Do you see any problem? On the Developers' Perceptions in Test Smells Detection

Rodrigo Lima
Federal University of Pernambuco
Recife, Brazil
rsl@cin.ufpe.br

Keila Costa
Federal University of Pernambuco
Recife, Brazil
kbcs2@cin.ufpe.br

Jairo Souza
Federal University of Pernambuco
Recife, Brazil
jrmcs@cin.ufpe.br

Leopoldo Teixeira
Federal University of Pernambuco
Recife, Brazil
lmt@cin.ufpe.br

Baldoino Fonseca
Federal University of Alagoas
Maceió, Brazil
baldoino@ic.ufal.br

Marcelo D'Amorim
Federal University of Pernambuco
Recife, Brazil
damorim@cin.ufpe.br

Márcio Ribeiro
Federal University of Alagoas
Maceió, Brazil
marcio@ic.ufal.br

Breno Miranda
Federal University of Pernambuco
Recife, Brazil
bafm@cin.ufpe.br

ABSTRACT

Developers are continuously implementing changes to meet demands coming from users. In the context of test-driven development, before any new code is added, a test case should be written to make sure new changes do not introduce bugs. During this process, developers and testers might adopt bad design choices, which may lead to the introduction of the so-called Test Smells in the code. Test Smells are bad solutions for implementing or designing test code. We perform a broader study to investigate the participants' perceptions about the presence of Test Smells. We analyze whether certain factors related to the participant' profiles concerning background and experience may influence their perception of Test Smells. Also, we analyze if the heuristics adopted by developers influence their perceptions about the existence of Test Smells. We analyze commits of open source projects to identify the introduction of Test Smells. Then, we conduct an empirical study with 25 participants that evaluate instances of 10 different smell types. For each Test Smell type, we analyze the agreement among participants, and we assess the influence of different factors on the participants' evaluations. Altogether, more than 1250 evaluations were made. The results indicate that participants present a low agreement on detecting all 10 Test Smells types analyzed in our study. The results also suggest that factors related to background and experience do not have a consistent effect on the agreement among the participants. On the other hand, the results indicate that the agreement is consistently influenced by specific heuristics employed by participants. Our findings reveal that the participants detect Test Smells in significantly different ways. As a consequence, these findings introduce some questions concerning the results of previous studies that do not consider the different perceptions of participants on detecting Test Smells.

KEYWORDS

Test Smells, Open Source, Human Factors, Empirical Study

ACM Reference Format:
Rodrigo Lima, Keila Costa, Jairo Souza, Leopoldo Teixeira, Baldoino Fonseca, Marcelo D'Amorim, Márcio Ribeiro, and Breno Miranda. 2023. Do you see any problem? On the Developers' Perceptions in Test Smells Detection. In *XXII Brazilian Symposium on Software Quality (SBQS '23), Nov 7–10, 2023, Brasília, Brazil*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3629479.3629485

SBQS '23, Nov 7–10, 2023, Brasília, Brazil

ACM ISBN 979-8-4007-0786-5/23/11…$15.00
https://doi.org/10.1145/3629479.3629485

1 INTRODUCTION

Test Smells indicate bad programming practices, when developers organize or implement test cases, which might indicate potential design problems in the test code [23]. Therefore, Test Smells detection is an elementary technique for supporting a wide range of quality improvement tasks, such as writing new test cases, avoid bugs, and ensure that the code is working as expected [12]. However, detecting Test Smells in practice is much harder than usually assumed or advertised [16]. Because, developers may diverge on the presence of Test Smells.

Developers might have divergent perceptions about the presence of the same Test Smells. In particular, the informal and subjective definition of certain Test Smell types [23] may lead professionals to reason about each Test Smell occurrence differently [23]. In spite of the extensive tool support for Test Smell detection available nowadays (e.g. [2, 5, 25]), developers still need to individually analyze each Test Smell and confirm its occurrence on the system. While a developer may confirm a test snippet as the host of a particular Test Smell, other developers may have different perceptions. For instance, an Assertion Roulette occurs when a test method has multiple non-documented assertions; this may lead developers not to agree on how many assertions should be considered a Test Smell.

The detection of different types of Test Smells follows a similar rationale. It is generally believed that high consensus among participants is advantageous for a variety of reasons. First, it promotes consistency in code reviews. However, achieving these benefits becomes more difficult when disagreement among professionals is more common than agreement. Currently, little is understood about how professionals detect Test Smells in a similar manner. It is important to explore whether certain factors affect the similarity or dissimilarity of professionals' perceptions regarding Test Smells occurrences. Factors such as professionals' characteristics (e.g. similar or diverse backgrounds and experiences) may play a role in the Test Smells detection process. Moreover, personal factors, such as each participants's individual heuristic for detecting Test Smells, may also influence their detection process.

In this context, this paper presents a study aiming at investigating the (dis)agreement among participants on detecting occurrences of 10 test smell types. The study also analyzes whether certain factors may influence such (dis)agreement. The study involves 25 participants who evaluate the presence of Test Smells into a set of test snippets from real projects, collected with a manual analysis commits from 32 java projects and validated by 2 researchers. Altogether, more than 1250 evaluations were collected and analyzed. We analyze the influence of factors on the participants' agreement. Such factors are mainly related to participants' background and experience. Also, we investigate how similar are the heuristics formulated by different participants to detect Test Smells.

We discovered three conclusions in our study. Firstly, participants exhibited low levels of agreement when evaluating all 10 Test Smells types, which contradicts key findings from prior studies [8, 17]. Secondly, participants' background and experience did not consistently affect their agreement levels. Lastly, the heuristic factor played the most significant role in determining agreement among participants. These findings highlight the importance of developing customizable techniques for detecting Test Smells that take into account each participants unique perception.

