A study on the existence of tolerance to certain code smells and of implicit gender bias in Code Reviews

Larícia Maria Mota Cavalcante

1Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brazil
lmmc2@cin.ufpe.br

Abstract. Code review is an activity with the goal of guaranteeing code quality, in which the identification of code smells is an important practice. In addition to detecting code smells, another aspect of code review that is being increasingly studied is gender bias in pull request reviews, which intensifies prejudice against women on software development when PRs from women are rejected because of the author’s gender. This research aims to analyze these two aspects of code reviews through a survey questionnaire with 180 participants, where they were asked to simulate that they were reviewing code snippets and they had to choose only one to approve in each pair.

We were able to classify 8 different types of code smells in terms of detection difficulty and to analyze their code choices in terms of code smells rejection rate and dependence on the demographic aspects of the participant. Typo was classified as the most difficult code smell to detect, and, in fact, it was by far the one that was chosen the most as the snippet that they felt most comfortable in approving, corroborating our hypothesis that the more difficult the detection, the lower the rejection. Using Chi-Squared statistical tests, the participant’s demographic aspects and their code choices were considered independent variables, therefore the participant’s choice does not depend on their gender, for example.

Keywords— code review, gender bias, code smells, gender diversity, software development

1. Introduction

One of the most important practices in Software Development is Code Review, as it is a way to guarantee the quality of code and maintain consistency of the project. Some of the characteristics analyzed by reviewers include readability, maintenance, performance, coding style and code smells. The latter, code smells, is a hint that something can go wrong, so there is an opportunity to refactor the code and increase its quality. Reviewing the code before releasing it can significantly reduce possible negative consequences [29], since code smells can be identified in this process and the code refactored before entering the production environment. Developers agree that the identification of code smells, and its removal, is a relevant task to improve the source code quality [17].

Besides the undeniable importance of studying code smells in the code review context, another important aspect that is being considered in studies is the impact of sexism in the code review process. It is incontestable that women still struggle for respect and for their space in Software Development [7, 9, 12, 19, 23, 26], but recent studies are bringing up yet another form of prejudice against them: women’s contributions tend to be accepted more often than those of men, but only when their gender is not displayed [32].
This present study aims to analyze code reviews considering these two spectrums, code smells and gender bias. Our purpose is to investigate how developers feel about some code smells and also investigate if developers express gender bias when reviewing and approving code. The main idea is to answer these two research questions: "Is there any preferable/overlooked or strongly rejected code smell in code reviews?", as we analyze how they classified the detection difficulty of each code smell and if there were any smell being more tolerable than others reflecting a kind of preference or being overlooked, and "Is there implicit gender bias in the context of code approval?".

The data used on our analysis were the responses to a survey questionnaire we developed for this purpose. In this survey, respondents were asked to choose a code snippet in each pair of snippets. There were 12 pairs: 6 were directed to our code smell analysis, so these 6 pairs compared two different code smells in codes written by authors of same genders, the other 6 were directed to the study of gender bias, so we compared codes with the same code smell written by authors of different genders.

Each code snippet contained one code smell and a display of the author’s name. They did not know beforehand that code smells and gender were being analyzed, as we did not want this awareness to affect their responses. Furthermore, as we wanted to identify the implicit gender bias, we had to insert some strategies to mislead them, which is why these 12 questions were scrambled and, after the last pair of snippets was presented, we asked what they thought was the objective of the research. Then, in the process of preparing the data to be aggregated and analyzed, we looked at the answers to this goal question and removed all responses from those who mentioned gender bias.

We also asked respondents some demographic questions to test whether any of them could affect their code choices in any way. Analyzing whether gender and political inclination influenced respondents’ choices, it was not possible to identify a relationship between these variables and the responses. We used the Chi-Squared ($\chi^2$) statistical test for this.

Regarding our other approach, we asked respondents to rate 8 code smells for the difficulty of detection, these code smells were present in the snippets, but that information was not disclosed to the participants. It was assigned labels of classifications - Really Easy, Easy, Difficult and Really Difficult - to each code smell from the responses to this classification question. We found that in the pairs in which we compared a code smell that was considered Really Easy and a Really Difficult one, people tended to choose the snippet in which the Really Difficult one was presented. One possible explanation is that the more difficult a code smell is to be detected, the more likely it is to go unnoticed and, therefore, code reviews are more susceptible to be approved if the existing code smell is really difficult to detect.

2. Background

2.1. Code Reviews

Developers believe that code review is the best way to increase the quality of the code under development [16]. Code review corresponds to the practice performed by professionals to verify code in order to detect errors, both technical and semantical, and to improve the code structure. This practice can speed up the development of software and prevent future spending, as some errors may not be detected through automated tests [24].

The benefits of code review go beyond improving quality code and reducing the time and cost of software development. Code reviews allow more experienced developers to mentor the least experienced, there is an improvement in code maintenance and an increase in the collaborative spirit of the team, among others. Due to the importance of code review, there are several
materials with best practice guidelines to help in the execution of this activity, both for the author and for the reviewer [3, 5, 6, 22].

The way a company handles code reviews can reveal aspects of their culture [4]. In this way, it becomes clear that without a strong culture, code reviews could lead to a fear environment, making people play it safe without criticism, which defeats the purpose of this practice, or even transforms the environment into an abusive one. Both scenarios are harmful, since code review benefits can not be reaped in these situations.

