UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ARAMIS SALES ARAUJO

A SURVEY ON DEVELOPER'S INTENTION UPON REFACTORING: ASSESSING REFACTORINGMINER'S EFFICACY

CAMPINA GRANDE - PB
2020

ARAMIS SALES ARAUJO

A SURVEY ON DEVELOPER'S INTENTION UPON REFACTORING: ASSESSING REFACTORINGMINER'S EFFICACY

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Rohit Gheyi.

CAMPINA GRANDE - PB

2020

A663s Araujo, Aramis Sales.
A survey on developer's intention upon refactoring: assessing refactoring. / Aramis Sales Araujo. – 2020.

6 f.

Orientador: Prof. Dr. Rohit Gheyi.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Desenvolvimento de software. 2. Refatoração de software. 3. Desenvolvedores de software - pesquisa. 4. Github. 5. RefactoringMiner. 6. Refactoring. 7. Firehouse interview I. Gheyi, Rohit. II. Título.

CDU:004.41(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

ARAMIS SALES ARAUJO

A SURVEY ON DEVELOPER’S INTENTION UPON REFACTORING: ASSESSING REFACTORINGMINER’S EFFICACY

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Rohit Gheyi
Orientador – UASC/CEEI/UFCG

Professor Dr. Everton Leandro Galdino Alves
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 2020.

CAMPINA GRANDE - PB

A survey on developer's intention upon refactoring: Assessing RefactoringMiner's efficacy

Aramis Araujo
aramis.araujo@ccc.ufcg.edu.br
Federal University of Campina Grande
Campina Grande, Paraíba, Brazil

Rohit Gheyi
rohit@dsc.ufcg.edu.br
Federal University of Campina Grande
Campina Grande, Paraíba, Brazil

ABSTRACT

Refactoring is an essential practice in software development, as it allows developers to improve design, readability, and maintainability. In this context, analysis tools such as *RefactoringMiner* intends to provide an arguably precise classification of refactoring types. However, there is a concern regarding the alignment between the refactorings identified by these tools and the developer's intention towards those changes. For example, the tool may fail to detect the developer's intention of doing a *Pull-up Method* refactoring. In this work, we are going to address this issue. Concretely, we will conduct a survey with over 200 experienced *Java* developers that contribute to highly rated and active open source repositories. Those developers will be surveyed regarding refactorings detected on specific commits they authored. The goal is to complement the evaluation provided by *RefactoringMiner*, asking developers whether they recognize and had the intention of performing the refactoring types detected by the tool rather than assuming the output as correct. Another goal is to assess whether the tool failed to detect refactorings performed by the developers.

KEYWORDS

Refactoring, RefactoringMiner, GitHub, Survey, Firehouse Interview

1 INTRODUCTION

Modern software thrives through the use of agile development techniques. Refactoring, being one of the most important, enables developers to promote changes to a software artifact without modifying its expected behavior [3]. It encourages improved maintainability and extensibility, substantial aspects of successful software products.

The study of refactoring is essential to understanding various aspects of coding behaviors and their consequences. This is upheld by the existence of many studies that often relate refactoring activities to other development phenomena, such as the introduction of bugs [2], occurrence of conflicts [4], and identification of code smells [1].

These studies often base their analysis on predefined samples of refactorings collected from code repositories. For this task, all the aforementioned studies used *RefactoringMiner*, a tool that automatically mines commit differences from a git repository, detects refactorings between changes, and classifies those into pre-established refactoring classes.

RefactoringMiner [9] has been evaluated concerning the tool's precision and recall rates, yet we aim to compare the reported results with the product of a different survey, potentially matching an expert's intention with the tool's output, assessing its false-positive and false-negative rates. False-positives refer to refactorings that were detected by the tool, but weren't confirmed by the developer, and false-negatives to refactorings that the developer had the intention of applying, but weren't detected. These metrics are important because they help to demonstrate the tool's accuracy and consequently its efficacy and usability.

The goal of this study is to improve the evaluation of the said tool, by validating results obtained from it with data collected from a survey conducted with over 200 developers. In order to better understand the cases where the tool fails and assess its validity, leading to impacts or reaffirmations on previous works that were based on *RefactoringMiner*.

*

2 RELATED WORK

mpirical studies about refactoring often rely on automated tools for gathering samples through mining code repositories. It is also common for these to perform a questionnaire-based survey with experienced developers to assess their opinion. For this task, one of the most famous tools is *RefactoringMiner*, which was used in a particular study by Silva et al.[8]. In their evaluation the tool was able to achieve very high precision (0.98) and recall rates (0.93)[8], which are the main object of this study.