The remaining of this document is structured as follows. Section 2 describes the design of our empirical study and the research questions. In Section 3 we present the results of the study and, Section 4 details the threats of the study. Next, Section 5 presents the related work. Finally, Section 6 presents the conclusions observed in our study.

2 STUDY DESIGN

This study aims at investigating the participants' agreement on detecting Test Smells in test snippets from open-source projects. We analyze whether participants tend to agree (or disagree) on the occurrence of Test Smells in a wide range of types. In particular, we investigate to what extent certain factors may influence common perceptions shared by different participants. Participants evaluated several instances of 10 smell types identified in real systems. As a result, we analyze the agreement among participants according to their evaluations. Also, we investigate if participants with common characteristics identified Test Smells similarly. In this way, two main research questions guide our study:

- **RQ**$_1$: *Do participants agree on the "smelliness" of test code?*

  This question investigates if participants agree on detecting Test Smells in real projects. In addition, we analyze the degree of such agreement to verify how differently the participants detect Test Smells in the same test snippet. Particularly, such analysis becomes difficult because it requires the participation of several participants with different characteristics to create a relevant sample. Such requirements may have reduced the conclusions of previous studies that investigated similar questions [10, 11]. Thus, we investigate deeper to increase the knowledge about how similar participants detect Test Smells in open-source test snippets. Furthermore, our results may shed light on constructing more efficient detection techniques.
- **RQ**$_2$: *What makes participants agree on the "smelliness" of test code?*

  In this question, we analyze if certain factors may influence participants' (dis)agreement on smell detection. In particular, we analyze whether participants detect Test Smells similarly when grouped according to common characteristics (participants' background and experience). Also, we investigate how participants' (dis)similar judgment is influenced by the heuristics that the participants formulate for test smell detection. The results may reveal the potential and limitations of customized detection techniques for smell detection.

2.1 Context Selection

In our study, we involve participants with different backgrounds and experiences to investigate how differently the participants detect smells in test snippets. We use the Prolific Platform [15][1] to recruit participants. Initially, we send a screening focused on Unit Testing to the participants. Then, we select the participants with minimum testing knowledge and who have already had some contact with Test Smell detection to participate in the study. We consider these requirements to avoid participants who did not have previous knowledge about testing or Test Smell detection aiming to avoid random answers given the lack of this knowledge. More details about the participants' profiles (in terms of their background and experience) are described in the support material.[2]

Upon selecting the study participants, we began our data collection by searching for Test Smells in commits from 32 Java projects on GitHub. We used test smell names as keywords in our search and specifically filtered issues that contained pull requests or commits associated with such keywords. From an initial set of 1250 commits identified, two authors, consisting of one Ph.D. researcher and one Ph.D. student researcher, manually analyzed each commit to extract test smells, resulting in 71 test smell snippets. Among these, only 10 types of test smells had at least five snippets each, which gave us the 50 test snippets that we used in our study. Subsequently, a separate set of two researchers, also a Ph.D. researcher and a Ph.D. student researcher, validated these commits. Our selection criteria for commits specifically included those that introduced modifications or insertions to a maximum of 2 test cases.

[1] https://www.prolific.co
[2] https://github.com/tests-smells/agreement

The participants evaluate then test snippets of 10 test smell types, as described in Table 1. We analyze these types because they are the ones that most frequently appear in the projects analyzed in our study.

For each smell type, the participants evaluated 10 test snippets with different characteristics. Our goal is to expose participants to snippets suspiciously containing the smell type under analysis. To select the test snippets analyzed in our empirical study, we analyze snippets from 32 open-source Java projects. To validate the presence of smells, we consider the smells definitions described in previous work [14, 23]. We select such projects because existing smell detection techniques have been used them. Additionally, the test code of these projects contains a variety of suspicious Test Smells that enable the execution of our empirical study. The use of these different projects is aimed at collecting potential suspicious smell instances that are detected in different ways and that, possibly, are defined by the different perceptions of their creators.

2.2 Operation

We design our study to collect the opinion of participants about the existence of Test Smells in certain test snippets. We collect the participants' perceptions from a series of evaluations obtained through the Internet-based application Prolific. Firstly, for each participant involved in the study, we register the Prolific ID, background (*Developers* or *Data Scientists*), and their experience with Test Smells. We randomly divide the participants into groups containing five participants. Each group of participants evaluates 10 test snippets, one for each smell type analyzed.

After defining the groups of developers, we send an individual invite to each participant in Prolific Platform. Once the participant accepts the invitation, we send a Google form containing 10 test snippets to be evaluated, one for each smell type. For each test snippet evaluated, we present a closed question to the developer: Would you observe any issue in this test code (as to refactor it)? and an open question: Justify your previous answer, where the participants justified their answer.

2.3 Open Questions

As described in the previous section, the participant must answer an open question reporting his understanding of the Test Smell type analyzed. First, after the participant evaluates each snippet, we expect he can report the heuristic used to detect the Test Smell during his analysis. In this way, the open question is also used as a control point to identify if the participant is negligent in his evaluations.

The answers to the open questions play an important role in the study since they help us to analyze how similar the participants detect Test Smells. Although we present a single definition of each smell type during the evaluation phases, we assume that the participants detect the same smell differently. Thus, from these answers, we apply a *coding* procedure [19] to collect the heuristics used by the participants to detect the Test Smells. Thereafter, we use such heuristics to analyze if the participants who follow a similar heuristic present a better agreement.