### 2.2. Code Smells

"If it stinks, change it." This is how Martin Fowler and Kent Beck introduced the concept of code smells to the world. A code smell is any characteristic or structure in the code that indicates the possibility of refactoring, which will improve the code without writing new functionality. Although some code smells were introduced in Martin Fowler’s Refactoring book [21] and, over time, others were identified, they argue that "no set of metrics rivals informed human intuition". Therefore, the identification and the degree of penalty given to a certain code smell is subjective, which can vary according to the programming language used, the developer’s intuition and personal beliefs, and development methodology.

Since code smells can have several negative future consequences, such as difficulty in maintainability and comprehensibility [28, 10], it is natural to think that identifying code smell is an important and valuable skill in the code review process. A study, which analyzed over 20 thousand code reviews, found that code reviews did in fact have a significant influence on the likelihood of reducing the severity of code smells [29].

### 2.3. Gender bias in Software Engineering

Teamwork is as important as technical expertise on software engineering. Besides code review, another way to improve the quality of the work in development is to increase team performance. A lot of research is being made for this purpose, one of the main aspects linked to this increase in performance is the team diversity. Diversity of skills [13] and personality [30] impacts positively on team performance.

Gender differences are identified in personality traits, both women and men exhibit negative and positive traits linked to teamwork, such as women have higher Honesty-Humility and Openness to Experience than men, although men may be more emotionally stable, but men also score higher in relation to psychopathic traits [31]. Considering that, it becomes natural to think that gender diversity also plays a role in improving teamwork. In fact, there are already studies that support this assumption [14, 33]. However, it is known that there is a gender imbalance in the Software Engineers community [12, 19], the male software engineering massively outnumbers the female ones, as it is observed in LinkedIn’s data, only 16% of high tech software companies software engineers are female [26]. Also, in a recent analysis it is shown that the percentage of women in the total workforce of computer science-related jobs has dropped, in comparison to 1990, from 32% to 25% in 2016 [27].

Women who work in domains dominated by men face more gender bias [23], as Software Engineering is one of these domains, 70% of women in tech say they have been treated differently at work due to their gender, compared to 11% of men in tech [9]. In a study of GitHub pull requests rate of acceptance, it was found that code written by women was accepted 4% more than code written by men, but this only happened when the programmer’s gender was not revealed, when the gender was identifiable women’s acceptance plummeted 16% [32]. Several other types of difference in the treatment given to women in the workplace, due to gender bias, can be observed.
in the field of computing, such as being excluded from opportunities and fearing for their personal safety [7].

The women’s underrepresentation on Software Engineering is intimately related to the gender bias in the workplace, since it perpetuates harmful stereotypes and fosters an atmosphere of prejudice. 39% of women believe that gender bias is the reason they are underrepresented in technology [8].

3. Methodology

One of the main goals of this research is to verify the existence of implicit gender bias in the code review process. The most applicable studies found in this regard were the Github ones [25, 32] and Wang’s empirical study of implicit gender biases [34]. The former ones already done extensive research through millions of pull requests from Github, and the latter suggested that further replication studies in places other than the Western New York, where this took place, would bring additional understanding of implicit gender bias.

Thus, we were inspired by the latter one. It was decided that a survey questionnaire would best suit our research questions and hypotheses, and this survey would be contextualized as a program comprehension task, in which the respondents would have to choose a snippet of code between two snippets that implement the same functionality. Each code snippet would be associated with an author name, which is identified as male[2] or female[1] in Brazil, such as Rafael and Camila, respectively. This process relates to the third step of Wang’s study: Contributions Evaluation Task, with the difference that we would have code smells in all code snippets. At the beginning of the survey it was explained that a series of questions would be presented and each question would have a contextualization of a functionality that 2 developers were asked to implement, so what the participants were asked to do was to choose one snippet of code among the pair of snippets for each question, in relation to the one they prefer to approve. One of the questions used is in Figure 1.

In order to cover our two main topics in the universe of code review, code smells and implicit gender bias, we summarized our intentions for the survey questionnaire in two research questions and our hypotheses to each one of them, which are described below.

RQ1 Is there any preferable/overlooked or strongly rejected code smell in code reviews?

   H1 - The difficulty of detection plays a role: the easier the detection, the more rejection it gets - the harder the detection, the less rejection it gets.

   H2 - Reviewers have code structures preferences, which affects the level of rejection a code smell receives.

RQ2 Is there implicit gender bias in the context of code approval?

   H1 - In similar quality codes the one from male is more likely to be the most chosen one

   H2 - Male respondents tend to choose code from a male developer. Likewise, female respondents tend to choose code from a female developer.

3.1. Survey

To answer the research questions and hypothesis, it was necessary to gather information from multiple diverse people, from different backgrounds, such as age, where they live, political view,
programming experience, among others. A survey questionnaire strategy was chosen for its sharing facility and low cost.

The survey questionnaire idea is to simulate a program comprehension task, that is, the respondent will be asked to suppose they are reviewing two snippets of code, in each question, which implements the same functionality, and their job is to choose the snippet they feel the most comfortable approving between those two. Each snippet is identified by the author’s name, which is fictitious.

