How Do Code Refactoring Activities Impact Software Developers' Sentiments? – An Empirical Investigation into GitHub Commits

Navdeep Singh
National Institute of Technology
Jalandhar, India
E-mail: navdeep.singhmhy@gmail.com

Paramvir Singh
National Institute of Technology
Jalandhar, India
E-mail: singhpv@nitj.ac.in

Abstract— **Affective software engineering deals in exploring and understanding the developers' sentiments associated with specific software development tasks with the goal to unleash various task-to-sentiment relationships. This paper empirically investigates the impact of software code refactoring on the sentiments of developers in open source projects. We perform a comprehensive analysis of sentiments attached with 15 different refactoring activities across the evolution of 60 open source Java projects through mining relevant commit messages. We investigate 3,171 refactoring related commit messages (out of total 615,625 commit messages) representing 4,891 refactoring instances. The study outcome shows that in general software developers express more negative sentiments than positive sentiments, while performing refactoring tasks. It is also found that 5 out of 15 refactoring activities are mainly responsible for this observed outcome.**

Keywords— ***commit messages; refactoring activities; mining software repositories; affective software engineering.***

I. Introduction

The people-centric or individual-centric approach followed by many agile methodologies such as XP, Scrum, etc. puts a strong emphasis on continuously monitoring the mental health of developers during the course of a software development process. The sentiment information about the human resources with respect to various software development tasks is hence considered vital to the success of software projects. Affective software engineering is one of the recent research areas in software engineering, which deals with the study of developers' sentiments observed during the execution of various software development tasks [1].

There is only a limited number of previous studies that analyze the impact of software development tasks (e.g. refactoring, bug fixing, debugging, etc.) on developers' sentiments [2], [3]. Most of these studies rely on extracting the sentiments of developers using either *indirect* or *direct* methods. The indirect methods involve extracting emotions from software development artifacts such as commit messages, issue reports etc., whereas the direct methods depend on asking developers to self-assess their sentiments immediately after performing their assigned tasks.

Refactoring is a widely practiced developer-centric task that improves the software's internal design without changing its external behavior [4]; positively impacting various quality attributes such as maintainability, readability, reusability, etc. [5]. It is hence important to understand the developers' sentiments attached with the application of various types of refactoring in order to assist a software team in further boosting the developer productivity and software quality. In this work, we study the impact of implementing 15 widely used refactoring activities during the software development process on the sentiments of software developers. We use an indirect method to analyze developers' sentiments expressed in 3,171 refactoring-specific commit messages registered across multiple versions of 60 GitHub projects.

The major contributions of this work are: 1) We conduct an empirical investigation into software developers' sentiments extracted from refactoring-based commit messages recorded over multiple versions of a set of 60 GitHub projects (Section-IV); 2) We develop an open source tool, named *RefTypeExtractor*, for automatically linking commit messages to their respective refactoring techniques (Section III-B); 3) We also contribute an open dataset, named *SentiRef*, which stores the identified developers' sentiments linked to each of 3,171 commit messages and 15 refactoring activities (Section III-D).

II. Motivations and Research Questions

Software development is a tedious process that generally spans over years. Today's developers are expected to write quick code with minimum time available to come up with the best solution for a given problem. This situation often increases design viscosity, which leads to accumulating substantial technical debt as the software grows [6]. One popular preventive measure is to regularly refactor existing code to get rid of the debt-causing situations. From the developers' perspective, early refactoring efforts may highly reduce the amount of bug fixing required in future, and hence ideally they should relish practicing refactoring tasks [7]. These tasks however have many challenges, such as analyzing large code bases and their inter-component dependencies, co-ordination problems with team members, lack of tool support and harsh time constraints [8]. Therefore, these refactoring activities may have impact on software developers' sentiments, in turn affecting the human related project factors like productivity, creativity, job satisfaction, task quality, etc. [9], [10]. Silva et al. [11] presented evidence of interleaving refactoring activities with other programming tasks. Hence, it can also impact the quality of other tasks. So, an intriguing question is - *Does a refactoring task allocated during the implementation of a software feature following a strict deadline invoke positive or negative sentiments in the developer?*