2.4 Data Analysis

To answer the research questions described in Section 2, we use a measure that computes the inter-rater agreement among the participants' evaluations. Such agreement is calculated using *Fleiss*' Kappa, a measure to evaluate the agreement among multiple raters [6]. This measure reports a number lower or equal to *1*. If the *Kappa* value is equal to *1*, then the raters are in complete agreement. If there is no agreement among the raters, then the Kappa value is lower than *0*. In addition, we consider the *Kappa* categories proposed by Landis and Koch [9], as described in Table 4. Such categories have been used by previous work related to smell detection [7] to verify the *strength* of the *Kappa* value.

We answer the research questions defined in our study by considering the use of the Kappa measure and the classification proposed by Landis and Koch. Aiming at answering $\mathbf{RQ}_1$, we perform the following procedure for each smell type:

(1) We collect the evaluations done by 25 participants. Each participant evaluates 10 test snippets, one for each smell type analyzed in our study. As a result, we produce a 10×25 matrix containing YES or NO answers according to each evaluation;
(2) From the evaluation matrix, we compute the *Fleiss' Kappa* measure [6] to obtain the degree of agreement among the 25 participants that evaluated the test snippets of the same smell type. We then classify the *strength* of the assessed inter-rater agreement according to the classification described in Table 4.

After obtaining the Kappa measure and its classification, we answer $\mathbf{RQ}_1$ by analyzing the *strength* of the agreement. To answer $\mathbf{RQ}_2$, we investigate which factors can influence the agreement among participants. We analyze the agreement among the evaluations of a subgroup of participants defined from each factor, as described in Table 5. The procedure to create these subgroups is described in Section 3.

2.5 Execution and Data Preparation

The execution of the empirical study occurred over three months, beginning on January/2022. In total, we invite 92 participants, but only 25 of them completed all the study phases. After collecting the data produced from the study execution, we perform a series of procedures to ensure the quality of the observed data. In particular, we analyze the participants' evaluations and their answers to the open question *Would you observe any issue in this test code (as to refactor it)?*, aiming at identifying any problem that could prejudice the data analysis.

We obtain answers from 25 participants with different experiences in testing. As mentioned, we select participants from the Prolific platform, and we use screening to filter them according to the testing experience. Their current job includes Developers and Data Scientists.

3 RESULTS

In what follows, we describe the main results of our study. To answer $\mathbf{RQ}_1$, we analyze the agreement levels assessed from the participants' evaluations. In the $\mathbf{RQ}_2$, we investigate the agreement among the participants by considering different factors related to

Table 1: Common Test Smells.

Test Smell	Definition
Assertion Roulette	A collection of unexplained assertions in a single test method that makes it difficult to trace which exact assertion had a problem in the event of test failure.
Eager Test	Occurs when a test method invokes several methods of the production object. This smell results in difficulties in test comprehension and maintenance.
Lazy Test	Occurs when multiple test methods invoke the same method of the production object.
Mystery Guest	Occurs when a test method utilizes external resources (e.g. files, database, etc.). Use of external resources in test methods will result in stability and performance issues. Developers should use mock objects in place of external resources.
Resource Optimism	This smell happens when test methods make optimistic assumptions about the existence or the state of external resources like files and databases.
Sensitive Equality	It is fast and easy to write equality checks using the toString method. A typical way is to compute an actual result, map it to a string, which is then compared to a string literal representing the expected value.
Conditional Test Logic	This Test Smell occurs when the test depends on a condition. It is a dangerous design since a test method may result in a passed status without ever having asserted a unit result.
Duplicate Assert	This smell occurs when a test method tests for the same condition multiple times within the same test method. If the test method needs to test the same condition using different values, a new test method should be utilized; the name of the test method should be an indication of the test being performed.
Sleepy Test	Explicitly causing a thread to sleep can lead to unexpected results, as the processing time for a task can differ on different devices. Developers introduce this smell when they need to pause execution of statements in a test method for a certain duration (i.e. simulate an external event) and then continuing with execution.
Exception Handling	This smell occurs when the test manually handles both exceptions and test outcome, and its frequency rivals with Assertion Roulette on Java projects.

Table 2: Participant Distribution by Role

Role	Number of Participants
Developers	18
Data Scientists	7

Table 3: Participant Distribution by Experience

Experience Level	Number of Participants
Less Experienced (0—1 year)	11
More Experienced (2+ years)	14

Table 4: Landis and Koch classification for Kappa values

Kappa Statistic	Strength of Agreement
< 0.00	Poor
0.00 - 0.20	Slight
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Substantial
0.81 - 1.00	Almost Perfect

them, such as background (Section 3.2), experience (Section 3.3), and the heuristics reported by the participants in the open questions (Section 3.4).

Table 5: Influencing factors investigated in the study

Factor	Description
Developer's background	Indicates if the participant is a developer.
Data Scientist's background	Indicates if the participant is a data scientist.
Experience	Indicates (in a scale from 1 to 10) the participant's experience.
Detection heuristic	Extracted from the open questions.

3.1 Overall Agreement

This section describes the agreement assessed from the participants evaluations over a set of test snippets related to the types of test smells analyzed in our study. A total of 25 participants evaluate 50 test snippets with different smell types and report if the evaluated snippet contains a Test Smell or not. We then use these evaluations to assess the agreement among the participants.

Table 6 reports how the 25 participants (columns) evaluate test snippets. The first column describes the smell type of the test snippet analyzed by the participants. The following columns represent the participant's evaluation. Each cell represents the `YES`/`NO` answer to the question: *Would you observe any issue in this test snippet (as to refactor it)?*. Grey cells represent `YES`, and white cells represent `NO` answers.