Considering both motivations of this research, which are: comparing difficulty detection versus rejection proportion of code smells and identifying implicit bias in relation to gender, those questions containing snippets of code are divided into two sets. A set of questions to address the RQ1 where two codes, each one with a different code smell but having code authors of the same gender\(^1\), was shown and another set of questions to address RQ2 where two codes, with the same code smell but different genders for the code authors, were shown. Those differences were inserted so that we could remove the gender bias variable from the first set and the tolerance or rejection to code smells from the second set, although we did not make a distinction between those sets in the

\(^1\)Due to research limitations of time and staff, in order to simplify the survey questionnaire and the results analysis, it was used only male and female genders. Those genders are inferred by names, we used names usually linked to a specific gender.
Figure 2. The code smell classification question in the survey questionnaire.

An important point of attention to this survey was to not display its intention to the respondents because it was all based on implicit aspects. If the respondents knew the research goal, the implicit gender bias, then the answers would be contaminated by this unwanted awareness. Therefore, it was asked to the respondent what was their hunch about the research goal, that way it enabled us to separate the answers from who got the idea of implicit bias in relation to gender. Also, the lack of distinction between the sets of questions with code snippets was purposeful, so that people would not get suspicious seeing a sequence of questions containing code snippets with authors of different genders being compared.

The code smells were chosen considering the ones that most appear in the literature, with Fowler’s Refactoring book [21] being our main reference on this subject, and, among these, the ones that are simplest to show using small snippets of code. It was important that the code smells were explicit in small snippets of code because in a survey questionnaire it is essential to be concise in order to keep the respondents’ attention and prevent dropouts. This is also the reason why there is not a high number of questions containing snippets of code. Like the gender bias aspect, we also did not reveal that every code snippet had a code smell associated. After these sets of questions with code snippets, we asked their hunch about the research goal and then we asked them to classify each code smell, as it’s shown in the Figure 2.

In this classification question, we briefly explained what a code smell is and what each one of the code smells we asked to be rated is, in order to be inclusive with respondents who were unfamiliar with these terms. The following code smells and descriptions were used:

1. Naming: when something is badly named, that is, it is not well explained or is unnecessarily long or way too short, for example;

2. Comments: some comments are used as “deodorant”, as in cases where the name of a
variable is not explanatory or a function is poorly written;

3. Typo: typographical error;

4. Long Function: the bigger the function, the more difficult it will be to understand and update. A function is considered long when it has more than one purpose, for example;

5. Long Parameters List: a long list of parameters can be unnecessary and often confusing;

6. Loops: compared to traditional loops, pipeline operations such as filter and map can help us to quickly see the elements in the process and what is being done to them;

7. Duplicated Code;

8. Repeated Switches: if a clause has to be added or changed, then different switches, or sequences of if, will have to be modified.

About the code snippets, JavaScript (JS) was the chosen programming language because we wanted a highly common and widely used language in order to get as many respondents with this language knowledge as we could. In the StackOverflow’s 2020 Developer Survey\(^2\), we can see JavaScript, for the eighth year in a row, as the most commonly used programming language.

The JS code snippets were extracted from the second edition of the *Refactoring* book [21] and some websites [20, 11]. For the questions that had codes with the same code smell, we adapted the examples to build two similar codes with the same problem. There are also a few snippets that were completely built by us, which is the case of the snippets with the Typo smell.

Demographic questions were inserted in the questionnaire in order to allow the aggregation of individual responses in groups, so that these groups serve as a means of synthesis and analysis of the data. The survey asked for the participant’s age, gender, education level, state of Brazil where they live, or if they live abroad, programming experience time, employment situation and political inclination.

Studies have already shown that social dominance orientation (SDO) and right-wing authoritarianism (RWA) predicted some kind of sexism, especially prejudice against working women [15]. Capturing that essence, we made some assumptions regarding the possible link between certain groupings of data and their code choices, such as if a specific political inclination is related to the respondent tendency to choose code from one specific gender.

### 3.1.1. Pilot

Firstly, a pilot version was created to be administered to only 15 people so the final survey would have modifications from the pilot’s feedback. Of these 15 participants, 3 were female and 12 were male. The pilot was composed of 10 questions containing snippets of code, a question to classify each code smell regarding its difficulty of detection, and a set of demographic questions.

The 10 code questions were divided into: 6 questions of the “same gender - different code smell” category and 4 questions of “different gender - same code smell”. The code smells used are described in Table 1.

\(^2\)https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-all-respondents
The topics we wanted to investigate with this pilot version were mainly: the questionnaire length, how people felt about answering question about their political inclination, the average time it took them to answer, and if they had any difficulty reading the code or understanding what they were supposed to do, in addition to any other feedback they wanted to give. Therefore, we interviewed the 15 respondents afterwards and most of them felt it would be acceptable to add more questions, their opinions ranged from 2 to 5 more code questions, only one said the pilot already had the maximum number of questions. They took an average of 15 minutes to complete the questionnaire and everyone was comfortable with answering the question about political inclination, they were just really curious about why there was this question.

Analyzing the responses we had other findings, such as, there were two unbalanced code questions in the set of 4 questions with "Different Gender - Same Code Smell", since 100% of the respondents chose the same code snippet. One of them was about the Naming smell, everyone chose the snippet we planned to have nomenclatures considered “unnecessarily long” over the one with nomenclatures “unnecessarily short”, but then we concluded that actually comparing these two, it made sense to choose the first one, because it was not so long and the other was way too short. The other was about the Repeated Switches smell, everyone chose the one which “switch” was being used instead of the one using “ifs sequence”, demonstrating a strong preference for switch in comparison to if.

