Investigating the Social Representations of Code Smell Identification: A Preliminary Study

Rafael de Mello*, Anderson Uchôa*, Roberto Oliveira*, Daniel Oliveira*,
Baldoino Fonseca†, Alessandro Garcia*, Fernanda de Mello‡
*Informatics Department, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil
Email: {rmaiani, auchoa, rfelicio, doliveira, afgarcia}@inf.puc-rio.br
†Computing Institute, Federal University of Alagoas (UFAL), Brazil
Email: baldoino@ic.ufal.br
‡ Rioeduca, Brazil
Email: fernanda.mello@rioeduca.net

***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 a preliminary study on the social representations of smell identification by two communities. One community is composed of postgraduate students involved in various investigations related to code smells. The other community is composed of practitioners from industry, with 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 only the community of practitioners strongly associates this task with *semantic smells*. This finding suggests research directions on code smells may be revisited, as they focus mostly on *measurable* or *structural* smells. Considering the novelty of using the social representations theory in software engineering, we also compiled a set of lessons learned. For instance, we observed some key challenges we faced in using the theory. These challenges include: (i) the predominance of associations with technical rather than non-technical concepts, and (ii) the fuzzy definitions of key concepts in our field. Conclusion: We found initial evidence that social representations analysis is a useful instrument to reveal discrepancies and commonalities on how different communities deal with a subjective task. Thus, we expect the experience reported in this paper may encourage and contribute to future studies of social representations in the field.**

***Index Terms*—social representation; code smells; qualitative research**

I. Introduction

Software engineering is composed of tasks influenced by human aspects [1]. Many of these aspects are of collective nature. A typical example is the common sense shared in a particular team or community of developers or researchers. However, collective aspects are typically not considered along the initial conception and design of technologies supporting software engineering tasks. Code smell identification is recognized as one of the most subjective tasks in software engineering [2]. Therefore, this task may be prone to the influence of collective aspects, which are neglected by the research in this field.

Code smells are often defined as symptoms of poor design observed in the low-level structure of a software program [3], [4]. Over the last two decades, the side effects of code smells have been widely studied. Moreover, several techniques and tools have been proposed to support smell identification. Many of the solutions are initially conceived in academic circles. They are not necessarily aligned with the common sense of industry settings.

Recent studies have revealed that certain human aspects, associated with the individuals performing the task, play a significant role in smell identification [5]–[7]. However, the opinion surveys and experiments applied in such studies are not sufficient to capture the collective aspects arising from researchers' and practitioners' communities. Collective aspects include beliefs, values, and behaviors shared among the members of particular communities. One way to investigate these aspects is by using the social representations theory.

The theory of social representations aims at characterizing how a particular community elaborates a particular social object [8], [9]. A *social object* can be any object socialized among individuals, such as the task of smell identification. The social representations emerge from the common sense, which composes the system of beliefs, values, and behaviors in a particular community. Therefore, social representations influence how individuals will behave and communicate [10].

Let us consider a team of developers sharing the following *belief*: 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 *behaviors* 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 conflicts exemplify how identifying potential gaps among the social representa-

tions can be useful. Not revealing these gaps may hamper the transference of technology from academic to industry circles, a prevailing challenge in software engineering.

In this context, this paper reports a preliminary study on the use of social representations theory. Given the motivation above, we apply this theory to the task of identifying code smells. We recruited individuals from two different communities. One community is composed of postgraduate researchers in software engineering. Their research focuses on either code smell identification or topics closely related to code smells. The other community is composed of experienced practitioners from companies where code reviews are part of their routine.

For identifying the social representations of each community, we asked each subject to quote what immediately comes to his mind when thinking about this task. After analyzing the associations, we found considerable differences between the communities. For instance, only the community of practitioners strongly associates smell identification with semantic smells and smell removal. Both associations seem to be misaligned with recurring characteristics of smell detection tools [11]. On the other hand, research-driven students strongly associates the task with measurable smells, typically detected with metrics and thresholds.

