, and Grégory Lo Monaco. 2015. Free associations and social representations: some reflections on rank-frequency and importance-frequency methods. *Quality & Quantity* 49, 2 (2015), 489–507.
[4] Rafael de Mello, Roberto Oliveira, Leonardo Sousa, and Alessandro Garcia. 2017. Towards effective teams for the identification of code smells. In *10th CHASE*. 62–65.
[5] Rafael de Mello, Anderson Uchôa, Roberto Oliveira, Daniel Oliveira, Baldoino Fonseca, Alessandro Garcia, and Fernanda de Mello. 2019. Investigating the social representations of code smell identification: a preliminary study. In *12th CHASE*. 53–60.
[6] Rafael Maiani de Mello, Roberto Felicio Oliveira, and Alessandro Fabricio Garcia. 2017. On the Influence of Human Factors for Identifying Code Smells: A Multi-Trial Empirical Study. In *11th ESEM*. 68–77.
[7] Paulo Sergio Medeiros dos Santos and Guilherme Horta Travassos. 2009. Action research use in software engineering: An initial survey. In *3rd ESEM*. 414–417.
[8] Tore Dybå, Rafael Prikladnicki, Kari Rönkkö, Carolyn Seaman, and Jonathan Sillito. 2011. Qualitative research in software engineering. *Empirical Software Engineering* 16, 4 (2011), 425–429.
[9] Eduardo Fernandes, Johnatan Oliveira, Gustavo Vale, Thanis Paiva, and Eduardo Figueiredo. 2016. A review-based comparative study of bad smell detection tools. In *20th EASE*. 1–18.
[10] Isabella Ferreira, Eduardo Fernandes, Diego Cedrim, Anderson Uchôa, Ana Carla Bibiano, Alessandro Garcia, João Lucas Correia, Filipe Santos, Gabriel Nunes, Caio Barbosa, and others. 2018. The buggy side of code refactoring: Understanding the relationship between refactorings and bugs. In *40th ICSE, Poster track*. ACM, 406–407.
[11] Martin Fowler. 1999. *Refactoring*. Addison-Wesley Professional.
[12] Caroline Howarth. 2006. A social representation is not a quiet thing: Exploring the critical potential of social representations theory. *Br. J. Soc. Psychol. (BJSP)* 45, 1 (2006), 65–86.
[13] Mario Hozano, Nuno Antunes, Baldoino Fonseca, and Evandro Costa. 2017. Evaluating the Accuracy of Machine Learning Algorithms on Detecting Code Smells for Different Developers. In *Proceedings of the 19th International Conference on Enterprise Information Systems*. 474–482.
[14] Hélène Joffe and Nadia Bettega. 2003. Social representation of AIDS among Zambian adolescents. *J. Health Psychol. (JHP)* 8, 5 (2003), 616–631.
[15] Foutse Khomh, Massimiliano Di Penta, and Yann-Gael Gueheneuc. 2009. An exploratory study of the impact of code smells on software change-proneness. In *16th WCRE*. 75–84.
[16] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. 2008. The focus group method as an empirical tool in software engineering. *Guide to advanced empirical software engineering* (2008), 93–116.
[17] Michele Lanza and Radu Marinescu. 2006. *Object-oriented metrics in practice*. Springer Science & Business Media.
[18] Mika V Mäntylä, Jari Vanhanen, and Casper Lassenius. 2004. Bad smells-humans as code critics. In *20th ICSM*. 399–408.
[19] Naouel Moha, Yann-Gael Gueheneuc, Laurence Duchien, and Anne-Francoise Le Meur. 2010. Decor: A method for the specification and detection of code and design smells. *IEEE Trans. Softw. Eng. (TSE)* 36, 1 (2010), 20–36.
[20] David S Moore and Stephane Kirkland. 2007. *The basic practice of statistics*. Vol. 2. WH Freeman New York.