The existing literature has only one instance [2] where the

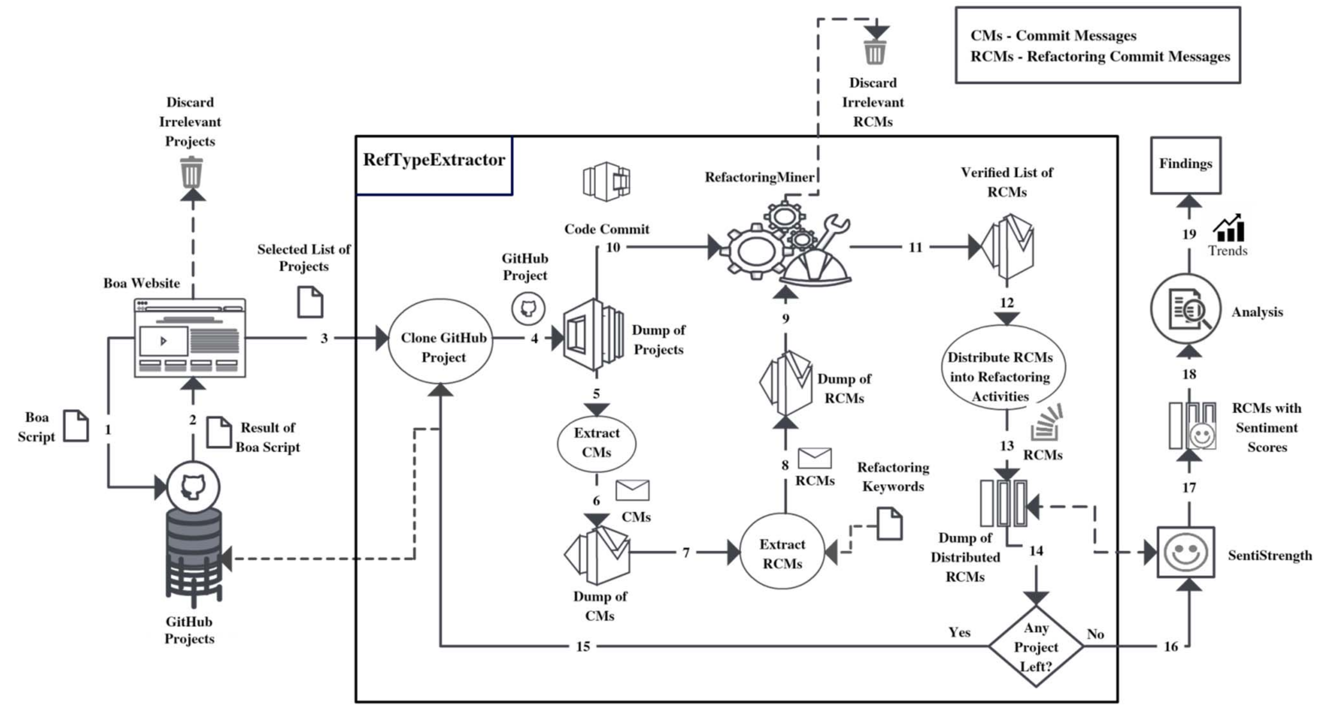

Fig. 1. Procedural steps of our research methodology.

impact of refactoring tasks (in general) on developers' emotions has been studied. Hence, it is important to understand the general sentiments prevailing across software developers implementing "specific" refactoring activities. Our motivations lead to defining the following two research questions:

RQ1. What are the general sentiments expressed by software developers while committing refactoring activities?

RQ1 does not differentiate between various refactoring activities, and aims to extract the general developers' sentiments linked to the refactoring tasks.

RQ2. Do developers express different levels (high, low, medium) and polarities (positive, negative, and neutral) of sentiments in committing different refactoring activities?

By understanding the sentiments linked to different types of refactoring activities, RQ2 aims to identify refactoring activities invoking predominantly positive or negative developers' sentiments. Such an information can guide a software team toward preventing, replacing, or reprioritizing certain kinds of refactoring activities that invoke negative sentiments in developers.

III. Study Design and Methodology

The overall experimental study design comprises four major elements: sample projects (Section III-A), *RefTypeExtractor* tool (Section III-B), sentiment analysis (Section III-C), and *SentiRef* dataset (Section III-D). The procedural steps of our research methodology (as shown in Fig. 1) are explained under the following subsections.