We observe that the developers do not present a complete agreement in any of the smell types. Notice also that while developer

#15 agrees with the presence of smells in all the smell types analyzed, developer #21 disagrees in all the cases analyzed. From the evaluations related to each smell type investigated in our study, we calculate the *Kappa* measure to assess the agreement among the participants. Then, we identify the category (or *strength*) of this *Kappa* value by considering the classification described in Table 4.

Figure 1 illustrates the *Kappa* values obtained for each smell type analyzed in our study. It also indicates the *strength* of the obtained values. The *Kappa* values are attached to the bar associated with each smell type, while the *strength* degree is represented by the *gray* columns crosscutting the bars.

We observe that the developers reach their highest agreement levels on evaluating the Resource Optimism, and Exception Handling smells. However, these agreement levels are still low. They range from 0.21 up to 0.28, thus not reaching a *strength* higher than *Fair*. Actually, Resource Optimism and Exception Handling are the only cases that achieve a *Fair strength* level. For the rest of the analyzed smell types, the *strength* of agreement falls in the lowest category (*Slight*). The developers only reach *Slight* agreement, with all of the *Kappa* values being lower than 0.2. Note also that most agreement levels are lower than 0.10.

These results suggest that participants tend to disagree on detecting Test Smells, given the low agreement levels among their evaluations, which do not reach even a *Moderate* strength. Different factors involved in the Test Smells detection may influence such a tendency, such as the participant' background and experience. In addition, the absence of a formal definition for Test Smells, and perhaps the impossibility for some test smells, might contribute to participants detecting the same type of Test Smell in different ways. Consequently, the participants tend to disagree on their evaluations. In the following sections, we describe the agreement among the participants' evaluations by considering different factors.

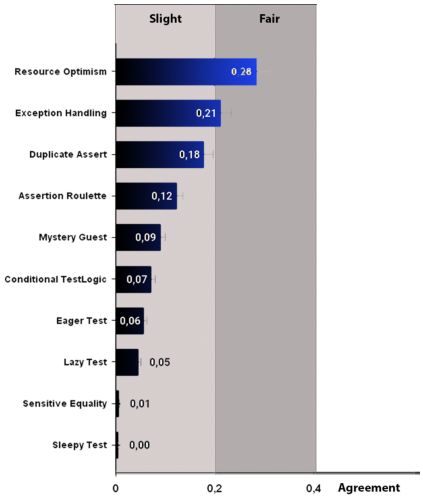

Figure 1: Inter-rater agreement for each Test Smell (Kappa)

3.2 Agreement from the participants' background factors

As described in Section 2.2, we collect the participants' *backgrounds*, indicating if they are *Developers* or *Data Scientists*. We use such information to investigate if the participants from the same background present higher agreement levels. In particular, we divide the 25 participants into two subgroups according to their backgrounds: *Developers* and *Data Scientists*. For each subgroup, we calculate the agreement among the evaluations of the participants belonging to the subgroup. Finally, we compare such subgroup agreement with the overall agreement depicted in Figure 1.

3.2.1 Factor: Participants' Background - Developers. Figure 2 illustrates the agreement observed in the evaluations done by the *Developers*. We use an *orange line* to represent the overall agreement levels replicated from Figure 1. This representation helps us to analyze if the *Developers* present higher agreement levels than the overall. Moreover, in order to identify the *strength* of each agreement, we represent (on top of each figure) the classification described in Table 4.

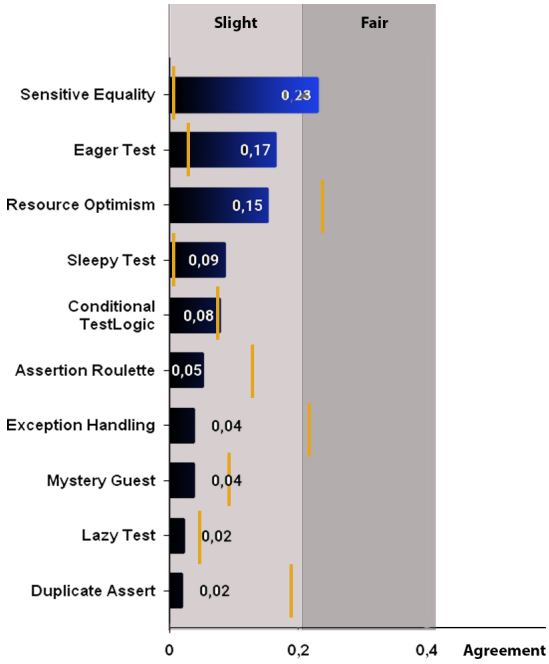

Figure 2: Agreement among Developers

We observe that the developers reach an agreement that varies from 0.02 up to 0.23. Similarly to the overall sample, the developers reach, at maximum, a *Fair* level. However, notice that we have a large increase in the agreement related to the Sensitive Equality when we only consider the developers. While the overall sample reaches an agreement of 0.01 (*Slight*), the developers reach an agreement of 0.23 (*Fair*). On the other hand, while the overall sample presents a *Fair* agreement in the Resource Optimism and Exception Handling, the developers present a *Slight* agreement in these smells types.

