What Programmers Say About Refactoring Tools?

An Empirical Investigation of Stack Overflow

Gustavo Pinto
Federal University of Pernambuco
ghlp@cin.ufpe.br

Fernando Kamei
Federal University of Pernambuco
Federal Institute of Sertão Pernambucano
fkk@cin.ufpe.br

Abstract

Programmers often use forums, such as StackOverflow, to easily and quickly solve their issues. Researchers then investigate those questions to better understand the state-of-use of software engineering techniques. Also, due to the quality and the great number questions and answers, the results found using such method might be difficult, or even impossible, to find using common survey techniques. In this study, we conducted a qualitative and quantitative research in order to categorize questions about refactoring tools. As a result, we presented a comprehensive classification of flaws and desirable features in refactoring tools. We also reported that programmers do not often rely on refactoring tools, but, at the same time, they are desiring number of unimplemented features.

Categories and Subject Descriptors D.2.3 [*[Software Engineering]*]: Coding Tools and Techniques

Keywords Refactoring tools, Programming Knowledge, Question-Answer Websites

1. Introduction

Stack Overflow is the most popular forum in the software development world, and it is a rapidly growing base of information about topics ranging from programming languages to algorithms. It contains over 1.5 million users and more than 4.5 million questions, and its data could be easily accessed through an open backup [1]. In Stack Overflow, users can post questions and obtain replies for it. The own community is responsible of assuring the quality of the questions and answers: if a question is good enough, users "upvote" the question; if not, they "downvote" it. Users that created those questions receive these "reputation points". That is, building reputation within the community is a key motivator for contributing to Stack Overflow, and contributors' reputations are quantified by scores based on their answers to questions posted on Stack Overflow. Thus, the reliability of the question/answer could be directly associated to the user who have created it.

Moreover, using Stack Overflow, we avoid common survey problems. In them, the subject may lie in a difficult questions, or may remember incorrectly about a long time question, or may not fully understand a question and then give a faulty response. Using Stack Overflow, we also have a higher number of subjects (more than 1,000 subjects), and the quality of the data might also be more reliable, because the subject will work hard to write a good question/answer. Otherwise, (s)he will be downvoted. With this data on hand, researchers can take the opportunity to mine countless questions about the state-of-use of the existing process, techniques and tools in the software engineering field.

Modern refactoring tools provide rich sets of functionality, which basically promise two benefits: First, refactoring tools promise to preserve functionality. Second, the tools promise to refactor faster than a programmer can refactor by hand. In that sense, the use of refactoring tools should increase the programmer productivity. Nonetheless, previous works have reported that programmers are underusing such tools [8, 9]. At the same time, researchers and tool vendors tend to propose new refactoring features. However, little is known about the features that application programmers are willing to use. Thus, Stack Overflow might prove a suitable candidate for exploring questions regarding refactoring tools, since building a refactoring tool that programmers want to use is hard.

In this sense, this paper presents an empirical study of questions and answers related to refactoring tools, created on Stack Overflow. We analyzed more than 1,400 messages – 324 questions and 1,115 answers to those questions – from more than 1,200 users, to uncover a number a issues regarding these tools. We translate this knowledge in to four research questions:

RQ1: What are the most desirable features in refactoring tools? Our qualitative analysis identified four groups of desirable features that are commonly discussed in Stack Overflow. These features include: (i) refactoring recommendations, (ii) refactoring for dynamic languages, (iii) refactoring for databases, and (iv) multi-language refactoring. We also noticed that programmers are willing to use these features, but, at the same time, some barriers are hindering the use.

RQ2: What are the barriers to adoption of refactoring tools? We have found out a number of barriers that are preventing the use of refactoring tools. Usability problems – such as difficult to select the code, and unknown error messages – are the most serious. Nonetheless, some users also reported a lack of trust in these tools. We then discuss some strategies as an attempt to fill this gap.

RQ3: How is interest in refactoring tools related to programmer expertise? As expected, questions come from novices (39.18% of it) and answers from hackers (40.44% of it). Moreover, the number of questions increases up to the reputation reaches the middle group (value 1K – 10K), and then starts to decrease. The answers to these questions, however, increase with the increase in reputation, up to the reputation reaches the last group ($>$ 100K).