A. Sample Projects

For choosing the required sample projects (also referred as repositories) for our work, we accessed the largest GitHub dataset (*full - 100%* as mentioned on the Boa website [12] till September 2015) from Boa repository mining tool, which consists of 7,830,023 GitHub projects. Boa has been widely used by existing studies [2], [10] related to mining software repositories. The main reason for using Boa is its user friendly web-based interface and scripting support.

We wrote Boa script[1] to extract the list of top 150 GitHub projects containing the highest number of commit messages. We chose projects with the highest number of commit messages because our focus is to identify sentiments expressed in a large set of refactoring related commit messages. Due to this work's dependency on RefactoringMiner tool (Section III-B), we needed to target only the projects written in Java language for further analysis. We hence checked the project availability and the programming language related to each of the 150 projects (as fetched from Boa dataset) by carrying out a manual comparison with the current GitHub records. The cloning of these projects also enabled the extraction of additional commit messages (relevant to selected projects) recorded between October 2015 to October 2016. A total of 60 Java projects[2] were identified after eliminating unavailable projects and the projects written in other languages. The selected projects vary in terms of size, contributors and application domains.

After downloading these projects, Total Commit Messages (TCMs) extracted from all 60 projects count to 615,625. Further, a total of 3,171 Refactoring related Commit Messages (RCMs) are identified from the set of TCMs using *RefTypeExtractor* (Section III-B).

B. RefTypeExtractor

In this study, we investigate developers' sentiments found in RCMs related to specific refactoring activities. Hence,

[1]https://github.com/navdeep08/SentiRef/blob/master/Boa Script.docx
[2]https://github.com/navdeep08/SentiRef/blob/master/Top 60 projects.xlsx

TABLE I. LIST OF 15 REFACTORING ACTIVITIES AND NUMBER OF INSTANCES FOUND IN 60 PROJECTS

SNo	Refactoring Activity	# RIs	# Projects
1	*Extract Method* (EM)	1,094	60
2	*Extract Interface* (EI)	86	32
3	*Extract Superclass* (ES)	124	40
4	*Rename Method* (RM)	950	60
5	*Rename Class* (RC)	589	58
6	*Inline Method* (IM)	287	54
7	*Push Down Attribute* (PDA)	37	18
8	*Push Down Method* (PDM)	46	27
9	*Pull Up Attribute* (PUA)	72	34
10	*Pull Up Method* (PUM)	107	38
11	*Extract and Move Method* (EMM)	59	27
12	*Move Attribute* (MA)	317	56
13	*Move Method* (MM)	350	58
14	*Move Class* (MC)	676	51
15	*Move Class Folder* (MCF)	97	31
Total		4,891	644

```
refactoring[0] experience[0] management[0] slightly[0] to[0] be[0]
more[0] friendly[1] to[0] writing[0] tests[0] [[Sentence=-1,2=word max,
1-5]] implementing[0] some[0] tests[0] for[0] thaumaturgy[0]
[[Sentence=-1,1=word max, 1-5]][[[2,-1 max of sentences]]]
```

Fig. 2. Output of SentiStrength for RCM.

tagging each RCM with its respective one or more refactoring activities is necessary. To perform this task, we developed a tool named *RefTypeExtractor*, which distributes RCMs among selected refactoring activities with the help of RefactoringMiner tool [11].

RefactoringMiner implements the UMLDiff algorithm, along with a set of rules, to identify instances corresponding to various refactoring activities performed between two successive code commits. RefactoringMiner is known to have the highest recall and precision rates (0.93 and 0.98, respectively) among all currently available refactoring detection tools [11]. Table I shows a list of selected 15 refactoring activities (supported by RefactoringMiner) along with their respective number of instances identified in 60 projects under study.

RefTypeExtractor works by taking a list of input projects from a text file, which comprises the details of all 60 GitHub projects. Each project is mentioned with its web URL and name. For example, the details of *Maven* project are written as: "*https://github.com/apache/maven.git maven*". Thereafter for each input project, *RefTypeExtractor* works as follows.