To the best of our knowledge, these gaps were not revealed by previous studies of smell identification, in particular those based on mere opinions of practitioners. The knowledge emerging from this study can be used to rethink the emphasis or design of smell identification technologies. Also, we did not find previous work exploring the social representations theory in the entire field of software engineering. In this sense, our study enabled us to identify key challenges for effectively applying the theory of social representations. We discuss alternatives to overcome these challenges based on the lessons we learned. We expect the experience reported in this paper stimulate and provide some guidance for the conduction of future investigations of social representations in the field.

The remainder of this paper is organized as follows. Section II presents the Theory of Social Representations. Section III describes our study settings. Section IV characterizes each community investigated. Section V presents social representation analysis. Section VI discusses threats to validity. Section VII discusses the lessons learned with the study. Section VIII present an overview of the related work. Finally, Section IX concludes the paper and suggests future work.

II. The Theory of Social Representations

Different types of primary studies have been adopted to support research in software engineering. These methods are applied in diverse fields of social sciences and natural sciences. They include experiments [12], opinion surveys [13], case studies [14], and action research [15]. However, none of the methods aforementioned provide resources *per se* to support the investigation of common sense arose from communities, which is proposed by the social representations theory [10].

Introduced by Serge Moscovici, the term *social representations* means the collective elaboration of a social object by a particular community. A social object corresponds to an object socialized by two or more members of a community. It can be, e.g., a software engineering task, such as the code smell identification. Social representation emerges from common sense, comprising 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 individuals, establishing a code for social exchanging and a code for naming and classifying the different aspects of their world [10].

With this theory, Moscovici distinguishes the common sense elaborated by a public layer from the scientific rules and procedures that compose the scientific knowledge [10]. The theory of social representations has been used to support research in fields such as health [8] and education [16], [17]. Howarth et al. [9] argues that social representations are 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 can also be used for investigating the differences among communities. Such comparison may be used for supporting bridging gaps among them.

The study of social representations in this paper is grounded in analyzing the free associations [18] regarding the identification of code smells. Free association is a technique used in psychoanalysis for emerging spontaneous and uncensored content from an individual. In free association tasks, the individual is asked to quote what first comes to mind when he/she thinks about the social object investigated. The question should be immediately answered, and the quotes provided should be noted the same way they were uttered, i.e., the order they came to mind. The set of quotes collected in the free association task composes the associations made by the subjects. Several associations with different frequencies may emerge from the free association task. In this sense, it is possible to identify and group quotes 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 [13]. 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.

III. Study Settings

Section III-A presents our goal and our research question. Section III-B describes the study population and sample. Section III-C overviews our study instrumentation. Finally, Section III-D describes the data analysis procedures.

A. 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): *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 [11]. On the other hand, it is expected that those tools should be 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. This is a preliminary study, which is part of a broader investigation of human aspects affecting the identification of code smells [5], [6], [19]. Some findings from this broader investigation are used in this paper for characterizing the communities investigated.

B. Population and Sample

The community of students is composed of Master and Doctoral students from the Informatics Department at a Brazilian University with a strong international reputation according to several international rankings. We sampled these students from the Empirical Software Engineering course offered by the department. All 17 subjects from this sample investigate software engineering topics, often closely related to code smells (e.g., code refactoring). Moreover, seven of these students actively work on code smell research. The community of practitioners is composed of developers actively working in the Brazilian industry. We obtained a sample of 13 practitioners from three software companies selected by convenience. These companies deliver software solutions to the Brazilian government, as well as, private clients. In these companies, code reviews are part of the routine.

C. 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 [18] 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 III-A).

Table I 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 I
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.

D. 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 [18]. 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 1 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.