Table 6: Grey cells represent YES answers and white cells represent NO answers

Smell Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Total YES
AssertionRoulette		■	■	■			■	■	■	■	■	■	■	■	■				■	■		■	■		■	17
EagerTest	■		■				■		■	■	■				■	■				■			■	■	■	12
LazyTest		■		■	■		■	■					■		■		■		■	■		■		■		14
MysteryGuest	■	■	■	■			■	■			■	■			■				■							10
ResourceOptimism		■		■			■			■		■	■		■					■		■			■	11
SensitiveEquality	■	■	■		■	■	■	■			■	■	■		■			■	■							14
ConditionalTestLogic	■	■	■	■			■				■	■	■		■	■	■		■			■			■	13
DuplicateAssert	■	■		■			■	■		■	■	■			■	■							■	■	■	13
SleepyTest	■		■	■	■			■	■	■	■	■	■		■					■		■		■		13
ExceptionHandling		■	■	■			■	■	■	■		■			■					■		■	■	■		12

This result is surprising since we expect the results for Assertion Roulette and Duplicate Assert to present a better agreement than Eager Test, and Sensitive Equality. After all, Assertion Roulette is usually the smell type that developers are more knowledgeable about [22]. Duplicate Assert can also be easily identified since it is an instance of duplicated code. We also expect developers to have a better agreement in general because the assumption is that they would usually do more automated tests.

3.2.2 Factor: Participants' Background - Data Scientists. Figure 3 illustrates the agreement observed in the evaluations done by the *Data Scientists*. Differently from the other samples, the agreement level for one of the smells reaches 0.44, which constitutes a *Moderate* agreement. Also, when compared with the overall sample, we observe an increase in the agreement in the cases of the Mystery Guest, Lazy Test, and Sensitive Equality. The agreement increases from *Slight* to *Fair* in such cases. On the other hand, we also observe a decrease in the agreement from *Fair* to *Slight* in the cases of the Resource Optimism and Exception Handling.

These results suggest that *Data Scientists* tend to agree on the Conditional Test Logic and Mystery Guest smells. The first relates to the presence of conditional statements on the test, while the second indicates the presence of an external resource such as a file or a database record. Contrasted with the results from the developers' groups, we notice a surprisingly higher agreement level.

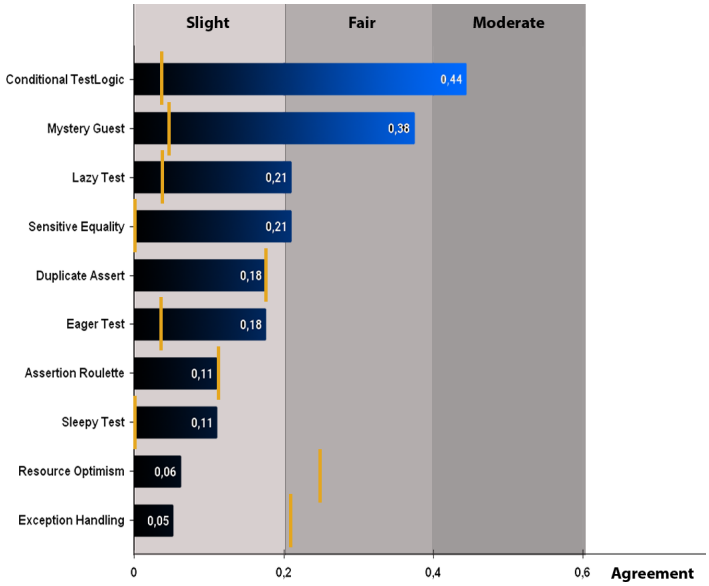

Figure 3: Inter-rater agreement among Data Scientist

3.3 Agreement from the participants' experience factors

Besides investigating if the background can influence the agreement among participants, we also investigate the influence of the participants' experience in the agreement. To do that, we analyze the agreement among the participants according to two subgroups: less and more experienced. The first subgroup contains participants with testing experience ranging between 0-2 years, and the second subgroup contains participants with more than three years of testing experience. The experience of each participant is defined by considering their self-reported experience in testing. In this way, we obtain two subgroups composed of at least five participants, which allows us to assess the agreement among their evaluations.

3.3.1 Factor: Participants's Experience - Less Experience. Figure 4 describes the agreement among the participants with less experience. We observe that the agreement levels vary from 0.02 up to 0.23. Similarly to the overall and developers sample, the less experienced reach, at maximum, a *Fair* agreement. Notice also that in both less experienced and developers groups, there is an increase in the agreement of the Sensitive Equality (from *Slight* to *Fair*), and a decrease in the agreement of the Resource Optimismand Exception Handling(from *Fair* to *Slight*).

The results suggest that less experienced participants tend to disagree with the presence of Test Smell. They are the only ones to have a fair agreement on Sensitive Equality. This smell occurs when the `toString` method is used within a test method. In general, we expect that less experienced participants would have a higher disagreement than experienced ones.

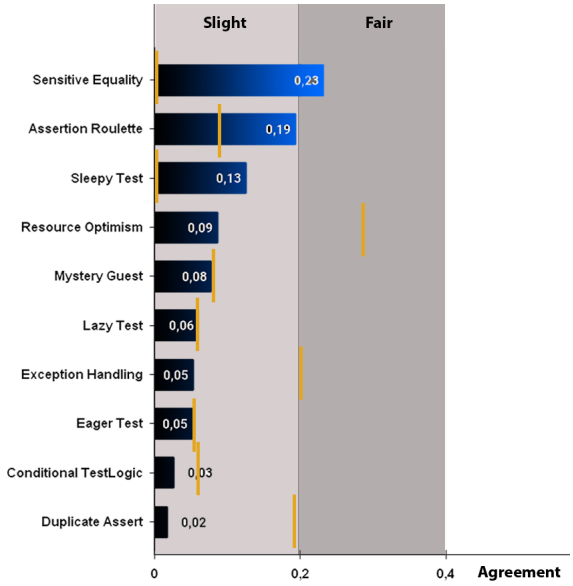

Figure 4: Inter-rater agreement for participant less experience (Kappa)

3.3.2 Factor: Participants's Experience - More Experience. Figure 5 describes the agreement among the participants with more experience. We observe that these participants reach, at maximum, a *Slight* agreement, varying from 0.02 up to 0.18. Similarly to the previous subgroups analyzed, when compared to the overall sample, the subgroup with more experience presents a decrease in the agreement in the Exception Handling smell (from *Fair* to *Slight*).