RefTypeExtractor automatically clones the input project using its web URL given in the text file. It then extracts TCMs of input project using *Git* commands. Parnin et al. [13] observed that commits performed with the main motivation other than refactoring (but still incorporating at least some refactoring) do not usually contain refactoring-related keywords in their respective commit messages. We aim to analyze only those commits, which not only represent code refactorings (with major or minor focus), but also contain at least some evidence about the performed refactoring tasks in their texts. Hence, in order to recognize the relevant RCMs, we prepare an initial list of 98 refactoring related keywords, which includes various synonyms for 'refactoring', and names of refactoring activities and code smells (Martin's catalog [4]). *RefTypeExtractor* mines all RCMs (total 8,807) from the extracted TCMs using these keywords. Such keywords for which no refactoring related commit was identified were discarded from the initial list of 98 keywords. The resultant list contained 20 keywords.

Henceforth, *RefTypeExtractor* parses the code to identify various code refactoring instances corresponding to each RCM using RefactoringMiner. The output of RefactoringMiner is used to validate our list of RCMs. A total of 63.9% false positives (5,636 out of 8,807) were found after this step, which were eliminated from the set of RCMs. We also discarded the unproductive keywords from the initial search keyword list in order to enable an accurate future replication of this process. Both the initial (98) and updated keyword (14) lists are openly available[3]. The remaining 3,171 RCMs are categorized into their respective refactoring activities. Commit messages related to each refactoring activity are stored in separate files.

Further a manual validation is conducted on two projects dvn/svn-import-test4 and nickboldt/temp-jbosstools-javaee to check the accuracy of *RefTypeExtractor*. Each step of the *RefTypeExtractor* is verified separately. Project cloning, extraction of commit messages and RCMs using keyword search approach, are found to be 100% accurate. The next step involves RefactoringMiner whose accuracy is already proven [11]. In order to strengthen our validation, we still conducted a manual examination of code commits related to the two selected projects. The results indicate that 73 RIs (linked to 46 RCMs) identified by RefactoringMiner do not contain any false positives. This manual effort should increase confidence in using *RefTypeExtractor* for the extraction of RCMs in future studies.

C. Sentiment Analysis

Sentiment analysis is the process of identifying opinions or sentiments from a piece of text. Various tools are available for performing sentiment analysis such as NLTK, SentiStrength [14], and Alchemy. SentiStrength not only detects the polarity of sentiments expressed in a text but also detects the strength of sentiments. The first reason for choosing this tool is its decent accuracy for short texts [14] (such as twitter data). As commit messages are also short and small, therefore, this tool was found to be appropriate for this study. Another reason is that it has been widely used in studies ([2], [9], and [10]) related to detecting sentiments in software engineering artifacts.

In this study, we use SentiStrength to analyze sentiments associated with commit messages of 15 refactoring activities. For this, output files generated by *RefTypeExtractor* containing RCMs belonging to each refactoring activity are passed to SentiStrength for performing the required sentiment analysis. SentiStrength then calculates sentiment scores for each commit message by first assigning individual scores to each relevant constituent word followed by compiling an overall score for the commit message as shown in Fig. 2.

2) Tuning of SentiStrength: Jongeling et al. [15] studied the impact of using sentiment analysis tools in software engineering studies. They observed that existing sentiment analysis tools (e.g. SentiStrength, NLTK, etc.) are trained on social web text data. Therefore, software engineering artifacts can be misinterpreted by these sentiment analysis tools due to the presence of technical keywords, code, and queries.

This misinterpretation can be reduced by tuning existing

[3]https://github.com/navdeep08/SentiRef

TABLE II. SENTIMENT ANALYSIS METRICS

Mean positive sentiment score (*Pos(C)*)	**Mean negative sentiment score (*Neg(C)*)**	**Mean cumulative sentiment score (*T(C)*)**
$Pos(C) = \frac{\sum_{c \in C_+} \rho_c}{\vert C_+ \vert}$	$Neg(C) = \frac{\sum_{c \in C_-} \eta_c}{\vert C_- \vert}$	$T(c) = \rho'_c + \eta'_c$ where, $\rho'_c = \begin{cases} \rho_c, & \text{if } \rho_c > 1 \\ 0, & \text{otherwise} \end{cases}$ $\eta'_c = \begin{cases} \eta_c, & \text{if } \eta_c < -1 \\ 0, & \text{otherwise} \end{cases}$

sentiment analysis tools for software engineering context.

