Reusing Code from StackOverflow: The Effect on Technical Debt

Nikolaos Nikolaidis, Georgios Digkas, Apostolos Ampatzoglou, Alexander Chatzigeorgiou
Department of Applied Informatics, University of Macedonia
Thessaloniki, Greece
it14189@uom.edu.gr, digasgeo@gmail.com, a.ampatzoglou@uom.edu.gr, achat@uom.gr

Abstract—**Software reuse is a well-established software engineering process that aims at improving development productivity. Although reuse can be performed in a systematic way (e.g., through product lines), in practice, reuse is performed in many cases opportunistically, i.e., copying small code chunks either from the web or in-house developed projects. Knowledge sharing communities and especially StackOverflow constitute the primary source of code-related information for amateur and professional software developers. Despite the obvious benefit of increased productivity, reuse can have a mixed effect on the quality of the resulting code depending on the properties of the reused solutions. An efficient concept for capturing a wide-range of internal software qualities is the metaphor of Technical Debt which expresses the impact of shortcuts in software development on its maintenance costs. In this paper, we present the results of an empirical study on the relation between the existence of reusing code retrieved from StackOverflow on the technical debt of the target system. In particular, we study several open-source projects and identify non-trivial pieces of code that exhibit a perfect or near-perfect match with code provided in the context of answers in StackOverflow. Then, we compare the technical debt density of the reused fragments, obtained as the ratio of inefficiencies identified by SonarQube over the lines of reused code, to the technical debt density of the target codebase. The results provide insights to the potential impact of small-scale code reuse on technical debt and highlight the benefits of assessing code quality before committing changes to a repository.**

Keywords-**Technical Debt; StackOverflow; Code Reuse; Software Quality;**

I. Introduction

Software developers reuse code fragments from libraries, other in-house projects, or the Web, to implement and maintain software products. Nowadays, software engineers are members of various knowledge sharing communities, which in the form of question and answer (Q&A) provide ready-to-use solutions for various development problems and facilitate the exchange of ideas among practitioners [1]. StackOverflow is the biggest Q&A site (i.e., approximately 15 billion answered questions) and high levels of loyalty by users, who suggest that more than 60% of them visit the website at least once per day [2]. There is also anecdotal evidence that a high reputation on StackOverflow can help developers obtain high-profile jobs. Another similar information source is GitHub, which is the most frequently used online repository for code development.

The popularity of such websites is so fast growing and their credibility is so high that the vast majority of software engineers reuse the fragments of code that they identify in StackOverflow or GitHub, without any filtering process and not being aware on how the reused code will affect their project. Nevertheless, according to Constantinou et al. [3] in some cases opportunistic reuse, might diminish the quality of the target application, e.g., in terms of coupling or cohesion. Possible interpretations of this phenomenon are: (a) the reusable asset is not identified in a systematic manner, or (b) reuse alternatives are not considered, or (c) the adaptation per se is not properly performed. Nevertheless, we cannot neglect that the majority of reuse activities take place just by copy-pasting code chunks from one system to the other.

Motivated by the increasing dependence of developers on StackOverflow this paper attempts to shed light on the possible implications of small-scale, opportunistic reuse of code retrieved from knowledge sharing communities, on the quality of the source code. With the term small-scale opportunistic reuse, we refer to the case when a software engineer, identifies a code fragment and clones/duplicates it in the source code of the target application. To achieve this goal, two aspects need to be captured:

- ***Identification of small-scale opportunistic reuse instances***. There are four categories of clone types depending on the level of similarity [4] and in order to identify them there are many state-of-the-art techniques and tools. Among those, in the context of this research, we used CPD (i.e., a token-based tool) [4], which identifies clones with modifications such as changed, added or deleted statements (i.e., 3rd-type clones). By using the tool, we can compare the source code of GitHub projects, with the retrieved StackOverflow code fragments. Since this process is by nature time consuming, we decided to specify the following two filters: (a) the GitHub projects should belong to the Apache Software Foundation (ASF) and (b) the source code files should have a reference to StackOverflow. StackOverflow has its own copyright rules: when someone wants to use an answer of a question he/she must also include the URL of that answer or question. But this phenomenon is very rare, approximately only 23% of the duplicated code fragments actually had a reference to StackOverflow [5]. At the same time, 75% of developers are not aware that they have to use a reference when they reuse code from StackOverflow [5].
- ***Measure Software Quality***. As a means to capture

the quality of source code we selected to employ the metaphor of Technical Debt (TD), which indicates the effort (in minutes) someone should spend on fixing all identified inefficiencies in a given piece of code. Technical Debt characterizes the gap between the current state of a code and a hypothetical optimal condition [6]. The gap includes known defects, code decay, outdated documentation, and features to be implemented [7]. To measure TD, there is a variety of available tools. A very popular platform that performs source code analysis to measure technical debt is SonarQube[1].