[21] Nicola Morant. 2006. Social representations and professional knowledge: The representation of mental illness among mental health practitioners. *Br. J. Soc. Psychol. (BJSP)* 45, 4 (2006), 817–838.
[22] Geraldo Eustáquio Moreira and Ana Lúcia Manrique. 2014. Challenges in Inclusive Mathematics Education: Representations by Professionals Who Teach Mathematics to Students with Disabilities. *Creative Education* 5, 7 (2014), 470–483.
[23] Serge Moscovici. 1988. Notes towards a description of social representations. *Eur. J. Soc. Psychol. (EJSP)* 18, 3 (1988), 211–250.
[24] Emerson Murphy-Hill and Andrew P Black. 2010. An interactive ambient visualization for code smells. In *5th SOFTVIS*. 5–14.
[25] Michael Murray. 2002. Connecting narrative and social representation theory in health research. *Social Science Information* 41, 4 (2002), 653–673.
[26] Adrian Nistor, Tian Jiang, and Lin Tan. 2013. Discovering, reporting, and fixing performance bugs. In *10th MSR*. 237–246.
[27] Masao Ohira, Yutaro Kashiwa, Yosuke Yamatani, Hayato Yoshiyuki, Yoshiya Maeda, Nachai Limsettho, Keisuke Fujino, Hideaki Hata, Akinori Ihara, and Kenichi Matsumoto. 2015. A dataset of high impact bugs: Manually-classified issue reports. In *12th MSR*. 518–521.
[28] Roberto Oliveira, Leonardo Sousa, Rafael de Mello, Natasha Valentim, Adriana Lopes, Tayana Conte, Alessandro Garcia, Edson Oliveira, and Carlos Lucena. 2017. Collaborative identification of code smells: A multi-case study. In *39th ICSE, SEIP Track*. 33–42.
[29] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Andrea De Lucia. 2014. Do they really smell bad?. In *30th ICSME*. 101–110.
[30] Hannu Räty, Katri Komulainen, and Laura Hirva. 2012. Social representations of educability in Finland: 20 years of continuity and change. *Social Psychology of Education* 15, 3 (2012), 395–409.
[31] Dener Carlos dos Reis, Andréa Gazzinelli, Carolina Angélica de Brito Silva, and Maria Flávia Gazzinelli. 2006. Health education and social representation: an experience with the control of tegumentary leishmaniasis in an endemic area in Minas Gerais, Brazil. *Cadernos de saude publica* 22, 11 (2006), 2301–2310.
[32] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Emp. Softw. Eng. (ESE)* 14, 2 (2009), 131.
[33] Carolyn B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Trans. Softw. Eng. (TSE)* 25, 4 (1999), 557–572.
[34] Marco Torchiano, Daniel Méndez Fernández, Guilherme Horta Travassos, and Rafael Maiani de Mello. 2017. Lessons learnt in conducting survey research. In *5th CESI*. 33–39.
[35] Nikolaos Tsantalis, Theodoros Chaikalis, and Alexander Chatzigeorgiou. 2008. JDeodorant: Identification and removal of type-checking bad smells. In *12th CSMR*. 329–331.
[36] Michele Tufano, Fabio Palomba, Gabriele Bavota, Rocco Oliveto, Massimiliano Di Penta, Andrea De Lucia, and Denys Poshyvanyk. 2015. When and why your code starts to smell bad. In *37th ICSE*. 403–414.
[37] Santiago A Vidal, Claudia Marcos, and J Andrés Díaz-Pace. 2016. An approach to prioritize code smells for refactoring. *Automated Software Engineering (ASE)* 23, 3 (2016), 501–532.
[38] Aiko Yamashita and Leon Moonen. 2013. Do developers care about code smells?. In *20th WCRE*. 242–251.
[39] Shahed Zaman, Bram Adams, and Ahmed E Hassan. 2011. Security versus performance bugs: a case study on firefox. In *8th MSR*. 93–102.