SentiStrength provides the facility to adjust emotional weights of keywords. Hence, we tune SentiStrength by assigning a neutral score to 49 misinterpreted keywords ('super', 'block', 'garbage', etc.) as mentioned in previous study [2]. After tuning SentiStrength, they found 26% increase in precision on the data evaluated by humans. We further examine and update the emotional weights of 7 additional misinterpreted keywords ('drag', 'dump', 'elimination', 'excel', 'overflow', 'subversion' and 'subtract') in the EmotionLookUpTable[4] file of SentiStrength. To verify the impact of our second tuning, we randomly selected a sample of 100 commit messages and manually analyzed them. We find a 4% increase in the precision of SentiStrength after the second tuning. Thus, we use a substantially tuned version of SentiStrength for identifying sentiments associated with our selected set of RCMs. We also recommend researchers to use this tuned version of SentiStrength for studies targeting software engineering domain.

1) Sentiment Analysis Metrics: For each commit message (c), SentiStrength assigns two output values (ρ, η), where ρ and η represent positive (1 to 5) and negative (-1 to -5) scores, respectively. The output of SentiStrength shows that a given text can have both negative and positive sentiments at the same time. The text with <+1,-1> emotional score is considered emotionally neutral. Sentiment analysis metrics used for answering the research questions defined in Section II are presented in Table II. From a set C of commit messages, we extract a new subset C′ defind as follows:

$$C' = \{c \text{ , where } c \in C, \text{ either } \rho_c > +1 \text{ or } \eta_c < -1\}$$

where, c denotes a commit message; C′ contains commit messages with atleast one non neutral sentiment score; ρ'_c denotes positive cumulative sentiment score of c; η'_c denotes negative cumulative sentiment score of c; and T(c) denotes total cumulative score of c.

D. *SentiRef*[5]

We contribute a dataset *SentiRef* that stores the sentiments associated with selected 15 refactoring activities. *SentiRef* is an open dataset that currently contains the sentiment information of 3,171 RCMs from the selected projects along with corresponding 4,891 refactoring instances identified using *RefTypeExtractor.*

1) Building SentiRef Dataset: Firstly, *RefTypeExtractor* (Section III-B) detects and distributes RCMs into their respective refactoring activities. Secondly, the tuned version of SentiStrength (Section III-C) assigns the sentiment scores to RCMs. The structure of *SentiRef* is shown in Table III.

2) How to Use and Contribute[6]*:* Researchers can openly use and extend the functionalities of *RefTypeExtractor* through our GitHub repository. They can also use the *SentiRef* dataset in their future studies related to refactoring and sentiment analysis. Researchers can also extend this dataset by submitting relevant data from other projects.

TABLE III. *SENTIREF* DATASET EXAMPLE – TOMCAT-70 GITHUB PROJECT FOR ES REFACTORING ACTIVITIES

Ref. Type	**Commit ID**	**Sentiment Score**	
		Pos(c)	***Neg(c)***
ES	5411f979ff0e122efe9fd6e09sfc200c5b5bb89cb	1	-1
	2185fbba9dbd4bb2aff8528c31f357029aab7bf8	2	-1
	df0b3fdfb74f13eb29ed3a2596b545e6afddc3a41	1	-1

TABLE IV. SENTIMENT PERCENTAGE OF RCMS ACROSS SELECTED PROJECTS

Sentiment Score	**Sentiment Percentage**
$\rho_c = \eta_c$	18.31%
$\rho_c > \eta_c$	38.00%
$\rho_c < \eta_c$	43.69%

TABLE V. MWW TEST BETWEEN POS(C) AND NEG(C) OF 60 PROJECTS

MWW Test	***p*-value**	**Significant?**
One Tail	0.00	Yes
Two Tail	0.00	Yes

IV. ANALYSIS AND FINDINGS

A. *General Sentiments in Refactoring Commit Logs (RQ1)*

To answer RQ1, we identify RCMs in the input projects using *RefTypeExtractor*, followed by the detection of developers' sentiments using SentiStrength. After extracting sentiments from all 3,171 RCMs identified, we remove 2,363 neutral (<1,-1>) RCMs to get 809 RCMs for further analysis. The results of sentiment percentage, which is the percentage of positive, negative and equal sentiment scores across selected RCMs, are shown in Table IV. We notice that a large number of RCMs (43.69%) record high negative sentiment scores as compared to positive scores. Further, we apply analysis metrics (Table II), which result in mean positive and mean negative sentiment scores of 1.6 and -1.7, respectively. This shows that, in general, developers express more negative sentiments than positive sentiments while committing refactoring tasks.