Given the aforementioned process, we have performed a case study for exploring the effect of reusing code from StackOverflow on the amount of TD at the source code level.

II. Related Work

As related work for this paper, we consider two categories of studies, depending on the aspect of research. On the one hand there are studies investigating the effect of reuse on quality and on the other hand studies related to the use of StackOverflow code fragments.

A. Effect of Reuse on Quality

In principle, reuse should enhance the quality of software, primarily because of the increased opportunity it provides for error discovery. Quoting Gaffney and Cruickshank [8], "*each time reusable code is used in a new application, an additional opportunity is provided for error discovery and removal*". Early empirical studies have provided evidence that systematic reuse can have a positive effect on software quality, especially leading to lower fault-density [9]–[11]. An overview of empirical studies on quality benefits from reusing software components is available in the study by Li et al. [12]. However, we have been able to identify only one study that focuses on the effect of reuse from StackOverflow, in which 22 Android projects have been investigated revealing that there is preliminary evidence of a negative effect on quality, in terms of bug fixing activity [13]. In our work we aim at extending the study of quality aspects that can be affected by code reuse, giving special emphasis on structural aspects that are covered by the technical debt concept and not only the fault density.

B. Reuse from/to knowledge sharing communities

Yang et al. [14] investigated how developers use snippets from StackOverflow in their projects. The authors cross checked 909k non-forked Python projects in GitHub with 1.9M Python snippets captured from StackOverflow using exact matching, matching on the tokens and near-duplication. The results show that exact duplication is very rare (less than 1%) and that the majority of duplicates are very small (typically 2 lines of code). The authors then focused on blocks of reused code that are somehow larger, as we also do in our study. A similar large-scale empirical study has been performed by Baltes and Diehl [15] aiming at analyzing the usage and attribution of non-trivial Java code snippets from StackOverflow in public GitHub projects. Up to 11.9% of the analyzed projects were found to contain a file with an explicit reference to StackOverflow. Studies have also investigated the opposite flow of information, i.e., from software projects to questions and answers in StackOverflow. Ragkhitwetsagul et al. [16] found clone pairs between 72,365 Java code snippets on StackOverflow and 111 open source projects in the curated Qualitas corpus. The study revealed that 153 clones have been copied from a Qualitas project to StackOverflow, with some of the reuse snippets in StackOverflow potentially violating the license of the original software.

III. Case Study Design

In this section we present the design of this case study. The study is designed according to the guidelines of Runeson et al. [17]. The goal is to determine whether reusing code fragments from StackOverflow has an impact on the quality of the host code. To this end, the following steps have been performed:

- Fragments of code were retrieved from StackOverflow
- Target projects/files from GitHub were selected
- Matching between StackOverflow snippets and target files was performed, looking for code duplicates
- For each match, the Technical Debt density of the reused fragment and the receiving code was measured

A. Objectives and Research Questions

Quality analysis tools such as SonarQube identify design and code inefficiencies such as code smells. Considering that 32% of software professionals are not aware of code smells [18] and acknowledging the importance of software quality for long-lived software projects, we aim at investigating whether the reused StackOverflow fragments of code have a different quality (in terms of Technical Debt) than the quality of the target project. More specifically, our study addresses the following research question (RQ): *Does the quality of StackOverflow code fragments differ from the quality of the projects in which the code is reused?*

B. Data Collection Process

To test whether StackOverflow fragments have a different TD density than the target projects, two different types of datasets are required: The first dataset should contain candidate fragments of code (i.e., successive instructions) from StackOverflow and the second dataset should contain the source code of candidate projects that reuse code from StackOverflow.