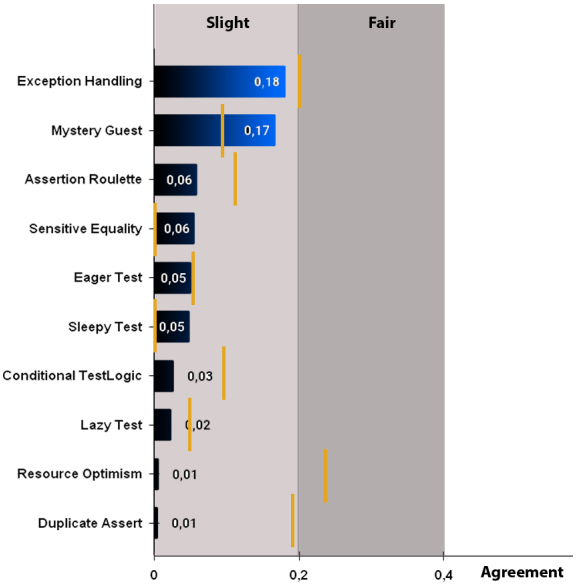

Figure 5: Inter-rater agreement for participant more experience (Kappa)

These results suggest that even the Most Experienced Participants tend to not agree with the presence of Test Smells. This result is a surprise because we do not expect the participants with more experience to have a lower agreement than the less experienced ones.

3.4 Agreement from the participants' heuristic factor

In addition to the analysis of the participants' background and experience, we also investigate if the heuristics to detect the Test Smells, which have been reported by the participants, influence the agreement among the participants' evaluations. Such heuristics are extracted from the participants' answers to the *open questions* provided during the study, as described in Section 2.3. In short, three specialists in Test Smell (three Ph.D. Students) applied a *coding* technique [19] to analyze the answers to recognize the heuristics adopted by the participants to detect the Test Smells analyzed in our study.

For example, from the participants' evaluations concerning the Assertion Roulette, the specialists recognize three different heuristics reported by the participants to detect this smell: (H1) Analysis of the test size, H2 Analysis of code duplication, and H3 Exception not properly handled. In this particular heuristic, the respondents are not specifically focused on a measurable attribute or language structure.

Table 7 describes the three heuristics identified and the *IDs* of the participants that reported each heuristic. We observe that seven participants detect this smell considering the H1 heuristic. Two participants consider the H2 heuristic, and the other one detects the smell by following the H3 heuristic.

Table 7: Assertion Roulette heuristics reported in open questions

#	Heuristic	Participants' ID
H1	Analysis of the test size only	22, 23, 25, 31, 41, 43, 55
H2	Analysis of code duplication	32, 33
H3	Exception not handled properly	24

The same procedure is also applied to identify the heuristics related to the other smell types. Table 8 reports the number of recognized heuristics for each smell type. Note that not all the smells types has recognized heuristics, since we identified those using the justification used by the participants for their answers.

Table 8: Heuristics recognized for each smell

Test Smell	Number of Heuristics
Assertion Roulette	3
Mystery Guest	1
Sensitive Equality	1
Conditional Test Logic	2
Eager Test	2
Sleepy Test	1
Exception Handling	2

We observe the most cited heuristic for each smell type, then we evaluate the agreement level among the participants who reported this heuristic. For example, considering the Exception Handling, we assess an agreement value of 0.63 (Substantial). This is the highest kappa value if we compare it with the corresponding *Kappa* values derived from the analysis described in the previous sections. By applying the same procedure to the other smells, we obtained the agreement levels depicted in Figure 6. We also observe that the Mystery Guest is the only smell type in which we have a decrease in the agreement when compared to the overall agreement.

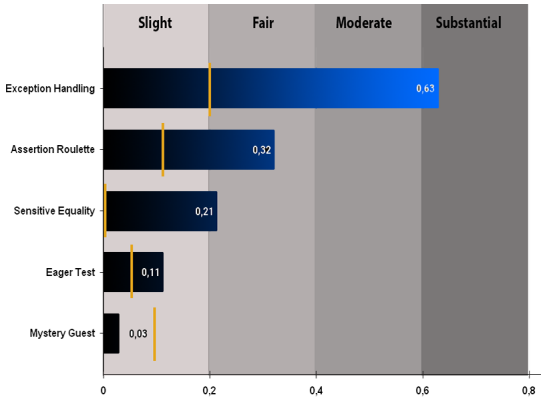

Figure 6: Inter-rater heuristics (Kappa)

4 THREATS TO VALIDITY

In this section, we discuss the validity threats in line with Wohlin et al.'s validity criteria [26].

4.1 Construct Validity

Construct validity involves potential threats during the study design and construction process. We do not provide participants with a definition of Test Smells to avoid introducing bias or influencing their responses. Instead, we aim for participants to assess test snippets based on their individual background and experience.