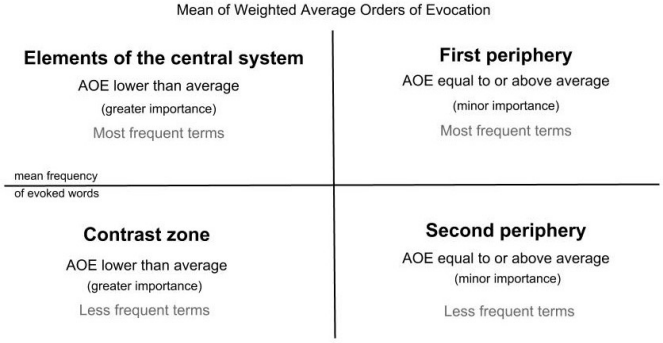

Fig. 1. Structure of the four-quadrant analysis

IV. Characterization of the Communities

We aim at obtaining the social representations of the identification of code smells by two different communities: postgraduate students and practitioners. Thus, it is important to identify 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 I). The results of the free association task are used in Section V to perform the social representations analysis.

Code smell definitions. The communities investigated are composed of individuals sharing a similar distribution of higher academic degree. Most of them hold bachelor's degrees or similar, whereas the rest hold master's degrees. However, as expected, practitioners are more experienced in reviewing code than students. On the other hand, seven from the 17 students work on investigations related to code smells. In both communities, most of the subjects declared familiarity with the term *code smell*. They provided definitions of the term compliant with different definitions available in the literature (e.g., [3], [4]). Even subjects not familiar with code smells provided acceptable definitions. For instance, one student reported that code smells are *(...) characteristics identified in the source code that may either represent or generate a maintenance problem in the future*, whereas another student defined code smell as *(...) a symptom of a design problem in the source code*. Among practitioners, one defined code smells as *(...) an implementation problem which hampers the comprehension and maintenance of certain code snippets*, whereas another practitioner defined code smells as *(...) some extraneous code snippet that does not follow design principles, patterns or idioms that should be used*.

Opinions on smells and their identification. The opinion of the subjects regarding smell identification is predominantly positive. In other words, most of the respondents from both communities partially or totally agree that identifying code smells promotes program quality and better program comprehensibility. We also asked the subjects which types of code smell 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 [20], [21]. We classified the code smells concepts according to the types of properties on which they apply – measurable, structural or semantic – adapting the classification proposed in [22]. *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.

The most frequent smell types mentioned by students (first quartile) in a non-free-association question are measurable smells: Long Method (14), God Class (14), Feature Envy (9) and Refused Bequest (6). Only these four code smell types represent more than 50% of all their mentions. On the other hand, the knowledge on the identification of structural smells was less frequent. Regarding semantic smells, only the misuse of words in *comments* was quoted. Similarly, the most frequent smells quoted by the practitioners were Feature Envy (6), God Class (6), and Long Method (4). As the students, practitioners also reported less frequent knowledge about the identification of structural smells than the measurable ones. Besides, at this point, no semantic smell was mentioned in this non-free-association question. The same will not hold for the free association question, discussed in the next section.

The results reported in this section indicate that both communities are aware of code smells, sharing positive opinions regarding the benefits of smell identification. However, their formal knowledge about smell types is concentrated in measurable ones. Despite these similarities, we emphasize that each community has its collective background, which influences the social representations discussed in the next section.

V. Social Representations of the Identification of Code Smells

This section describes and discusses the analysis of the social representations of smell identification by the communities of postgraduate students and practitioners, both characterized in Section IV. For this purpose, the study subjects performed

a free association task (see Section III-C). Three researchers performed separated analyses of the terms evoked by the members of each community, aiming at identifying and clustering similar terms into a single association. For this purpose, the researchers eventually used the respondents' explanation regarding the more relevant association and other answers in the questionnaire (see Section III-C). This was important to check the actual meaning of the terms quoted. For instance, the *review* and *inspection* quotes were grouped into the *inspection* association. The researchers also grouped the different types of code smells quoted based in types of properties on which they apply (see Section IV). In cases of doubt regarding its meaning, the association was not grouped. After concluding the grouping activity, one researcher composed the final set of associations and performed the four-quadrant analyses , following the methodology presented in Section III-D.