Our study focuses on Java StackOverflow fragments and GitHub projects developed in Java. Knowing that fragments with few lines of code (e.g., only 2) have been excluded from the dataset, in the sense that they might generate many false positives: similar sequences of instructions might have been written from the developers without any intent to reuse. To avoid such false positives, and based on experimentation, we excluded fragments with less than seven lines of code. Also, we have set a maximum limit, which was necessary, due to time and resource constraints. To come up with a

[1]https://www.sonarqube.org/

number for the upper limit, we adopted the "*rule of 30*", which states that if a code element consists of more than 30 sub-elements then the comprehension and maintenance of the element becomes intractable [12]. This rule applies to the lines of code in a method, in the methods that exist in a class and so on. Thus, we decided to set as upper limit for the lines in a fragment of code to 30 lines. To retrieve information from StackOverflow threads, a web scraper [19] has been implemented and the total number of fragments that we acquired was 126,340. The second dataset contains the candidate files against which the StackOverflow fragments will be tested for match. To limit the search space and reduce computation time, we retrieved projects that have already at least one reference to StackOverflow in their source files as comments.

To check for duplicates between the two datasets we used PMD[2], which is a widely used static code analyzer. PMD offers clone detection functionality with the Copy/Paste Detector (CPD) module, which can detect duplicated code in one or more files, in several programming languages, including Java. Each code fragment retrieved from StackOverflow was wrapped in a Java file and CPD was used to check for duplicates between each StackOverflow file and all candidate GitHub files. Once the files containing duplicates have been identified, we manually examined them for false positives. The automated search resulted in 47 files, while the manual inspection narrowed down the actual reuse cases to 34.

To enable the comparison between the quality of the reused code fragment and the quality of the reuse target project, we calculated their TD density using SonarQube. For the reused fragment, SonarQube parsed the artificial file in which the corresponding fragment was placed. For the host project, we identified the parent (previous) commit from the one in which the code fragment was reused, and measured the Technical Debt of the entire project for that commit. As already explained, to normalize the TD measurement against the size (of the project or the code fragment) we calculated the TD density as the ratio of the effort to remediate total TD over the non-commented lines of code.

C. Data Analysis

At the end of the process our dataset contained four variables: (a) project name, (b) TD density before the reuse activity, (c) TD density of the reused code, and (d) change proneness of the reused artifact. Analysis was performed with descriptive statistics, boxplots and hypothesis testing.

IV. Results

The search for StackOverflow-reused code in 260 files from GitHub projects that already had some references to StackOverflow in their comments has led to 37 code fragments of a substantial size (i.e. at least 7 lines of code) that have been reused. The quality of the reused fragments of code and especially the maintainability can be indirectly assessed from the frequency and extent of modification that the reused code has underwent. In the context of our study we have also investigated the change proneness of the reused code fragments, i.e., the times that the corresponding code was modified since the initial introduction from StackOverflow. It is noteworthy that only 8.5% (4 cases) of the examined fragments changed after they were inserted from StackOverflow. The extent of modification in these 4 cases ranged from 1 to 4 LOC (constituting 4.5% to 80% of the reused code).

With respect to the RQ concerning the quality of the reused code versus the quality of the host code, data analysis resulted in the findings of Table I. For each project we compare the TD density of that project file prior to the reuse (i.e., in the previous commit) to the TD density of the reused fragment. We note that initially, the dataset contained 37 rows, which after analysis of the outliers was reduced to 34. The Table contains only 10 cases of reuse, as for the rest 24 cases the TD density of the reused code was zero, and these cases are not listed for brevity.

TABLE I
TECHNICAL DEBT OF HOST PROJECT AND REUSED CODE

Project Name	**TD density of project prior to reuse**	**TD density of Reused Code**
brooklyn-server	0.968	0.294
pdfbox	0.297	0.714
storm	0.377	0.167
giraph	0.218	0.182
maven-shared	0.566	0.071
hbase	0.569	2.000
apex-core	0.541	1.000
sling-org-apache-sling-crankstart-launcher	0.000	1.000
velocity-tools	0.499	1.000
pdfbox	0.296	0.167

The distribution of TD density for the host project and the reused code is graphically depicted in the boxplot of Figure 1. It becomes evident that the reused code tends to be 'cleaner' than the host project, exhibiting a substantially lower TD density. This is also validated by a paired-samples two-tailed t-test on the two sets of data, which reveals a statistically significant difference ($p < 0.01$) between the TD density of new code (M=0.194, SD=0.441) and the TD density of the code in which the reused fragments are introduced (M=0.468, SD=0.034); t(33) = 3.216, p = 0.003. The results suggest that the reused code has fewer rule violations per line of code according to SonarQube.