WRT '13, October 27, 2013, Indianapolis, Indiana, USA.

http://dx.doi.org/10.1145/2541348.2541357

RQ4: Does interest in refactoring tools increase over the years? Not as expected. We have found out that the number of the questions are fairly the same when compared to the firsts months of Stack Overflow. However, the number of answers per questions can vary greatly, including some periods with 3 times more answers than others.

The rest of the paper is organized as follow: Section 2 presents some related work. Section 3 describes our study setup: the research questions (Section 3.1) and the data analyzed (Section 3.2). Next, in Section 4 we present our results. Finally, in Section 5, we conclude the work and present future directions.

2. Related Work

During the last decade, many studies have been proposed to understand the programmers' interest in refactoring tools [3, 6, 7, 9].

Erb [3] concluded that the existing refactoring tools are immature, when compared to others transformation tools that need to preserve behavior, such as optimizing compilers. This fact is directly associated with the usability problem that programmers are addressing. Mens [6] identified some features that programmers are willing to use. However, they also conclude that some tools do not guarantee that the program behaviour will be preserved, which can be a serious problem for its adoption. Moreover, Hill and colleagues [9] found that, although floss refactoring is more used than root canal refactoring, the overall use of refactoring tools is underused. In a similar study, the same author asked 16 object-oriented programming students whether they had used refactoring tools – only two said they had, reporting using them only 20% and 60% of the time [7].

Nonetheless, to the best of our knowledge, there are no study on the literature that try to understand the state-of-use of refactoring tools from a large body of subjects. Moreover, this work differs from the state of the art mainly because we identified new refactoring features that programmers are willing to use, as well as some barriers to adoption. Also we observed that this topic is receiving great attention during the years.

3. Study Setup

This section presents our study setup.

3.1 Research Questions

In this paper, we extracted Stack Overflow data to uncover questions about the state-of-use of refactoring tools. The goal of this work is to answer the following four research questions:

- **RQ1:** What are the most desirable features in refactoring tools?
- **RQ2:** What are the barriers to adoption of refactoring tools?
- **RQ3:** How is interest in refactoring tools related to programmer expertise?
- **RQ4:** Does interest in refactoring tools increase over the years?

To achieve our goal, we semi-automatically extracted questions related to the subject. We then investigate the selected data using qualitative methods. Next we consider the member reputation to observe its association with the our data. We also present some statistical correlations. Finally, we consider how this information can help the research and practice of software engineering to improve the use of refactoring tools. In the follow section, we explain issues that need to be addressed when considering explore such database.

3.2 The Data that We Analyzed

The work described in this paper is based on the data dump of the main Stack Overflow website, which consists of nearly 40GB of text file logs from July 31, 2008 to July 31, 2102 [1]. All the data is freely available under Creative Commons license. In this data, it is possible to find questions, answers, and the users that have created those data, among others informations. Within this time frame, over 3.4 million questions were created, and 6.8 million answers were provided. We parse this data line by line and extracted individual questions – and their answers – associated with questions about refactoring tools. This data is loaded into a single database table, indexed by user id, the question, the answers of the questions, and ordered by timestamp.

To filter only the related data, we firstly selected questions that had the words "refactoring" and "tool" in the body, or in the subject, or in the tag name. Thus, an example of valid question is one that might have the "refactoring" word in the subject, and "tool" word on the body. After this process, we observed that a high number of false positive questions remained in the data. A false positive question is, for example, a questions which was created by a user that is interested in a particular *refactoring* approach to use in his own *tool*. Thus, to eliminate the false positive questions, we manually analyzed each two times (one time by each author). At the end of the first extraction process, a total of 754 questions and 2,637 answers were found. After the second extraction phase, a total of 324 questions and 1,115 answers were selected. Hereafter in this study, we only considered the latter group.

4. Study Results

This section presents our findings. We organize the results in terms of the research questions.

4.1 RQ1: What are the most desirable features in refactoring tools?

Our qualitative analysis identified the following four groups of desirable features. It is worth to notice that some features are already implemented in wide-spread tools, and/or have already been proposed as a research topic. But, in a sort of programming languages, such as dynamic languages, these features are not fully available or, otherwise, they are available in only commercial tools. We then list these features in order of interest.