We manually gather Test Smells instances and perform cross-validation of test snippets to minimize potential issues. There should be any remaining false positives after inspection, we can still examine whether participants agree or disagree on the presence of Test Smells in those snippets, regardless of detection approach outcomes.

During the study, participants evaluate if a test snippet contains a specific Test Smell type. The scope of a test snippet may be insufficient for participants to make an accurate judgment, which we acknowledge and accept as a limitation.

Another potential threat pertains to the participants' assessments. We offer only "YES" or "NO" options, which may not allow them to express the confidence level in their responses. To mitigate this, we include open questions for participants to elaborate on their understanding of and uncertainties about test smells, enabling deeper analysis of their evaluations.

4.2 Internal Validity

Participants evaluate one test snippet for each smell type (10 in total), which might lead to some evaluations being completed without due diligence.

Conducting the study online helps us reach a reasonable number of participants, but we cannot strictly control communication among them during evaluations. To mitigate this, we randomize the order of test snippets and note the high disagreement in Section 3.1, suggesting limited collaboration among participants.

We do not thoroughly analyze whether participants properly applied the heuristics they reported, although we observe their application for Conditional Test Logic in Section 3.4.

Lastly, we examine the impact of participants' self-reported experience on evaluation agreement. Although self-assessment may be imprecise, previous work [7] suggests it is a reasonable proxy for experience.

4.3 External Validity

Our study involves 32 Java projects of varying sizes and domains, which are well-known and widely used in previous Test Smells detection research [4, 13, 20, 24]. However, our findings may not be generalizable to other projects with distinct characteristics.

We use 25 participants with diverse backgrounds and experiences (Section 2.4), but our results might not be universally applicable due to subjective factors affecting Test Smellsunderstanding.

Finally, participants evaluate 10 different Test Smell types, with generally high disagreement on detection. Our results may not extend to other smell types not analyzed in this study.

5 RELATED WORK

The study performed by Bavota et al. [1] analyzes the Test Smells distribution in software systems and whether their presence is harmful. As part of the investigation, they perform a controlled experiment involving 61 participants among students and industrial developers, which are asked to perform maintenance activities on smelly and refactored test code of two software systems. The results of the study indicate the negative impact of Test Smells in program comprehension during maintenance activities. Similarly to Bavota et al., our study performs experiments involving developers in the context of Test Smells, but our study focuses on the participants' perceptions about the existence of Test Smells.

The study performed by Soares et al. [3] conducts a study aiming to assess open-source developers' awareness about the existence of test smells and their refactoring strategies. They conduct a study by applying a survey with 73 open-source developers to assess their

preference and motivation to choose between 10 smelly test code examples, and the results indicate that most of the surveyed developers preferred the refactored proposal. Based on the results, they provide a empirical validation for the literature-proposed refactoring strategies.

Santana et al. [18] evaluated a tool called RAIDE, designed for the automatic identification and refactoring of Test Smells. They presented an empirical assessment of RAIDE, in which they analyzed its capability to refactor Assertion Roulette and Duplicate Assert Test Smells and compared the results against both manual refactoring and a state-of-the-art approach. The results showed that RAIDE provides a faster and more intuitive approach for handling Test Smells than using an automated tool for smell detection combined with manual refactoring.

6 CONCLUSIONS

We conducted an empirical study aimed at investigating the possible factors that influence the agreement of participants when evaluating Test Smells. To carry out our study, we collected test snippets from open-source projects and presented them to 25 participants.Precisely, we analyzed whether participants tend to agree (or disagree) on the occurrence of 10 different types of Test Smells.

Our study showed that, overall, software developers presented low agreement levels when evaluating all smell types, contradicting key results from previous studies [3, 21]. On top of that, although the developers' background and experience did not present a consistent influence on the level of agreement, the heuristics used by the developers for identifying test smells played an important role, being the factor that was better correlated with the agreement on the evaluations.

Our findings suggest that although many studies have been recently conducted on the topic, researchers and practitioners should not take for granted that developers will agree on the presence and harmfulness of test smells. This can be partially influenced by the fact that the definition of some test smells is naturally vague, but more studies are required to investigate this hypothesis.

We discovered three conclusions in our study. Firstly, participants exhibited low levels of agreement when evaluating all 10 Test Smells types, which contradicts key findings from prior studies [8, 17]. Secondly, participants' background and experience did not consistently affect their agreement levels. Lastly, the heuristic factor played the most significant role in determining agreement among participants. These findings highlight the importance of developing customizable techniques for detecting Test Smells that take into account each participants unique perception. The artifacts produced as part of our study are publicly available [3].

ACKNOWLEDGMENTS

This work is partially supported by INES (http://www.ines.org.br), CNPq grant 465614/2014-0, CAPES grant 88887.136410/2017-00, and FACEPE grants APQ-0399-1.03/17 and PRONEX APQ/0388-1.03/14. We also acknowledge support from CAPES (88887.496429/2020-00), FAPEAL (60030.0000002725/2022, 60030.0000000161/2022, 60030.0000000462/2020), FACEPE (APQ-0570-1.03/14), and CNPq (423125/2021-4, 315532/2021-1, 312195/2021-4). The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

[3] https://github.com/tests-smells/agreement

REFERENCES

[1] Gabriele Bavota, Abdallah Qusef, Rocco Oliveto, Andrea De Lucia, and Dave Binkley. 2015. Are test smells really harmful? an empirical study. *Empirical Software Engineering* 20, 4 (2015), 1052–1094.