A. Students

Most of the 17 students made five quotes, resulting in 75 associations, including repetitions. After performing analyses, we consolidated a set of 31 distinct associations. From these, 17 associations were quoted only once. Therefore, they were excluded from further analysis. Table II presents the four-quadrant analysis, distributing the associations- translated from the Portuguese- by frequency and average order of evocation (AOE). In this community, the associations with the frequency equal or higher than 4 and with AOE lower than 2.81 composed the central system.

TABLE II
FOUR-QUADRANT ANALYSIS OF THE SOCIAL REPRESENTATIONS OF SMELL IDENTIFICATION BY STUDENTS

Central system			**First periphery**		
Frequency $\geq$ 4	**AOE $<$ 2.81**		**Frequency $\geq$ 4**	**AOE $\geq$ 2.81**	
	Frequency	**AOE**		**Frequency**	**AOE**
Measurable smells	8	2.25	Removing smells	6	3.17
Inspection	4	1.50	Source code	4	3.75
Detection	4	2.25	Design	4	4.25
			Structural Smells	4	3.50
Contrast zone			**Second periphery**		
Frequency $<$ 4	**AOE $<$ 2.81**		**Frequency $<$ 4**	**AOE $\geq$ 2.81**	
	Frequency	**AOE**		**Frequency**	**AOE**
Design Problem	3	2.00	Problem	3	3.00
Bug	3	1.67	Quality	3	4.00
Anomaly	3	2.67	Maintenance	2	3.00
Strategy	2	1.50			

Table II reveals some interesting findings. By analyzing the central system (top-left quadrant), we observe that students promptly associate the smell identification with *measurable smells*. The detection of these smells is supported by a combination of metrics and their thresholds. We also observed that the core system of the social representations is composed of *detection*. This term comprises formal rules and automated support to detect different smell types. These results indicate the emphasis of research-driven students in automated solutions for smell identification. The detection of measurable smells is largely supported by existing solutions despite prevailing issues with precision and recall. Nevertheless, some students also promptly associate the smell identification with *inspection*, recognizing the need for further manual analysis once smell suspects are automatically detected.

The first periphery (top-right quadrant) indicates that *removing smells* is a relatively frequent concern although not promptly associated (AOE $\geq$ 2.81). In other words, post-graduate students seem to think of smell identification and refactoring as non-intermingled tasks, which is not the case for practitioners (Section V-B). This quadrant also suggests a balanced concern of the community between the *source code* and its underlying *design*. Besides, we emphasize the incidence of *structural smells*, contrasting with the presence of measurable smells in the central system. Surprisingly, we could not observe among students a strong association between smell identification and the incidence of design problems although a considerable part of recent research in the field associates both concepts.

B. Practitioners

Most of the 13 practitioners quoted five terms, resulting in 49 associations, including repetitions. After analyses, we consolidated a final set composed of 17 distinct associations. From these, four associations were quoted only once. Therefore, they were excluded from the analysis of social representations. Table III presents the analysis of the four quadrants for the community of practitioners. Associations with the frequency equal or higher than 3 and with AOE lower than 2.38 compose the central system of the social representations. We can observe that the associations of the practitioners are concentrated in quadrants located in the extremes, i.e., the central system and the second peripheral system. It indicates a clearer definition of the behaviors, beliefs and values shared by this community.

TABLE III
FOUR-QUADRANT ANALYSIS OF THE SOCIAL REPRESENTATIONS OF SMELL IDENTIFICATION BY PRACTITIONERS

Central system			**First periphery**		
Frequency$\geq$3	**AOE $<$2.78**		**Frequency$\geq$3**	**AOE$\geq$2.78**	
	Frequency	**AOE**		**Frequency**	**AOE**
Inspection	6	1.83	Bugs	3	3.00
Removing smells	4	2.00			
Structural smells	4	2.00			
Design problem	3	1.67			
Semantic smells	3	2.00			
Contrast zone			**Second periphery**		
Frequency$<$3	**AOE$<$2.78**		**Frequency$<$3**	**AOE$\geq$2.78**	
	Frequency	**AOE**		**Frequency**	**AOE**
Detection	2	1.50	Attention	2	2.50
			Maintenance	2	2.50
			Knowledge	2	2.50
			Patience	2	3.00
			Bad performance	2	3.50
			Code complexity	2	4.50