Furthermore, to verify the statistical significance of our observations, we conduct a Mann-Whitney-Wilcoxon (MWW) test [16] (with α=0.05) between positive and negative sentiment scores of previously selected RCMs. The MWW test is a non-parametric test, which does not require the data to have a normal distribution. Since the data in our work do not conform to a normal distribution, this particular test suits well for our purpose. A comparison of resulting *p*-values (shown in Table V) with α value suggests the statistical significance of our observations for various refactoring tasks. From the statistical significance of our observations and the values of metrics (*Pos(C)* and *Neg(C)*), we conclude that developers express high negative sentiments as compared to positive

[4]https://github.com/navdeep08/SentiRef/blob/master/EmotionLookupTable.txt
[5]https://github.com/navdeep08/SentiRef

[6]https://github.com/navdeep08

sentiments while performing refactoring tasks.

It is noteworthy that Islam and Zibran [2] in their work conclude that developers express significantly higher positive sentiments than negative sentiments when performing refactoring tasks. This difference in results of these two studies (our work and [2]) could be attributed to a number of factors. The major factors could be the different sample projects used and the second custom tuning performed on SentiStrength in our study. Moreover, they consider 50 projects written in multiple programming languages, whereas we conduct our study on 60 Java projects. We apply a number of stronger verification criteria on various experimental steps. For instance, they used only two root keywords (*refactor* and *code clean*) to distinguish refactoring related commit messages, while we used a comprehensive list of 14 keywords covering a wider spectrum of refactoring related terms. We also detect actual refactoring activities practiced in code commits using RefactoringMiner to support and verify the commit messages based text analysis. Hence, we believe that the results of this study are much more reliable than the ones presented in [2].

Answer to RQ 1: The observations and statistical analysis show that, in general (without distinguishing different refactoring activities), software developers express higher negative sentiments than positive sentiments when performing refactoring tasks. The main reason behind the negative sentiments can be attributed to the overhead of manually practicing refactoring activities (86% on average) on large and complex code bases, instead of using automated refactoring tools [8]. Although developers want to use automated refactoring tools, they are generally discouraged by tools' unpredictable outputs, confusing names of refactoring activities, and overhead of learning [17].

B. Variations of sentiment polarities in different refactoring activities (RQ2)

RQ2 extends the exploration of RQ1 by investigating sentiments associated with selected refactoring activities. To answer RQ2, we distribute RCMs of 60 projects into categories representing 15 refactoring activities. We reiterate that a developer can perform more than one refactoring activity in a single commit. Therefore, one commit message may be accounted for more than one refactoring activity.

Table VI presents the mean positive and mean negative sentiment scores of each refactoring activity across 60 projects. These observed values show that 2 out of 15 refactoring activities (ES and MCF) have equal mean positive and mean negative scores. 10 out of 15 refactoring activities (EMM, EI, EM, MA, MC, MM, PUA, PUM, RC and RM) record higher mean negative scores than mean positive scores. Rest 3 activities (IM, PDA and PDM) show higher mean positive scores than mean negative scores.

To check the statistical significance of our observations, we conduct MWW (Table VI) test between positive and negative scores of each refactoring activity. The outcome of MWW test shows that the observations of 10 (EMM, EI, EM, ES, IM, MCF, PUM, PDA, PDM, and RM) out of 15 refactoring activities are not statistically significant. Hence, it can be inferred that the developers express different polarities of positive and negative sentiments while committing remaining 5 (MA, MC, MM, PUA, and RC) refactoring activities.