Through the question of code smells difficulty we got the classification shown in the Figure 3.
3.1.2. Final questionnaire

Based on the pilot’s findings, we decided to make some changes before releasing the final questionnaire. First, we added two more code questions with the same code smell each, so we would have 6 questions with “same gender - different code smell” and 6 of “different gender - same code smell”. In addition, we replaced the Long Function smell with the Long Parameters List smell based on the code smells difficulty answers and our assumption that the harder it is to detect the smell, the more active your implicit bias will be on your decision making. This new set is described in Table 2. Some of the code smells are repeated because we wanted to present each one in different orders, but we could not present all possible combinations because it would greatly increase the time required to respond the questionnaire.

Considering the questions that obtained 100% of answers in a single option, we decided to tweak their codes to balance them, since the choices had nothing to do with implicit bias and was clearly about the codes not being at the same level of quality. In the Naming smell we made both codes have a little bit longer names, and in the Repeated Switches smell we used “switch” in both codes.

About the demographic questions section, we added some more, such as experience time with JavaScript.

3.2. Data visualization and Analysis

In order to analyze the answers obtained through the questionnaire and aggregate this data into different categories to help us better understand our results, we implemented a data visualization script. We used the Python language due to its ease of data manipulation and visualization. Also, Google Colab service was used to help the implementation and execution of code.

This visualization was divided into two different approaches, each to correspond to one of our research questions. The part used to try to answer the one about implicit gender bias had a peculiarity, the dataframe used was a subset of that used for the other part. The criterion used for this separation was one of the questions in the survey questionnaire, which was about what the respondent’s assumption about the research objective would be. Our goal with this question was to identify who imagined that the research had something to do with implicit gender bias, if someone imagined it, then possibly their responses could be consciously biased with that. As it was not what we wanted, we built another dataframe with all the responses, except those. Since the second part was not about implicit bias, no response was removed based on that.

Overall, we had 180 responses, of which 9 mentioned the concept of gender bias. People from all regions of Brazil and even some from outside the country participated, but the responses were massively from the Northeast with 125 responses, more specifically from the state of Pernambuco, which had an impressive 106 responses. Their ages ranged from 16 to 39 years, but their

<table>
<thead>
<tr>
<th>Same Gender - Different Code Smell</th>
<th>Different Gender - Same Code Smell</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Naming x Comments</td>
<td>1- Duplicated Code</td>
</tr>
<tr>
<td>2- Long Function x Long Parameters List</td>
<td>2- Long Parameters List</td>
</tr>
<tr>
<td>3- Long Parameters List x Loops</td>
<td>3- Loops</td>
</tr>
<tr>
<td>4- Comments x Typo</td>
<td>4- Naming</td>
</tr>
<tr>
<td>5- Loops x Duplicated Code</td>
<td>5- Repeated Switches</td>
</tr>
<tr>
<td>6- Typo x Naming</td>
<td>6- Typo</td>
</tr>
</tbody>
</table>

Table 2. Code smells used in the code questions of the final questionnaire.
gender identity did not have the same diversity, 147 identified themselves as male, while only 32 as female and 1 as a non-binary person. Over 80% of respondents work with software development and they have a varied programming experience, ranging from people with less than 1 year of experience to people with more than 7 years. Specifically about the experience with JavaScript, the range was the same, but the majority were people with 1 to 3 years, 52.8% of the answers, and there were even people with no experience, 7.2%.

We also asked some unconventional questions, about political inclination and religion/spirituality/existential worldview. On the political question, we obtained the majority with leftist inclination, 56.7% joining the full left and the center-left, against 20.2% with right-wing beliefs. Meanwhile, the religion question had many different answers, but the most frequent ones were the people who have a connection with Christianity, 30%, and the agnostics, 23.9%.

Some questions had several options to choose from or even the possibility of being answered in an open field, so when these questions were used in our analysis we grouped them according to their frequency. Age was one of them, we divided the answers into 4 different groups, each with 40 and a few responses, 16 to 22 years, 23 to 24, 25 to 28 and 29 to 39. Other examples were political inclination with 3 groups, left, right and others, and the question about where they live, also with 3 groups, Pernambuco, Southeast and Others.

4. Findings

In this section, we will briefly recall some adjustments made after the pilot version, showing the analyses on which we based ourselves to make these adjustments decisions. However, we will dedicate more space to show our findings in relation to the responses to the final questionnaire, in which we did a more extensive analysis. Therefore, at the end of this section, we will use a space to discuss some of these findings from the final questionnaire.

4.1. Pilot

The pilot’s goal was to understand whether the survey questionnaire was making sense and whether any adjustments were needed. We received feedback from the 15 participants that resulted in improvements, but we also analyzed the responses using data visualization.

Analyzing the data of our, still 4 - in the pilot -, questions of same smells, but different genders, we obtained the results that, in general, the snippets written by male were being chosen more and that there were 2 questions with 100% answers for a specific option, as previously stated in Subsection 3.1.1. This can be seen in Figure 4, in which the first chart shows the frequency in general, grouping all 4 questions, and the second shows the frequency divided by each of the 4 questions. In addition, there was a tendency for females to choose female code and for males to choose male code as shown in Figure 5.
As it was essential that the codes were balanced in terms of quality, we interpreted the fact that there were options receiving 100% of “votes” as an indication that these snippets should be recalibrated. Before launching the final questionnaire, this issue was addressed. Also, we used the difficulty detection classification [Figure 3] given by these respondents to build the 2 new questions added to the final survey, choosing the 2 most difficult code smells: Duplicated code and Typo.