Practitioners frequently and promptly associate the identification of code smells with *structural smells*, i.e., code smells which detection is grounded in the static analysis of constraints. Another strong association is the *semantic smells* (Section IV). Although semantic smells are not frequently catalogued or supported by detection tools [11], the identification of semantic problems in the source code is typically supported by manual code inspections. The community of practitioners also strongly associates the *removal of code smells* and *design problems* with smell identification. It suggests the concern of this community on fixing smells indicating relevant problems in the software structure.The incidence of *bugs* is another

association performed by practitioners. However, this association is not promptly made as those presented in the central system. *Detection* figures out in the contrast zone (bottom-left). In this quadrant, the association is promptly made, but the frequency is low. Although less relevant, the secondary peripheral system reveals the awareness of the practitioners with behaviors (*attention*, *patience*) and the proper *knowledge* needed to perform the task.

C. Discussion

The results presented in Section IV indicate that both communities share similar knowledge or opinions on code smells, their identification, and their impact on software quality. On the other hand, the social representations analysis reveals more differences than similarities in the common sense shared by each community. In a big picture, both communities strongly associate the smell identification with code inspection. This finding indicates their recognition that manual analysis is needed in smell identification tasks. However, the other core elements of each community are considerably different.

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 [11]. This association is compatible with the formal knowledge of code smells (see Section IV). On the other hand, the community of practitioners show 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 IV). Interestingly, we note that the current catalogues and tools available for supporting the identification of code smells barely address semantic ones [11]. Finally, only practitioners strongly associate smell identification with the immediate concern on removing them. 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.

Therefore, the differences observed in 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 on guiding the code reviewers 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 an impact analysis of the different decisions that could be taken.

VI. Threats to Validity

We recruited postgraduate students to represent the research side of our study, 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.

Although the small sample sizes of each community, it is important to note that they are composed of individuals with considerable diversity of background and experience. Despite the surveyed students are from a single university, they collaborate with other research groups investigating different topics, including requirements, security, bugs, code smells, design problems, and refactoring. Regarding the practitioners, it is important to mention they come from three independent software companies, playing different roles in software projects from considerably different domains.

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 for 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 (see Section III-C). Besides, our option by questionnaire was also motivated by time constraints in the industrial settings.

VII. Lessons Learned

The findings of this preliminary study suggest the usefulness of investigating the social representations of software engineering tasks. However, applying the social representations theory for the first time was far from trivial. We faced particular challenges in planning the study, executing it, and analysing results. Most of these challenges are due to the inherent characteristics of the Software Engineering field. Aiming at supporting future studies of social representations in software engineering, we compiled the lessons learned as follows.

Predominance of technical associations. Considering the previous studies of social representations in fields such as education and health [8], [16], [17], we expected to collect a set of associations predominantly composed of non-technical concepts, such as feelings and adjectives. However, the associations quoted in our study were predominantly composed of technical concepts. As presented in Section V, the core elements of the communities investigated include only technical concepts and activities. Some non-technical concepts were quoted, such as *need, challenge, patience, attention* and *amateurism*. However, most of them were quoted only once. This scenario could be partially explained by the technical bias of the Software Engineering field but also by the inexperience in performing free association in software engineering. In this

sense, conducting warming-up group activities for practising free association regarding other topics would help.

Fuzzy and diverging terminology in the field. Even with the already mentioned predominance of technical concepts, arguably "well-established" or "popular" ones in our field, understanding such concepts used by participants was really difficult. The reason is that our field often has fuzzy terminology or diverging definitions for many concepts. Moreover, sometimes there is a uniform academic definition for a certain term, but professionals overload this term with many organization-specific definitions. As a consequence, the analysis of the associations in our study was challenging. Even though there are international standards for specific subjects in software engineering, they frequently do not comply with the technical literature. A classic example is the definition of a "bug". It is a buzzword used in software engineering research and practice with several meanings. [23]–[25]. Therefore, it is often not trivial to identify the actual meaning of an association. Besides, we observed that different terms were used to make the same association. Therefore, we opted by analyzing the actual meaning of each term and the possibility of grouping them in a single association (see Section V). We used the answers given to the other questionnaire items in order to catch the actual meaning of the subjects' associations. For instance, we found the opportunity of grouping the different associations of *review* and *inspection* into a single association (inspection).