Another study performed by Oliveira et al.[6] surveyed a group of developers with the goal of better understanding their behaviour when performing refactoring activities. From their study, we could gather a list of refactorings commonly available in popular *IDE* which was used to setup questions for a survey.

The survey conveyed in this work aims to ask experienced developers about their intention towards the detected refactorings, without assuming it as a correct finding. Developers are also asked for input in refactorings that may not have been detected by the tool.

3 METHODOLOGY

3.1 Repositories and commits

In order to reach experienced and active developers, repositories were queried through the use of *GitHub's API v3*. In this query, repositories that had Java as its programming language were selected and ordered regarding both their *stargazer* count and latest modification date, which are common characteristics among popular and active projects.

This observatory activity extended 90 days (from September to November 2020) and in total the topmost 2000 active repositories had their commits filtered by pushed date to match the targeted time frame of 90 days (prior to the starting date of the study).

Through an automated mining process, commits were analyzed and, with the use of RefactoringMiner, the detected refactoring types were obtained (Figure 1).

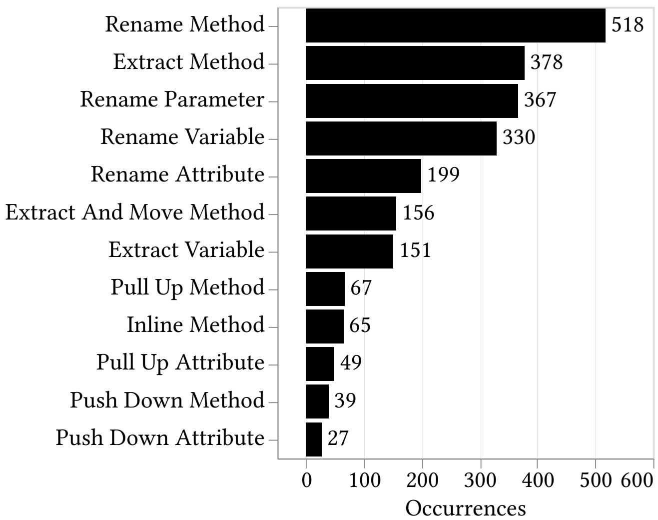

Figure 1: Detected refactorings per type

3.2 RefactoringMiner

RefactoringMiner is a tool that tries to detect and classify refactorings made throughout the commit history of a Java project. Through the use of its standalone executable, the repository's git *URL*, and a commit's *hash*, a *JSON* output is given containing the classified refactoring types detected by the tool at the specified commit.

3.3 Survey

Inspired by a *firehouse interview* [5, 8] through which participants are asked to answer questions related to recent activity, the survey targeted developers' commits that were at most 90 days old. For this task, e-mails were sent to the authors of these commits with simple questions regarding their activity in that particular commit. Those e-mails were limited to one per address to diversify answers and avoid confusing them with spam.

3.3.1 First Question. This question is related to the refactoring types that were detected by *RefactoringMiner.* Authors were asked whether they had the intention of performing the refactoring detected by the tool. For this question, authors were requested to answer one to at most three sub-questions with *Yes* or *No* referring to detected refactorings.

3.3.2 Second Question. With this question, the author is requested to answer whether it had the intention to perform any other refactoring not mentioned on the previous question(s) and, by answering affirmatively, it is asked to list the types of refactoring it had the intention of applying with that commit.

3.3.3 Third Question. Finally, it is asked whether the author performed any of the previously mentioned refactorings with the automated refactoring support of an *IDE.*

4 RESULTS

Of the 206 e-mails sent to distinct developers, 39 were answered denoting a response rate of 18.96% which is a higher rate than the 5% commonly found in questionnaire-based software engineering surveys[7]. This is expected behavior due to the *firehouse interview*[5, 8] approach. In total 61 instances of refactoring types had been analyzed by the developers, specifically, regarding the first question and its sub-questions.