4.2. Final questionnaire

To facilitate understanding, this section will be divided into two subsections according to our research questions mentioned in the Section 3.

4.2.1. RQ1: Is there any preferable/overlooked or strongly rejected code smell in code reviews?

Since the pilot version, we asked our participants to rate each code smell used in the code snippets in terms of their detection difficulty, therefore, in the final questionnaire we kept this question, and the updated result is shown in Figure 6. In addition, we wanted to see if there was any visible difference between the classification given by male participants and female participants, as well as by less experienced compared to more experienced ones. The charts by gender are in the Figure 7 and the ones by experience are in the Figure 8.

After summarizing the classifications given by all participants, we obtained the following labels for the code smells:

1. Really easy: Comments and Naming;
2. Easy: Long Parameters List and Long Function;
3. Difficult: Duplicated Code, Loops and Repeated Switches;
4. Really difficult: Typo;

For each question, we plotted a frequency chart of the choices in relation to the code smell associated with the chosen code snippet. All of these charts are present in Figure 9.
Lastly, we noticed that it would be interesting to further analyze half of the 6 questions addressed to this RQ, because each of these 3 questions compared a pair of different code smells that have different classifications. Two of them compared a Really Difficult smell with a Really Easy one and the third compared a Difficult smell with an Easy one. That said, we wanted to compare the frequency of the participants’ choices regarding the ratings for each of these sets, Really Easy x Really Difficult and Easy x Difficult, the 2 charts are in the Figure 10.

4.2.2. RQ2: Is there implicit gender bias in the context of code approval?

Primarily, we re-run the same analyzes made with the pilot version, now using the responses of our 171 participants and the 6 questions of the same smell, but different genders. These charts can be seen in the Figure 11.

Furthermore, other demographic answers were taken into account, such as age, where they live, political inclination and experience with JavaScript, all followed the same pattern.

In order to compare the respondent’s gender, age, political inclination and JavaScript experience with their choices divided by the gender of the code snippet’s author, the Chi-Squared ($\chi^2$) statistical test was used. The use of $\chi^2$ is indicated when testing independence between categorical variables is needed, so we are able to determine whether the output variable, in our case the chosen code snippet option, is dependent on or independent of the input variables, in our case the respondent’s demographic answer. This statistical test does this using a contingency table, which is a table summarization of two categorical variables. An example of a contingency table is shown

---

$^3$180 minus the 9 participants who mentioned gender bias in the goal question
in the Table 3, all of our contingency tables follow the same pattern, one variable to be tested is always the code snippet’s author gender and the other is the possible options for the respondent’s demographic aspect being analyzed.

Especially for the respondent’s gender, we also conducted a separate test for each question with the same code smell, but different gender. Therefore, for gender, we did 7 statistical tests. Using a probability of 95%, only 1 of these tests rejected the null hypothesis, that is, it resulted in “dependent” variables. This test compared the gender aspect in the question related to the Naming smell. Those tests’ statistic and p value are shown in Table 4, the first column represent the tests conducted, that is, the 7 tests related to the participant’s gender and the others related to political inclination, JavaScript experience or age.
5. Discussion

Once again valuing clarity and organization, this section will be divided into two parts according to the research questions.

5.1. RQ1: Is there any preferable/overlooked or strongly rejected code smell in code reviews?

Upon further analysis on each of the 6 questions with different code smells, some assumptions were raised. We will be discussing each question - different code smell, same gender - in the list below, following the order shown in the first column of Table 2. All code snippets are present on the Appendix A.

1. **Naming (Really Easy) x Comments (Really Easy):** In the code snippet associated with Comment there was a comment instead of an assert, but even so, it was chosen by almost 75% in comparison to the code associated with Naming. This indicates that bad naming is more rejected than the addition of comments.

2. **Long Function (Easy) x Long Parameters List (Easy):** After distributing the questionnaire we discovered that the second code snippet of the second question had an error, but still
it was the most chosen, with almost 74%. This may indicate that Long Parameters List is more overlooked or other aspects of the code influenced, such as the fact that this code snippet was more modularized than the one associated with Long Function.

3. **Long Parameters List (Easy) x Loops (Difficult):** Once again, Long Parameters List had more than 70% of “votes”, showing that even when compared to a different code smell from a different classification - Long Function, from the previous question, is considered Easy, while Loops is Difficult - still maintains a high rate of choice. This makes us question whether the Long Parameters List is really a code smell. In Fowler’s Refactoring book, the negative aspect cited is “long parameters list are often confusing”, we may raise the question whether this is a popular opinion. If we were to classify code smells in terms of their degree of penalties, then Long Parameters List would be at the bottom of the list as one of the smells with the least negative effects and that would support our Hypothesis 2.

4. **Comments (Really Easy) x Typo (Really Difficult):** Unlike the first question, Comments
had a high rejection rate, over 82% chose the code with a Typo. Two interpretations could be extracted, in this code the comment may have been considered more “unnecessary”, but also the typo may have gone unnoticed because it occurred only once in the word “circumference”.