High diversity of the associations. The 162 associations evoked in this preliminary study was sufficient to apply the four-quadrant analysis (see Section V). However, we found relatively low frequencies even in the core elements. None of the associations depicted was evoked by more than half of the subjects from a community. Moreover, the mean frequencies calculated were considerably low. Although it may be partially explained by the terminology issues discussed above, we still observed a considerable diversity of the associations. This diversity was observed even after the careful analysis performed for grouping terms with similar meanings. Therefore, this scenario may demand the analysis of larger samples, which should be balanced with the control required for conducting free association tasks.

The importance of listening "non-qualified" subjects. Code smell is a concept often taught nowadays in undergraduate courses. However, the identification of code smells may not be a practice adopted in certain software projects. Therefore, some individuals may have no actual experience in identifying code smells through code reviews. In some cases, they may even have a misconception of code smells, which we could not observe in our study. In the context of surveys, case studies and experiments, one may suggest excluding such individuals, arguing they are "non-qualified", using a much more rigorous criterion for "qualifying" subjects. However, the participation of any member from a particular community is potentially useful in the social representations analysis. The theory of social representations is grounded in common sense, influenced by all members of the community. In this way, social representations is also a useful tool to detect preconceptions and misconceptions influencing one or more communities.

The importance of control in data collection. Data collection to support social representations analysis is based on free association. This method requires providing prompt and objective answers to the stimulating questions [18]. In other words, the free association task should not be influenced by any external factor or even by any "time to think". Thus, conducting free association tasks through individual interviews seems to be the best choice. However, conducting interviews in a large-scale setting may be costly. In our study, we adopted an alternative method. We combined the convenience of applying questionnaires with the opportunity of supervising small samples in controlled environments. One disadvantage of this approach is the impossibility of assessing the associations made by the subjects *on the fly*, which can be easily performed through interviews. Therefore, in spite of the useful results derived from our study, our experience reinforces the preferential use of interviews for free association tasks.

VIII. Related Work

In our study, the subjects performed a free association task for analyzing the social representations of smell identification. Although we could not find a previous work investigating the social representations of a particular software engineering activity, a free association task was previously adopted in a survey on software agility [26]. The task had supported the analysis of a large sample of developers, identifying opportunities for grouping them according to the similarities found among their associations.

In this study, we claim the novelty of investigating the collective aspects of smell identification. However, subjectivity and certain individual aspects of smell identification have been investigated in previous work. These investigations are predominantly grounded in controlled experiments and surveys. A recent study [2] investigated the level of agreement of several software developers regarding the incidence of different code smells. In most cases, the level of agreement found was significantly low. Moreover, different arguments were provided to take the same decision. The authors concluded that developers tend to have different perceptions about the smell incidence.

Yamashita and Moonen [7] conducted an exploratory survey with developers from different countries regarding code smells. They found different interpretations of code smell and different perceptions about the impact of code smells on software design and its overall quality. Mäntylä et al. [27] 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. [4] investigated the developers' perception of poor design and implementation choices by developers. The researchers con-

cluded that the developers' experience and previous knowledge of the modules to be reviewed influences their perceptions.

We recently [5], [6] conducted empirical studies to observe the influence of three human aspects in the precision of smells correctly identified by developers: the professional background, the module knowledge, and the use of single or pair reviews. The studies were conducted in industrial settings, having software professionals as subjects and real software projects as objects of study. We found evidence that these factors (and certain combinations) clearly influence the precision of smell identification tasks. Therefore, one can see that the different works discussed above have been investigating individual aspects of smell identification rather than collective ones.