I. Refactoring for dynamic languages. Dynamic languages play an increasingly prominent role in modern software development. They are used in domains as diverse as web programming and scientific computing, for developing simple scripts as well as large applications. However, as pointed by Schäfer [11], specifying and implementing refactorings for dynamic languages is a challenging endeavor. We found out 78 questions (24% of the total) related to refactoring for for dynamic languages. These languages are PHP, JavaScript, Python, Ruby, Groovy, R, Perl, and Matlab. However, most of these questions are a variation of "*What refactoring tools do you use for language X?*", and the answers often suggest the use of well-known IDEs, such as Netbeans and Eclipse, such as the following question:

> **Question:** [...] So, is anyone aware of any php refactoring tools that allow you to determine state modifications and side effects of a particular piece of code (including new references to variables which may be stored as members and modified later). [...][1]

Nonetheless, tools to support for these languages are fair new, and often unable to ensure that program behavior will be preserved [4]. Moreover, often programmers that use these languages rely on command line editors, such as vi or emacs, which make the refactoring support much more difficult.

[1] http://stackoverflow.com/questions/581292/does-anyone-know-of-any-good-php-analysis-refactoring-tools

II. Refactoring recommendations. Refactoring tools often assume that a programmer already knows how to refactor and is familiar with the catalog of refactorings [5]. Nonetheless, it is not necessarily true. There are cases where the programmer is not able to identify a refactoring opportunity, whether by lack of knowledge in refactoring, or by lack of understanding the legacy code. We have found out that refactoring recommendations is one of the feature that most of Stack Overflow users desire (13% of them). It is also possible to group these features into groups: (i) highlight duplicate or dead code, (ii) highlight code smells (using metrics such cyclomatic complexity or LoC as back up) (iii) highlight optimization opportunities. Moreover, in minor scale, we also observe that well-known refactorings might be suggested, as pointed out by the following question.

> **Question:** [...] I would prefer a tool that just makes suggestions about possible refactorings: names the refactoring, optionally provides a short description of it (great for learning purposes), highlights the code section and lets me do the refactoring myself. [...] [2]

Nonetheless, part of this work was done by Weissgerber [12], but there is an ample room of improvements.

III. Refactoring for Databases. Refactoring tools for databases, in general, and for SQL, in particular, is fairly common in Stack Overflow. We found 12 questions related to the subject. The refactoring approaches mentioned are: (i) automatic simplification of SQL, (ii) refactoring table or views structure, and (iii) fixing sql vulnerabilities. We observe that, some of the desirable features are covered by well-known SQL tools. However, there are a lack of tools for refactoring object-relational mapping frameworks, as well as for NoSQL databases.

IV. Multi-language Refactoring. Most of the automated refactoring tools only work in a single programming language. However, an increasing number of programmers rely on polyglot programming as a way to build more complex systems. We have observed an increasing tendency in the amount of questions related to multi-language refactoring. We found a total of 9 questions, and 73% of them were created between the years of 2010 and 2012. Nonetheless, most of these questions were not answered appropriately. For example, the following comments are vague and unclear:

> **Question:** Is there a way to convert Groovy to Java automatically? [...] I have tried to manually convert some pieces to Java, it's been a pain. Are you aware of any tools or plugins that help with this conversion? [3]
>
> **Answer X:** I found and alternate solution, using Groovy++. It has almost all the advantages of Groovy, but with performance and strong typing from Java.
>
> **Answer Y:** My top tip is write lots of unit tests.
>
> **Answer Z:** I would focus on becoming more comfortable with Groovy instead of trying to convert the code to Java

One of the reason for these misunderstanding, is that refactoring for multi-language is much more difficult and would be even more onerous than refactorings in the same language [2]. Another important point is the existing gap in refactoring tools to support this activity. This fact poses a challenge not only for researchers in the academia, but to practitioners in the industry as well.

Covering all these different features in detail is out of the scope of this paper. However, a comprehensive and historical review of the subject has been conducted Prez and Crespo [10]. The authors model the current state of the art and identify open challenges, current trends and further research opportunities.

[2] http://stackoverflow.com/questions/785667/python-tool-that-suggests-refactorings

[3] http://stackoverflow.com/questions/5302103/is-there-a-way-to-convert-groovy-to-java-automatically

4.2 RQ2: What are the barriers to adoption of refactoring tools?

There is a strong belief in the academia that programmers are not using refactoring tools as much as they could [8, 9], and experiments were conducted as an attempt to understand this fact [7]. Some of these studies are based on personal experiences, surveys, and controlled experiments. Results of these studies show that error messages and usability problems are one of the key barriers in the adoption of refactoring tools. Nonetheless, identify barriers of use in popular forum could produce much more reliable results, since there are an absence of external factors that could biased the question/answer. In this section we identify and briefly describe two significant barriers encountered by Stack Overflow users.