5. **Loops (Difficult) x Duplicated Code (Difficult)**: The snippet associated with Loops was chosen by over 90% of the interviewees, this high rate raised a flag in our analysis and when we reviewed the codes we realized that it is possibly unbalanced in terms of quality. This is due to readability, the second snippet, the one associated with Duplicated Code, even using pipelines, such as filters, is really complicated to understand. It lost a lot of its readability because of the duplicated code, so it makes sense that the normal loop would be preferred to this one. This indicates that it is not always advisable to replace a loop with a pipeline.

6. **Typo (Really Difficult) x Naming (Really Easy)**: This had the most astonishing result, the code associated with Typo was chosen by almost 98% of the participants. This supports the high rejection that Naming receives, since it is not in the code snippet chosen by the participants as the one they would like to approve, and also how much typo is often overlooked. The code snippets used can be seen in Figure 13

Analyzing the charts that use classification labels, we found that they do not change noticeably in relation to the respondent’s experience with JavaScript or their gender. The perception of the level of difficulty remains unchanged.

Something interesting that we noticed when comparing codes of different classifications is that the more distant the classifications are from each other, the greater the tendency for people to choose the most difficult one, probably because being hard to detect makes it go unnoticed. This corroborates the statement we made in Hypothesis 1. Indeed, there is a link between the ability to identify code smells and individual skills, and a study provides evidence that human aspects may improve smell identification tasks [18]. Therefore, code reviewers should practice their detection skills with those code smells that are considered difficult, in order not to approve code with a smell just because they have not seen it.
5.2. RQ2: Is there implicit gender bias in the context of code approval?

To answer this question, two approaches were used, investigating the frequencies of male or female code being chosen and statistical tests to check if any demographic aspect of the respondent affected their choice.

Following the pattern of the previous subsection, we will discuss each question of different gender and same code smell in the list below. All code snippets are present on the Appendix B.

1. **Duplicated Code (Difficult):** For this question it is even relevant to show the code snippets used [Figure 14], as they were practically the same, both duplicating the same code, changing only a nomenclature, but both preserving the same idea of naming. Both male and female interviewees chose more the code associated with a male author, almost 60%.

2. **Long Parameters List (Easy):** This one also had practically no difference, the difference was that the male code used a “foreach” while the female code used a normal “for”. The latter was the most chosen one with almost 60%. Compared to what happened in the previous question, perhaps the order of the code influences the decision, since in the two questions the most chosen was the second code snippet and both snippets were very similar.

3. **Loops (Difficult):** In the third question there was also the “foreach”/“for” difference, but this time the most chosen was the “foreach” one. However, the most chosen, with almost 65%, was again the second snippet, written by a female author, reinforcing the idea of order having influence.

4. **Naming (Really Easy):** This question associated with the Naming smell was actually recalibrated after the pilot version, it was one of the questions that received 100% of “votes” for a single option. It was recalibrated by increasing the size of the names of the variables, because in the pilot version the first snippet’s variables names were not long enough to be considered bad and those in the second snippet were too short. The gender of the author associated with the snippets was also switched, but the most chosen snippet remained the one with long variables names. In this final questionnaire, this snippet was associated with the female gender and was chosen by 70% of the participants, so very long nomenclatures are preferable to short ones, regardless of the gender of who wrote the code.

5. **Repeated Switches (Difficult):** Another question that had recalibration in the snippets, since it was found in the pilot version that people had a preference for “switch” instead of a sequence of “ifs”, the recalibrated version used switch case in both snippets. However, the most chosen one had the characteristic of being modularized, which helped in legibility and comprehension, in addition to increasing maintainability. This shows an imbalance in relation to quality. The most chosen one was the female code, with 64%.

6. **Typo (Really Difficult):** The question with the most difficult code smell of our selection, according to the classification extracted from our responses: Typo. The snippets had the same amount of typing errors, the one associated with male had “discont”(instead of discount) and “shiping”(instead of shipping) while the one associated with female had “iten”(instead of item) and “discont”(discount). The one chosen by the majority was the male one. Nevertheless, the participant’s level of English proficiency may have made it difficult to identify typos, in addition to the fact that “iten”(item) appeared on the very first line. This assumption is corroborated by feedback from some of the participants of this final questionnaire, some people commented that they initially thought that the codes were the same, but later saw “iten”(item). Nobody talked about the other typos.
Therefore, by the charts presented and this discussion, we can affirm that our hypothesis I has not been proven. Regarding the second hypothesis, we were also unable to prove it on the basis of statistical tests. The Chi-Squared tests refuted our second hypothesis, since the variables of the respondent’s gender and the choice of code are independent. Hence, we could not claim that there is a tendency to choose according to the respondent’s gender, and even going a little further than what was stated on the H2, we can extend this discovery to the fact that no dependency was found between the choice of code and the respondent’s political inclination, age or JavaScript experience.

6. Conclusion and Future Research

In the Section 5, we examined the results and found that for the first research question we could corroborate the hypotheses formulated, however for the second we were unable to use our responses as a means of proving our hypotheses.

Although it has been discussed a little before, in this section we will make it clearer about the problems identified that threaten the validity of this research or that prevented us from having more assertive conclusions. In addition, we will end by making suggestions for future work that can be done to continue this research.

6.1. Threats to Validity

An important point of attention is the calibration of the code snippets. After the pilot, some snippets were recalibrated as explained, but others were added and we were unable to recalibrate and distribute the questionnaire again. Therefore, there is a possibility that the snippets being compared are not as similar as they should be in terms of quality. Therefore, the choice would be more about the code itself and the implicit bias would not be manifested.
In the same sense, it is important to consider human failure, since the code snippets were created from scratch or extracted from some reference and adapted to meet our needs by a person. So, snippets are likely to have different levels of quality that should be recalibrated as other people review them.