IX. Conclusion and Future Work

The exploration of the social representations for smell identification enabled us to reveal the communities' commonalities and differences, as discussed in Section V. These observations can be used to potentially rethink the emphasis or design of smell identification technologies. Most of the existing automated techniques [11] do not: (i) support a wide range of semantic smells (which are much harder to detect), and (ii) offer a unified framework for smell detection and refactoring as part of a continuous integration process. For example, existing academic techniques do not anticipate or detect if recommended refactorings, while targeting the removal of a certain smell, introduce other (even more critical) smells.

Moreover, to the best of our knowledge, the use of the social representations theory for investigating a software engineering task is a novelty. Our experience obtaining and analyzing the social representations of smell identification resulted in deriving important lessons. These lessons consist of warnings or recommendations for conducting studies exploring social representations. Different from the observations in other fields, where the theory has been applied, our study revealed that the social representations comprised much more technical than non-technical concepts.

Moreover, the common fuzziness of terminology in Software Engineering demands more rigorous instrumentation and analysis to support our understanding of the associations made by the subjects. We also observed along the study design and execution the need for rigorous control of the free association task. Based on these lessons, we intend to evolve our study design and execution for future replications. We also plan to use the experience gathered in this study to investigate the social representations of other software engineering tasks highly influenced by human aspects.

Acknowledgment

This work is funded by CAPES/Procad grant #175956 and CNPq grant #153363/2018-5.

References

[1] P. Lenberg, R. Feldt, and L. G. Wallgren, "Behavioral software engineering: A definition and systematic literature review," *J. Syst. Softw. (JSS)*, vol. 107, pp. 15–37, 2015.

[2] M. Hozano, A. Garcia, B. Fonseca, and E. Costa, "Are you smelling it? investigating how similar developers detect code smells," *Inform. Softw. Tech. (IST)*, vol. 93, pp. 130–146, 2018.

[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] 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.

[6] 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.

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

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

[9] C. Howarth, "A social representation is not a quiet thing: Exploring the critical potential of social representations theory," *Br. J. Soc. Psychol. (BJSP)*, vol. 45, no. 1, pp. 65–86, 2006.

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

[11] 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.

[12] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[13] 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.

[14] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Emp. Softw. Eng. (ESE)*, vol. 14, no. 2, p. 131, 2009.

[15] P. S. M. dos Santos and G. H. Travassos, "Action research use in software engineering: An initial survey," in *3rd ESEM*, 2009, pp. 414–417.

[16] G. E. Moreira and A. L. Manrique, "Challenges in inclusive mathematics education: Representations by professionals who teach mathematics to students with disabilities," *Creative Education*, vol. 5, no. 7, pp. 470–483, 2014.

[17] 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.

[18] 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.

[19] 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.

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

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

[22] N. Moha, Y.-G. Gueheneuc, L. Duchien, and A.-F. Le Meur, "Decor: A method for the specification and detection of code and design smells," *IEEE Trans. Softw. Eng. (TSE)*, vol. 36, no. 1, pp. 20–36, 2010.

[23] S. Zaman, B. Adams, and A. E. Hassan, "Security versus performance bugs: a case study on firefox," in *8th MSR*, 2011, pp. 93–102.

[24] M. Ohira, Y. Kashiwa, Y. Yamatani, H. Yoshiyuki, Y. Maeda, N. Limsettho, K. Fujino, H. Hata, A. Ihara, and K. Matsumoto, "A dataset of high impact bugs: Manually-classified issue reports," in *12th MSR*, 2015, pp. 518–521.

[25] A. Nistor, T. Jiang, and L. Tan, "Discovering, reporting, and fixing performance bugs," in *10th MSR*, 2013, pp. 237–246.

[26] R. M. de Mello, P. C. da Silva, and G. H. Travassos, "Sampling improvement in software engineering surveys," in *8th ESEM*, 2014, p. 13.

[27] M. V. Mäntylä, J. Vanhanen, and C. Lassenius, "Bad smells-humans as code critics," in *20th ICSM*, 2004, pp. 399–408.