I. Many refactoring tools are unknown or difficult to learn and to use. Many refactoring tools suffer from deep discoverability problems that make them less useful for general development. A significant number of questions are variation of "What refactoring tool do you use to do *X* in language *Z*". Since these questions are fairly basic, it would be reasonable to assume that only novice users are having problems in discover these tools. However, we noticed that not only unexperienced users – users with reputation between 1 and 1K – create such questions (33% of it), but even more experienced users – reputation higher than 10K – create such questions (9% of it).

Moreover, it is well-known that the usability of refactoring tools is far from optimal. This is critical, because even the best refactoring implementations are worthless if they are not used. Some of the most common usability problems are characterized in the following groups: (i) **Human Interaction** (e.g.: difficult to manually – with mouse and keyboard – select the code to be refactored); (ii) **Productivity** (e.g.: renaming locals variable can be quicker to do it by hand); (iii) **Learning Curve** (e.g.: Much time to learn); (iv) **Error messages** (e.g.: Lack of error messages or not understandable errors), and (v) **Resource Usage** (e.g.: A large memory footprint for a large code base).

To overcome these barriers, we suggest that refactoring tools should follow the list of five principles that characterize a good refactoring support [8]. Other studies have found that some refactoring tools tend to only partially adhere to these principles [3].

II. Lack of trust. Distrust of a tool can prevent the programmer from experimenting new refactorings. Moreover, some users believe that refactoring tools will introduce bad design choices in the code, as pointed out by a Stack Overflow user, *"Some times the tool will change the meaning of your code without you expecting it, due to bugs in the tool or use of reflection etc in your code base."*. Therefore, it is important to indicate what a refactoring will do before the programmer decides whether to apply it. We believe that preview hinting can play a major role in decreasing the resistance to applying refactorings by helping to build the programmer's trust for a tool. Finally, we have found out that the lack of trust could be so high that, even if the refactor will reduce the possibility of concurrent errors, such as atomicity violation, programmers tend to avoid it – as one user have pointed out: *"it is not good to decrease readability in favor of atomicity"*.

Finally, on top of all these barriers, programmers might not even be familiar with the catalog of refactorings, and therefore have great difficulties in discovering suitable refactorings.

4.3 RQ3: How is interest in refactoring tools related to programmer expertise?

In this section we intent to discover whether only hackers talk about refactoring tools, or otherwise if it is a wide topic with in the Stack Overflow community. To this end, Figure 1 shows the number of questions per user reputation.

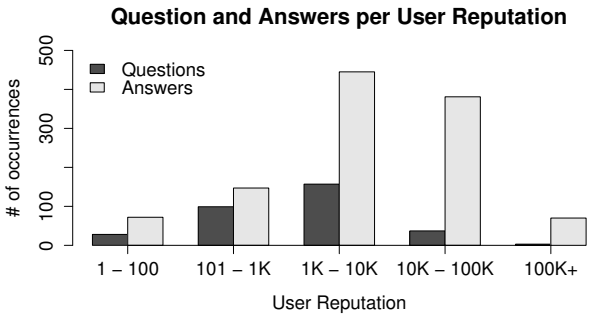

Figure 1. The number of question and answers per user reputation.

From the above figure we see that the number of questions increases up to the reputation reaches the middle group (1K – 10K), and then starts to decrease. On the other hand, the answers to these questions increase with the increment in reputation, up to the reputation reaches the last group (100K+), which is expected. While it is not surprising that novice users ask more questions, it is notable that members with reputation ranging from 1K to 10K provide the most questions and answers in Stack Overflow.

Moreover, we try to correlate the user's reputation with questions and answers created in Stack Overflow. To analyze the effect of the user expertise, we ran Pearson Correlation for *reputation x questions* and *reputation x answers*. For Pearson $|r| < 0.3$ indicates small correlation, $0.3 > |r| < 0.5$ indicates medium correlation, and $|r| > 0.5$ indicates strong correlation. First, the questions ration ($r = 0.12890$) produced a small correlation with the Reputation ratio, indicating that the availability of more expertise increases both the quantity and the quality of questions. Second, the answers ratio ($r = 0.12444$) produced also a small correlation, indicating that the availability of more expertise increases the number of answers per question.