Besides the level of quality, the choice of which code to accept is quite subjective, which may have something to do with the gender of the code author and/or many other aspects that are taken into account. Also, the code snippets presented are relatively straightforward, which may be the reason why gender bias did not have a strong impact on the final questionnaire.

Lastly, the number of female or non-binary gender of respondents was incredibly low. Less than 18% identified themselves as female and only 0.6% as non-binary. This affects our study of how much the respondent’s gender can influence their choice, which corresponds to our H2 of the RQ2.

6.2. Future Research

This research was a preliminary one, so an extension would be to run it again without associating gender to the code snippets, referring to them by number, for example, and analyzing whether something changes.

Some improvements would be to put the mentioned recalibration into practice with a team of reviewers, distribute a set of pilot versions and invest in divulgence to increase the diversity of respondents. An improvement directed to the questioning we made about whether the order of the snippets influenced the participants’ choices would be to use the Latin Square concept. This concept should be used to distribute the snippets, sometimes associating it with one gender and sometimes with the other, and also making it the first snippet to appear and at another time the second to appear.

In this research, the combination of code smells in each questions with different code smells was made in an arbitrary way. A future research could structure better those different code smells questions in order to compare different classifications of code smells. Besides adding more questions comparing the same code smells in order to have more data to support conclusions about those comparisons.

Finally, another direction that could be taken would be to analyze whether there is a difference in the ability to detect code smells depending on the reviewer’s gender.

References


Figure 15. Naming (Really Easy) x Comments (Really Easy)


Appendices

A. Code snippets: RQ1

Question 1: Naming (Really Easy) x Comments (Really Easy) - Figure 15

Question 2: Long Function (Easy) x Long Parameters List (Easy) - Figure 16

Question 3: Long Parameters List (Easy) x Loops (Difficult) - Figure 17

Question 4: Comments (Really Easy) x Typo (Really Difficult) - Figure 18

Question 5: Loops (Difficult) x Duplicated Code (Difficult) - Figure 19

Question 6: Typo (Really Difficult) x Naming (Really Easy) - Figure 20
B. Code snippets: RQ2

Question 1: *Duplicated Code (Difficult)* - Figure 21

Question 2: *Long Parameters List (Easy)* - Figure 22

Question 3: *Loops (Difficult)* - Figure 23

Question 4: *Naming (Really Easy)* - Figure 24

Question 5: *Repeated Switches (Difficult)* - Figure 25

Question 6: *Typo (Really Difficult)* - Figure 26
function trackSummary(tracker) {
    let totalTime = tracker.endTime - tracker.startTime
    let totalDistance = 0
    const earthRadius = 6371e3
    for (let i = 1; i < tracker.points.length; i++) {
        const x = (tracker.points[i].endLong - tracker.points[i].startLong) * Math.cos((tracker.points[i].startLat + tracker.points[i].endLat) / 2)
        const y = tracker.points[i].endLat - tracker.points[i].startLat
        const d = Math.sqrt(x * x + y * y) * earthRadius
        totalDistance += d
    }
    const pace = totalTime / 60 / totalDistance
    return {time: totalTime, distance: totalDistance, pace: pace}
}

function trackSummary(tracker, points, numberOfPoints, earthRadius, startTime, endTime) {
    const totalDistance = calculateDistance(tracker, points, numberOfPoints, earthRadius, 0)
    const timeAndPace = calculateTimeAndPace(tracker, startTime, endTime, totalDistance)
    return {
        timeAndPace, distance: totalDistance
    }
}

function calculateTimeAndPace(tracker, startTime, endTime, totalDistance) {
    return {
        time: endTime - startTime, pace: (endTime - startTime) / 60 / totalDistance
    }
}

function calculateDistance(tracker, points, numberOfPoints, earthRadius, totalDistance) {
    for (let i = 1; i < numberOfPoints; i++) {
        const x = (points[i].endLong - points[i].startLong) * Math.cos((points[i].startLat + points[i].endLat) / 2)
        const y = (points[i].endLat - points[i].startLat)
        const d = Math.sqrt(x * x + y * y) * earthRadius
        totalDistance += d
    }
}

function getProgrammersNames(team, programmerJob, names) {
    names = team
        .filter((member) => member.job == programmerJob)
        .map((member) => member.name)
    return names
}

function getProgrammersNames(team) {
    const names = []
    for (const member of team) {
        if (member.job === 'programmer') {
            names.push()
        }
    }
}

Figure 16. Long Function (Easy) x Long Parameters List (Easy)

Figure 17. Long Parameters List (Easy) x Loops (Difficult)
function circumference(radius) {
    // calculate circumference of a circle
    return 2 * Math.PI * radius
}

function calculateCircumference(radius) {
    return 2 * Math.PI * radius
}

function amountToPay(employees) {
    let amount = 0
    for (const employee of employees) {
        if (hasMinimumPercentage(employee)) {
            amount += 0.05 * employee.salary;
        }
    }
    return amount
}

function hasMinimumPercentage(employee) {
    return employee.monthsDisabled > 6 || employee.isPartTime
}

function amountToPay(employees) {
    let remainingEmployees = employees
    let amount = 0
    amount += remainingEmployees
    .filter((employee) => employee.monthsDisabled > 6)
    .map((employee) => 0.05 * employee.salary)
    .reduce((a, b) => a + b, 0);
    remainingEmployees = remainingEmployees.filter(
        (employee) => employee.monthsDisabled <= 12
    )
    amount += remainingEmployees
    .filter((employee) => employee.isPartTime)
    .map((employee) => 0.05 * employee.salary)
    .reduce((a, b) => a + b, 0)
    return amount
}
const AUTHENTICATION_ERROR = 'Sorry, we faced an issue authenticating, Try again!'
const DUPLICATE_ERROR = 'Oops, it seems this name already exists. Try another one!'
const CONNECTION_ERROR = 'Please check your internet connection and try again later.'
...