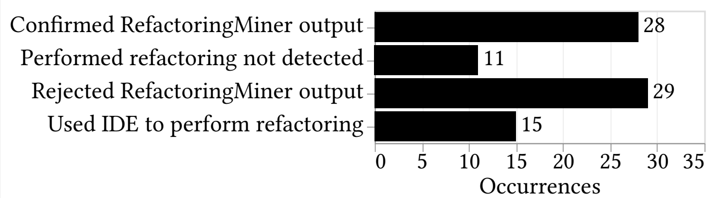

Figure 2: Classification of answers

With regards to the first question and its sub-questions, developers confirmed that they had the intention to perform the refactoring indicated by *RefactoringMiner* (subsubsection 3.3.1) in 28 out of 61 instances, resulting in a precision rate of 0.45 (Equation 1).

$$precision = \frac{|\{developer\ agreed\ with\ the\ tool\} \cap \{questions\}|}{|\{questions\}|} \tag{1}$$

Additionally, with the collected answers for the second question (subsubsection 3.3.2), the recall was determined as 0.49 (Equation 2).

$$recall = \frac{|\{developer\ agreed\ with\ the\ tool\}|}{|\{developer\ agreed\} \cup \{developer\ disagreed\}|} \tag{2}$$

Furthermore, developers affirmed not having the intention of performing the indicated refactoring in 11 instances (34.37%), indicating a much higher false-positive rate than seen in previous studies. With both Precision and Recall rates, the F_1 score for accuracy was calculated at 0.56 (Equation 3).

$$F_1\ accuracy = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

Concerning the third question (subsubsection 3.3.3), 15 candidates (38.46%) revealed they had performed the refactoring(s) with help from their *IDE.* Few spontaneously mentioned which IDE was used, the most popular being *IntelliJ IDEA*, *Eclipse*, and text editors such as *EMACS.*

5 CONCLUSION

Although the survey had an unremarkable number of responses, the findings of this study go against the stunningly high precision and recall rates proclaimed by previous studies for the tool in question.

RefactoringMiner is an efficient tool for detecting refactorings in Java projects, however, some indications conflict with the validity of the said tool, namely its low accuracy rate. It is also noticeable that the tool failed to identify some refactorings performed by the developers.

We believe that in regards to survey-based studies, especially those that involve analyzing the activity of a restricted group, a longer observation period is very beneficial, contrasting with the short four months period that this study was restricted to.

ACKNOWLEDGMENTS

I would like to thank my parents Analucia and Edson, for their utmost love and care. My sisters Milena and Lorena for setting up a high standard and helping me climb to it with all their love and patience.

To my friends, in special Adauto, Pedro Henrique (Ph), Samara, Alysson and Neize who took care of me during the hard, dark and scary times.

Thanks to professor Rohit Gheyi, for all his teachings, patience and opportunities that led me to my objectives. To professor Francisco Neto, whose "seal of approval" inspired me to move forward. To professors Joseana Macêdo, Tiago Massoni and Cláudio Campelo for the second chance.

Finally, thanks to all developers who took a sliver of their time to answer the survey.

REFERENCES

[1] Diego Cedrim, Alessandro Garcia, Melina Mongiovi, Rohit Gheyi, Leonardo Sousa, Rafael de Mello, Baldoino Fonseca, Márcio Ribeiro, and Alexander Chávez. 2017. Understanding the impact of refactoring on smells: A longitudinal study of 23 software projects. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 465–475.

[2] Isabella Ferreira, Eduardo Fernandes, Diego Cedrim, Anderson Uchôa, Ana Carla Bibiano, Alessandro Garcia, João Lucas Correia, Filipe Santos, Gabriel Nunes, Caio Barbosa, et al. 2018. The buggy side of code refactoring: Understanding the relationship between refactorings and bugs. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. 406–407.

[3] Martin Fowler. 2018. *Refactoring: improving the design of existing code.* Addison-Wesley Professional.

[4] Mehran Mahmoudi, Sarah Nadi, and Nikolaos Tsantalis. 2019. Are refactorings to blame? an empirical study of refactorings in merge conflicts. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 151–162.

[5] Emerson Murphy-Hill, Thomas Zimmermann, Christian Bird, and Nachiappan Nagappan. 2014. The design space of bug fixes and how developers navigate it. *IEEE Transactions on Software Engineering* 41, 1 (2014), 65–81.

[6] Jonhnanthan Oliveira, Rohit Gheyi, Melina Mongiovi, Gustavo Soares, Márcio Ribeiro, and Alessandro Garcia. 2019. Revisiting the refactoring mechanics. *Information and Software Technology* 110 (2019), 136–138.

[7] Forrest Shull, Janice Singer, and Dag IK Sjøberg. 2007. *Guide to advanced empirical software engineering.* Springer.

[8] Danilo Silva, Nikolaos Tsantalis, and Marco Tulio Valente. 2016. Why we refactor? confessions of github contributors. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 858–870.

[9] Nikolaos Tsantalis, Matin Mansouri, Laleh Eshkevari, Davood Mazinanian, and Danny Dig. 2018. Accurate and efficient refactoring detection in commit history. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 483–494.