It is also noteworthy that in the majority of the cases the reused code was absolutely 'TD-free' with no rule violations at all. Figure 2 illustrates the distribution of TD (measured by the effort to eliminate the identified rule violations) for the analyzed reused code fragments. Nevertheless, we have spotted limited cases, in which the TD density of the reused code is quite high. For instance, in the `hbase` project the inserted code (10 lines) suffers from two issues (namely: *"Change this "try" to a "try-with-resources" and Use or store the value*

[2]https://pmd.github.io/

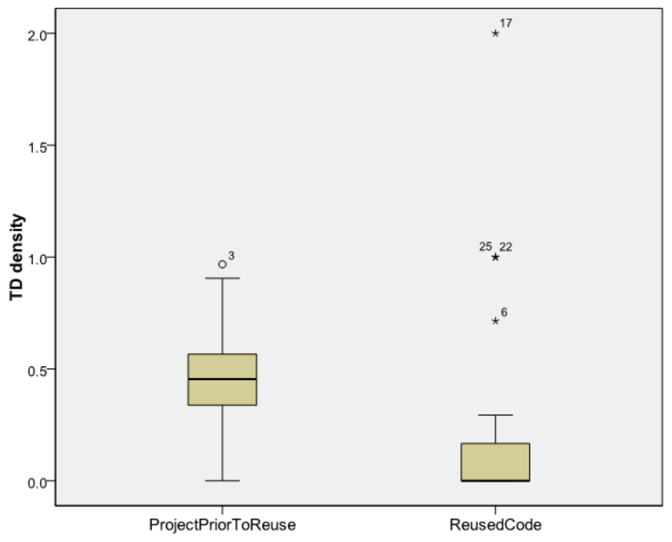

Fig. 1. Distribution of TD density for reused target projects and reused code.

returned from "readLine" instead of throwing it away") that cumulatively insert 20 minutes of TD (TDdensity=2.00). Such cases (as presented in the boxplot of Fig. 1) are quite rare and they could be considered as outliers. Nevertheless, we decided to retain them in the dataset.

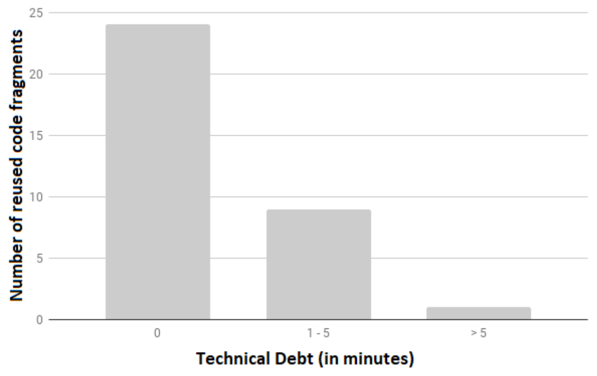

Fig. 2. Distribution of Technical Debt in reused code fragments.

V. Discussion

A. Interpretations of Results

The main finding of this study points out to the fact that code fragments in StackOverflow have low TD density and in many cases are completely free of design or code rule violations, as identified by SonarQube. Thus, reusing code from knowledge sharing communities not only boosts productivity (as a solution to a problem is rapidly found), but can also increase structural software quality. Although one could argue that this occurs because code snippets in StackOverflow are often small-sized and less complex, the retrieved code fragments in our study had size from 7 to 30 lines of code, indicating considerable functionality and complexity. Moreover, we do not rely on any absolute measure of TD, but on the more 'fair' measure of TD density, i.e., the effort to address rule violations per LOC.