TABLE VI. RESULT OF MWW TEST BETWEEN POS(C) AND NEG(C) SENTIMENT SCORES OF 15 REFACTORING ACTIVITIES

Refactoring Activity	**Pos(C)**	**Neg(C)**	***p*-Value**		**Significant?**
			One Tail	***Two Tail***	***One Tail/Two Tail***
EMM	1.54	-1.89	0.13	0.25	No/No
EI	1.54	-1.75	0.22	0.45	No/No
EM	1.68	-1.77	0.05	0.11	No/No
ES	1.74	-1.74	0.41	0.81	No/No
IM	1.71	-1.57	0.10	0.20	No/No
MA	**1.53**	**-1.93**	**0.00**	**0.00**	**Yes/Yes**
MCF	1.60	-1.60	0.46	0.93	No/No
MC	**1.56**	**-1.75**	**0.01**	**0.02**	**Yes/Yes**
MM	**1.59**	**-1.80**	**0.04**	0.07	**Yes**/No
PUA	**1.16**	**-2.32**	**0.00**	**0.00**	**Yes/Yes**
PUM	1.59	-1.77	0.13	0.27	No/No
PDA	1.83	-1.33	0.07	0.15	No/No
PDM	1.82	-1.76	0.37	0.73	No/No
RC	**1.56**	**-1.85**	**0.00**	**0.00**	**Yes/Yes**
RM	1.72	-1.74	0.40	0.81	No/No

Answer to RQ2: Based on the observations of Table VI, it is found that developers express more negative sentiments than positive sentiments in case of 5 (MA, MC, MM, PUA, and RC) out of 15 refactoring activities. Hence the conclusion drawn in response to RQ1 is mainly due to the negative sentiments of these 5 refactoring activities. While for remaining 10 refactoring activities, there is no significant difference between the positive and negative sentiments.

Generally, Version Control Systems (VCSs) are sensitive to selective refactoring activities, especially those belonging to move and rename refactoring families, such as MC, RC, etc. [8]. This discourages software developers to perform, merge and integrate refactoring-based code changes on VCSs. This might possibly be the reason for high negative sentiments associated with all move and rename based refactoring activities (Table VI), which dominate the set of 5 refactoring activities (4 out of 5) showing more negative sentiments than positive sentiments.

V. DISCUSSION

Even though our findings indicate that the refactoring tasks trigger negative sentiments in software developers, we cannot ignore refactoring as it forms a major part of the overall software development process (almost 10% of developers' work) [8]. The reason for developers' negative sentiments in refactoring tasks could be due to the complexity of refactoring activities, lack of dedicated tools [17] and risks of introducing errors [8]. There is a need for efficiently handling different refactoring activities. This can be achieved by improving the tool support for refactoring activities that developers perceive as hard (negative sentiments). Project managers can also improve their refactoring strategies by replacing or rescheduling refactoring activities that invoke negative sentiments in developers.

Finally, we believe there could be a number of factors working behind the sentiment scores observed in this study, such as implementation effort required for a specific activity, motivation [11] behind performing an activity, past experience with a specific activity, indirect factors like dragging emotions from another non-refactoring activity to the current refactoring activity, project domain, etc. However, our work limits to finding interesting sentiment-to-refactoring relationships which form the basis of an exploration into investigating the most likely causes for such relationships.

VI. Threats to Validity

The internal validity of this work depends on the accuracy of open source tools RefactoringMiner and SentiStrength. RefactoringMiner is known to have the highest recall and precision rates (0.93 and 0.98, respectively) among all currently available refactoring detection tools [11]. Similarly, SentiStrength has been reported suitable for extracting sentiments from commit messages [9]. It has a precision of 60.7% and 64.3% for positive and negative texts, respectively [2]. We further tune SentiStrength (Section III-C-1) with respect to software engineering domain, which leads to increased precision.

The selection of 60 Java projects for examining the task-sentiment relationships can be questioned. Note that these projects are the top 60 open source Java projects (prioritized on the basis of number of commit messages) hosted on GitHub, which is a large sample of data for dependable analysis. Also, this work is based on open source repositories extracted from a web-based VCS - GitHub. Hence, the findings of this work are context-dependent and can be affected by the type of VCS (such as SVN, Mercurial etc.) used [8].

This research is the first attempt to investigate the sentiments associated with different refactoring activities; hence no findings are available to compare our findings. We make several efforts to enhance the reliability and reproducibility of our study. *RefTypeExtractor*, RefactoringMiner and SentiStrength are freely available tools along with the sample projects studied in this work, which are freely accessible through GitHub.