[2] Jonathan Immanuel Brachthäuser, Sukyoung Ryu, Nathaniel Nystrom, Jonas De Bleser, Dario Di Nucci, and Coen De Roover. 2019. SoCRATES: Scala radar for test smells. *Proceedings of the Tenth ACM SIGPLAN Symposium on Scala* (2019), 22–26. https://doi.org/10.1145/3337932.3338815

[3] Everton Cavalcante, Francisco Dantas, Thais Batista, Elvys Soares, Márcio Ribeiro, Guilherme Amaral, Rohit Gheyi, Leo Fernandes, Alessandro Garcia, Baldoino Fonseca, and André Santos. 2020. Refactoring Test Smells: A Perspective from Open-Source Developers. *Proceedings of the 5th SAST* (2020), 50–59. https://doi.org/10.1145/3425174.3425212

[4] Jonas De Bleser, Dario Di Nucci, and Coen De Roover. 2019. Assessing diffusion and perception of test smells in scala projects. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 457–467.

[5] Prem Devanbu, Myra Cohen, Thomas Zimmermann, Anthony Peruma, Khalid Almalki, Christian D Newman, Mohamed Wiem Mkaouer, Ali Ouni, and Fabio Palomba. 2020. tsDetect: an open source test smells detection tool. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2020), 1650–1654. https://doi.org/10.1145/3368089.3417921

[6] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[7] Mário Hozano, Alessandro Garcia, Baldoino Fonseca, and Evandro Costa. 2018. Are you smelling it? Investigating how similar developers detect code smells. *Information and Software Technology* 93 (2018), 130–146.

[8] Nildo Silva Junior, Larissa Rocha, Luana Almeida Martins, and Ivan Machado. 2020. A survey on test practitioners' awareness of test smells. *arXiv preprint arXiv:2003.05613* (2020).

[9] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[10] Mika V Mantyla. 2005. An experiment on subjective evolvability evaluation of object-oriented software: explaining factors and interrater agreement. In *2005 International Symposium on Empirical Software Engineering, 2005.* IEEE, 10–pp.

[11] Mika V Mäntylä and Casper Lassenius. 2006. Subjective evaluation of software evolvability using code smells: An empirical study. *Empirical Software Engineering* 11, 3 (2006), 395–431.

[12] Luana Martins, Heitor Costa, and Ivan Machado. 2023. On the diffusion of test smells and their relationship with test code quality of Java projects. *Journal of Software: Evolution and Process* (2023). https://doi.org/10.1002/smr.2532

[13] Anthony Peruma, Khalid Almalki, Christian D Newman, Mohamed Wiem Mkaouer, Ali Ouni, and Fabio Palomba. 2020. Tsdetect: An open source test smells detection tool. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1650–1654.

[14] Anthony Peruma, Khalid Saeed Almalki, Christian D Newman, Mohamed Wiem Mkaouer, Ali Ouni, and Fabio Palomba. 2019. On the distribution of test smells in open source android applications: An exploratory study. (2019).

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

[16] Railana Santana, Daniel Fernandes, Denivan Campos, Larissa Soares, Rita Maciel, and Ivan Machado. 2021. Understanding practitioners' strategies to handle test smells: a multi-method study. *Brazilian Symposium on Software Engineering* (2021), 49–53. https://doi.org/10.1145/3474624.3474639

[17] Railana Santana, Daniel Fernandes, Denivan Campos, Larissa Soares, Rita Maciel, and Ivan Machado. 2021. Understanding practitioners' strategies to handle test smells: a multi-method study. In *Brazilian Symposium on Software Engineering*. 49–53.

[18] Railana Santana, Luana Martins, Tássio Virgínio, Larissa Soares, Heitor Costa, and Ivan Machado. 2022. Refactoring Assertion Roulette and Duplicate Assert test smells: a controlled experiment. *arXiv* (2022). https://doi.org/10.48550/arxiv.2207.05539 arXiv:2207.05539

[19] C.B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 25, 4 (1999), 557–572. https://doi.org/10.1109/32.799955

[20] Elvys Soares, Márcio Ribeiro, Guilherme Amaral, Rohit Gheyi, Leo Fernandes, Alessandro Garcia, Baldoino Fonseca, and André Santos. 2020. Refactoring test smells: A perspective from open-source developers. In *Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing*. 50–59.

[21] Elvys Soares, Marcio Ribeiro, Rohit Gheyi, Guilherme Amaral, and Andre Medeiros Santos. 2022. Refactoring Test Smells With JUnit 5: Why Should

Developers Keep Up-to-Date. *IEEE Transactions on Software Engineering* PP, 99 (2022), 1–1. https://doi.org/10.1109/tse.2022.3172654

[22] Davide Spadini, Fabio Palomba, Andy Zaidman, Magiel Bruntink, and Alberto Bacchelli. 2018. On the relation of test smells to software code quality. In *2018 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 1–12.

[23] Arie Van Deursen, Leon Moonen, Alex Van Den Bergh, and Gerard Kok. 2001. Refactoring test code. In *Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001)*. Citeseer, 92–95.

[24] Tássio Virgínio, Luana Almeida Martins, Larissa Rocha Soares, Railana Santana, Heitor Costa, and Ivan Machado. 2020. An empirical study of automatically-generated tests from the perspective of test smells. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*. 92–96.

[25] Tássio Virgínio, Luana Martins, Railana Santana, Adriana Cruz, Larissa Rocha, Heitor Costa, and Ivan Machado. 2021. On the test smells detection: an empirical study on the JNose Test accuracy. *Journal of Software Engineering Research and Development* 9 (2021). https://doi.org/10.5753/jserd.2021.1893

[26] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.