The relatively high quality of code that can be retrieved from StackOverflow can probably be attributed to the motivation of posting tested and well-thought-out solutions on such forums. This is even more probable for questions/answers having a positive rating, rendering the included code more reliable. At the same time, code reused from StackOverflow is certainly domain-agnostic so as to be applicable in many problem spaces. This property relieves the code from external dependencies or ties to business logic, which are often the cause of shortcuts and inefficiencies in software development. Finally, it should be borne in mind that the host code of the projects underwent maintenance, often over a large number of revisions, whereas the reused code is 'fresh'. Considering that maintenance under time pressure is usually the root for quality decay, it is reasonable to observe improved quality for independent pieces of code that have not been maintened.

B. Implications for Practitioners and Researchers

The improved quality in terms of TD density for StackOverflow code fragments obviously implies that StackOverflow is a reliable source of code related information. Although blind reuse without checking for design, code or security issues is a bad practice, developers can assume that highly-rated answers are usually accompanied with code that has limited or no rule violations. Contemporary methodologies advocate the use of quality gates as a means of preventing the introduction of further issues into the codebase of a project. Thus, code reuse from StackOverflow combined with internal processes for quality control can ensure high quality software artifacts. It should be stressed that only in very few cases the reused code contained rule violations, which in the majority concern inappropriate exception handling. As exception handling is strongly dependent on the context in which code is introduced, these violations might be even less important. In other words, most probably the developer, who copied that piece of code, did not tailor it to his/her specific context. Under this perspective, almost all reused code fragments from StackOverflow were safe to use without introducing any TD.

For developers interested in posting code to Q&A forums such as StackOverflow, these findings underline the importance of thoroughly checking a solution before making it public. Although wisdom of the crowd can filter or promote code fragments by voting against or for them, code quality analysis (e.g. through a tool like SonarQube) before posting, can sustain the reputation of these communities. These findings, if validated by other studies, open up an interesting research field, in which the reuse from knowledge sharing communities can be studied more thoroughly for its effect on software quality. On a large scale, it can be empirically studied whether projects that systematically reuse code exhibit higher quality than more introvert projects. On a more fine-grained level research could focus on the qualities of the reused code fragments over time. As an example, it would be valuable to learn whether reused code fragments are easily maintainable on the long term and whether they are change or error prone.

VI. Threats To Validity

The number of StackOverflow code fragments for which a match in an open-source project was found, is relatively small. Thus, our main conclusion is subject to external validity threats, implying that we cannot argue that reused code from StackOverflow has always lower TD density, when compared to the quality of the host projects. To address this threat, we plan to replicate this study on a larger dataset of StackOverflow code fragments and a larger set of potential target projects.

One threat to the construct validity of the study stems from the fact that the host code refers to an evolving system, which might have underwent multiple revisions. Quality degradation usually occurs as software systems evolve (software aging). On the other hand, the reused code fragments are assessed at the time they are introduced, that is, prior to any maintenance. To mitigate this threat, further studies should look into the evolution of the reused code and their tendency to accrue TD.

Another threat to the validity of our conclusions with respect to the impact of the reused code on the quality of the host code, is related to the extent of reuse. In other words, even if the reused code fragments exhibit lower TD density, their effect on the host code is bounded by their contribution to the overall codebase. It would be irrational to claim that reusing 5-15 lines of code can have a significant effect on the overall quality. However, if reuse from StackOverflow is a continuous practice, as it is often the case, the cumulative effect can become significant. As for the previous threat, evolution studies can shed light into the effect of repeated reuse on quality.

Finally, it should be emphasized that SonarQube is one of the available tools for quantifying the principal of TD and consequently reflects SonarQubes ruleset to identify liabilities in the source code. Other TD tools may lead to different results as they rely on diverse techniques for TD measurement.

VII. Conclusion

Seeking answers by asking more experienced peers is a natural and wise approach for problem solving. In software development, it is commonplace, both for junior and senior developers, to consult StackOverflow on daily basis for code fragments that can solve technical issues that they are facing. As a result, reusing code from such knowledge sharing communities can increase productivity and enable knowledge transfer. One key question is whether this type of opportunistic reuse has an effect on the quality of the receiving codebase.