4.4 RQ4: Does interest in refactoring tools increase over the years?

Finally, we have analyzed whether the interest in refactoring tools is increasing or decreasing over the years. To that end, Figure 2 shows the number of questions, and answers to these questions created during the analyzed period.

From the above figure we noticed that questions and answers had different behavior. On the one hand, the number of questions seem to be quite similar when compared with the firsts months of Stack Overflow, with few variations. The average number of questions per period is 19.06, with standard deviation of 6.50. On the other hand, the number of answers per questions can vary greatly, with some periods with 3 times more activities than others. The average number of answers per period is 66.76, with standard deviation of 33.29.

With a careful investigation, we realize that about 80% of the questions have less than 5 answers, which is quite low. Nonetheless, 4.32% of these questions have more than 10 answers, which is the reason behind this variation. Our qualitative analysis identified the following characteristics whereby these questions receive more attention than others: (i) question is simple and right on target;

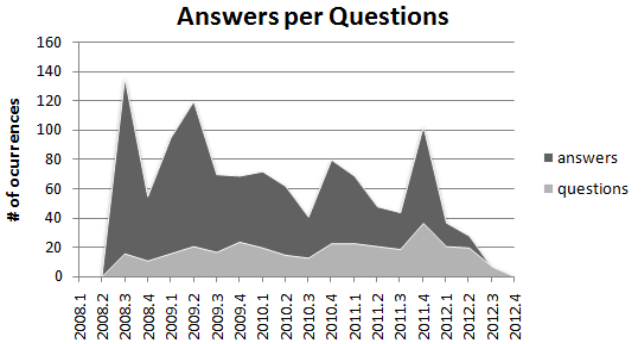

Figure 2. The number of questions x answers to these questions created during the period of July 31, 2008 and July 31, 2102.

and (ii) are fairly well written and relevant. Removing these biased questions, the standard deviation turns to be 12.89.

5. Conclusions and Further Work

In this paper, we investigated questions and answers in the Stack Overflow community, where most questions concern technical problems that software developers have. By comparing the list of questions and answers, we are able to discover what are the most desired features that programmers are willing to use and, at the same time, we also discover a few scenarios that appear to be the common cause for adoption barriers. Moreover, we also observed that user with all reputation are interested in such questions, but they are more likely to novices, and the interest in refactoring tools seems to be the same as was in the early days of Stack Overflow. Further work includes correlating this results with data from surveys, and analyze the user emotion through sentiment analysis. We leave these to future work.

References

[1] A. Bacchelli. Mining challenge 2013: Stack overflow. In *The 10th Working Conference on Mining Software Repositories*, page to appear, 2013.

[2] N. Chen and R. Johnson. Toward refactoring in a polyglot world: extending automated refactoring support across java and xml. In *Proceedings of the 2nd WRT*, 2008.

[3] S. Erb. A survey of software refactoring tools. Technical report, 2010.

[4] A. Feldthaus and A. Møller. Semi-automatic rename refactoring for JavaScript. In *Proc. ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, October 2013.

[5] M. Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ISBN 0-201-48567-2.

[6] T. Mens and T. Tourw. A survey of software refactoring. *IEEE Transactions on Software Engineering*, 30(2):126–139, February 2004.

[7] E. Murphy-hill. Improving refactoring with alternate program views., 2006.

[8] E. Murphy-Hill and A. P. Black. Refactoring tools: Fitness for purpose. *IEEE Softw.*, 25(5):38–44, Sept. 2008. ISSN 0740-7459.

[9] E. R. Murphy-Hill and A. P. Black. Why don't people use refactoring tools? In *Proceedings of the First WRT*, pages 60–61, 2007.

[10] J. Pérez and Y. Crespo. Perspectives on automated correction of bad smells. In *Proceedings of the ERCIM*, pages 99–108, 2009.

[11] M. Schäfer. Refactoring tools for dynamic languages. In *Proceedings of the Fifth WRT*, pages 59–62, 2012.

[12] P. Weigerber, B. Biegel, and S. Diehl. Making programmers aware of refactorings. In *Proceedings of the First WRT*, pages 58–59, 2007.