`feature/errors-constants` Blame Daniel (1 hour ago)

const ERROR_1 = 'Sorry, we faced an issue authenticating. Try again!'
const ERROR_2 = 'Oops, it seems this name already exists. Try another one!'
const ERROR_3 = 'Please check your internet connection and try again later.'
...

`feature/errors-constants` Blame Rafael (1 hour ago)

---

Figure 20. Typo (Really Difficult) x Naming (Really Easy)

```java
class Manager extends Employee {
    //...
    constructor(name, id, employees) {
        this.name = name
        this.id = id
        this.employees = employees
        //...
    }
    get totalAnnualCost() {
        return this.monthlyCost * 12;
    }
    //...
}

class Engineer extends Employee {
    //...
    constructor(engName, engId, manager) {
        this.name = engName
        this.id = engId
        this.manager = manager
        //...
    }
    get totalAnnualCost() {
        return this.monthlyCost * 12;
    }
    //...
}
```

---

class Manager extends Employee {
    //...
    constructor(managerName, managerId, employees) {
        this.name = managerName
        this.id = managerId
        this.employees = employees
        //...
    }
    get annualCost() {
        return this.monthlyCost * 12;
    }
    //...
}

class Engineer extends Employee {
    //...
    constructor(name, id, manager) {
        this.name = name
        this.id = id
        this.manager = manager
        //...
    }
    get annualCost() {
        return this.monthlyCost * 12;
    }
    //...
}

`feature/employees` Blame Camilla (1 hour ago)

---

Figure 21. Duplicated Code (Difficult)
function owingSystem(
  invoiceOrders,
  invoiceDueDate,
  invoiceCustomer,
  outstanding,
  tax,
  amount
)
{
  console.log("**** Customer Owes ****")
  // calculate outstanding
  invoiceOrders.forEach(order => {
    amount = tax * order.amount
    outstanding += amount
  });
  console.log(`name: ${invoiceCustomer}, amount: ${outstanding},
              due: ${invoiceDueDate.toLocaleDateString()}\`)
}

function owingSystem(
  invoiceOrders,
  invoiceDueDate,
  invoiceCustomer,
  outstanding,
  tax,
  amount
)
{
  // calculate outstanding
  for (const order of invoiceOrders) {
    amount = tax * order.amount
    outstanding += amount
  }
  // print details
  console.log("**** Customer Owes ****")
  console.log(`name: ${invoiceCustomer}, amount: ${outstanding},
              due: ${invoiceDueDate.toLocaleDateString()}\`)
}
function acquireData(input, country) {
    const lines = input.split("\n")
    let firstLine = true
    const result = []
    for (const line of lines) {
        if (firstLine) {
            firstLine = false
            continue
        }
        if (line.trim() === "") continue
        const record = line.split(",")
        if (record[1].trim() === country) {
            result.push({
                city: record[0].trim(), phone: record[2].trim()
            })
        }
    }
    return result
}
const theNomenclatureOfTheFirstFruit = 'apple'
const theColorationOfTheFirstFruit = 'red'
const theNomenclatureOfTheSecondFruit = 'banana'
...

const nameFstF = 'apple'
const colorFstF = 'red'
const nameSndF = 'banana'
...

Figure 24. Naming (Really Easy)
function plumage(bird) {
    switch (bird.type) {
        case "EuropeanSwallow":
            return "average";
        case "AfricanSwallow":
            return bird.numberOfCoconuts > 2 ? "tired" : "average";
        case "NorwegianBlueParrot":
            return bird.voltage > 100 ? "scorched" : "beautiful";
        default:
            return "unknown";
    }
}

function airSpeed(bird) {
    switch (bird.type) {
        case "EuropeanSwallow":
            return 35;
        case "AfricanSwallow":
            return 40 - 2 * bird.numberOfCoconuts;
        case "NorwegianBlueParrot":
            return bird.isNailed ? 0 : 10 + bird.voltage / 10;
        default:
            return null;
    }
}

function plumageAndAirSpeed(bird) {
    return { plumage: plumage(bird), airSpeed: airSpeed(bird) }; 
}
function checkout(order) {
    const basePrice = order.quantity * order.itemPrice
    const quantityDiscount = Math.max(0, order.quantity - 500) * order.itemPrice * 0.05
    const shipping = Math.min(basePrice * 0.1, 100)
    return basePrice - quantityDiscount + shipping
}

function checkout(order) {
    const basePrice = order.quantity * order.itemPrice
    const quantityDiscount = Math.max(0, order.quantity - 500) * order.itemPrice * 0.05
    const shipping = Math.min(basePrice * 0.1, 100)
    return basePrice - quantityDiscount + shipping
}