To this end, we have performed an empirical study to compare the Technical Debt density of StackOverflow code fragments against that of the projects in which these fragments have been reused in. By crosschecking 126,340 code fragments against 260 Java files from Apache projects we identified 34 reused snippets, ranging from 7 to 30 lines of code. The results revealed a statistically significant lower TD density for the StackOverflow code. Furthermore, in the majority of the cases the reused code was 'TD-free' containing no design or code rule violations. These findings confirm the high reputation that StackOverflow enjoys among developers, indicating that beyond an improvement of productivity, reuse of code can also enhance software quality.

Acknowledgment

Work reported in this paper has received funding from the European Union H2020 research and innovation programme under grant agreement No. 780572 (project: SDK4ED).

References

[1] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 188–195.

[2] "Stack Overflow Insights - Developer Hiring, Marketing, and User Research." [Online]. Available: https://insights.stackoverflow.com/

[3] E. Constantinou, A. Ampatzoglou, and I. Stamelos, "Quantifying Reuse in OSS: A Large-Scale Empirical Study," *Int. J. Open Source Softw. Process.*, vol. 5, no. 3, pp. 1–19, Jul. 2014.

[4] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of computer programming*, vol. 74, no. 7, pp. 470–495, 2009.

[5] S. Baltes, R. Kiefer, and S. Diehl, "Attribution required: Stack overflow code snippets in GitHub projects," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017, pp. 161–163.

[6] N. Brown, Y. Cai, Y. Guo, R. Kazman, M. Kim, P. Kruchten, E. Lim, A. MacCormack, R. Nord, I. Ozkaya, R. Sangwan, C. Seaman, K. Sullivan, and N. Zazworka, "Managing Technical Debt in Software-reliant Systems," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 47–52, event-place: Santa Fe, New Mexico, USA.

[7] E. Tom, A. Aurum, and R. Vidgen, "An Exploration of Technical Debt," *J. Syst. Softw.*, vol. 86, no. 6, pp. 1498–1516, Jun. 2013.

[8] J. E. Gaffney, Jr. and R. D. Cruickshank, "A General Economics Model of Software Reuse," in *Proceedings of the 14th International Conference on Software Engineering*, ser. ICSE '92. New York, NY, USA: ACM, 1992, pp. 327–337, event-place: Melbourne, Australia.

[9] W. B. Frakes and C. J. Fox, "Quality improvement using a software reuse failure modes model," *IEEE Transactions on Software Engineering*, vol. 22, no. 4, pp. 274–279, Apr. 1996.

[10] S. Haefliger, G. von Krogh, and S. Spaeth, "Code Reuse in Open Source Software," *Management Science*, vol. 54, no. 1, pp. 180–193, Jan. 2008.

[11] P. Mohagheghi and R. Conradi, "An Empirical Investigation of Software Reuse Benefits in a Large Telecom Product," *ACM Trans. Softw. Eng. Methodol.*, vol. 17, no. 3, pp. 13:1–13:31, Jun. 2008.

[12] J. Li, A. Gupta, J. Arvid, B. Borretzen, and R. Conradi, "The Empirical Studies on Quality Benefits of Reusing Software Components," in *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 2, Jul. 2007, pp. 399–402.

[13] R. Abdalkareem, E. Shihab, and J. Rilling, "On code reuse from StackOverflow: An exploratory study on Android apps," *Information and Software Technology*, vol. 88, pp. 148–158, 2017.

[14] D. Yang, P. Martins, V. Saini, and C. Lopes, "Stack overflow in github: any snippets there?" in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 280–290.

[15] S. Baltes and S. Diehl, "Usage and attribution of Stack Overflow code snippets in GitHub projects," *Empirical Software Engineering*, Oct. 2018.

[16] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, and R. Oliveto, "Toxic Code Snippets on Stack Overflow," *arXiv:1806.07659 [cs]*, Jun. 2018, arXiv: 1806.07659.

[17] P. Runeson, *Case study research in software engineering: guidelines and examples*. Hoboken, NJ: Wiley, 2012, oCLC: 868367873.

[18] A. Yamashita and L. Moonen, "Do developers care about code smells? An exploratory survey," in *2013 20th Working Conference on Reverse Engineering (WCRE)*. IEEE, 2013, pp. 242–251.

[19] E. Vargiu and M. Urru, "Exploiting web scraping in a collaborative filtering-based approach to web advertising." *Artif. Intell. Research*, vol. 2, no. 1, pp. 44–54, 2013.