VII. Related Work

The interest in leveraging information pertaining to developers' sentiments associated with the software development process for the betterment of the overall project, has increased over the last five years. Researchers are hence focusing hard on utilizing various direct and indirect methods to identify the developers' emotional states expressed during various software development tasks.

As such, Khan et al. [3] analyze the impact of emotions on the debugging performance of software developers. Similar to Khan's work, Lesiuk [18] also analyze the impact of emotions generated through music on the design performance of software developers.

The research studies mentioned above, focused on capturing emotions using direct methods, which is a difficult task when dealing with geographically distributed project teams. Thus, to overcome this problem, software artifacts like commit messages, mailing conversations, etc. are exploited for extracting emotional information [9], [19].

Murgia et al. [19] observed that issue reports do express emotional information, and that the issue reports with associated positive sentiments take less time to be resolved. Guzman et al. [9] investigate the relationship of developers' emotions with factors like programming languages, different time zones, etc. Similarly, Sinha et al. [10] correlated the emotional variations with number of changed files and different days of the week.

VIII. Conclusions and Future Work

Understanding the sentiments of software developers when performing various development tasks is vital to the successful completion of a project. This paper presents an empirical analysis on the impact of committing 15 refactoring activities on developers' sentiments.

In our study, we found that software developers generally express more negative sentiments while practicing 15 refactoring activities. Further, MA, MC, MM, PUA and RC refactoring activities form the main reason behind high negative sentiment scores for refactoring in general. The outcome of this study informs software teams about refactoring activities invoking positive or negative sentiments in developers, and enables them to take necessary decisions regarding the selection of appropriate refactoring activities.

In future, we plan to replicate this study on a much larger set of projects, including projects written in languages other than Java. We also aim to extend our investigation beyond the 15 refactoring activities considered in this work.

References

[1] D. Graziotin, F. Fagerholm, X. Wang and P. Abrahamsson, "Unhappy developers: Bad for themselves, bad for process, and bad for software product," In Proc. of the Int. Conf. on SE Companion, 2017, in press.

[2] M. R. Islam and M. F. Zibran, "Towards understanding and exploiting developers' emotional variations in software engineering," In Int. Conf. SERA, pp. 185–192. 2016.

[3] I. A. Khan, W. P Brinkman and R. M. Hierons, "Do moods affect programmers' debug performance?," Cognition, Technology & Work, vol. 13, no. 4, pp. 245-258, 2010.

[4] M. Fowler, K. Beck, W. Brant, W. Opdyke, and D. Roberts, Refactoring: Improving the Design of Existing Code, Add.-Wes., 1999.

[5] T. Mens, and T. Tourwé, "A survey of software refactoring," IEEE Transactions on software engineering, vol. 30, no. 2, pp. 126-139, 2004.

[6] Yli-Huumo, Jesse, A. Maglyas, and K. Smolander, "How do software development teams manage technical debt?–An empirical study," Journal of Systems and Software, vol. 120, pp. 195-218, 2016.

[7] M. Leppänen, S. Mäkinen, S. Lahtinen, O. Sievi-Korte, A. P. Tuovinen and T. Männistö, "Refactoring-a Shot in the Dark?," In IEEE Software, vol. 32, no. 6, pp. 62-70, Nov.-Dec. 2015.

[8] M. Kim, T. Zimmermann, and N. Nagappan, "A Field Study of Refactoring Challenges and Benefits," In Proc. of SIGSOFT, 2012.

[9] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in GitHub: an empirical study," In Proc. of Int. Conf. on MSR, pp. 352–355. 2014.

[10] V. Sinha, A. Lazar, and B. Sharif, "Analyzing developer sentiment in commit logs," In Proc. of International Conference on Mining Software Repositories. 2016.

[11] D. Silva, N. Tsantalis, and M. T. Valente, "Why we refactor? confessions of github contributors," In Proc. of Int. Symposium on Foundations of Software Engineering, 2016.

[12] H. Rajan, T. N. Nguyen, R. Dyer, and H. A. Nguyen, "Boa:Mining Tool" [Online]. Available: http://boa.cs.iastate.edu/.

[13] C. Parnin, and C. Görg, "Improving change descriptions with change contexts," In Proceedings of Int. working conf. on MSR, 2008.

[14] M. Thelwall, K. Buckley, and and G. Paltoglou, "Sentiment Strength Detection for Social Web," Journal of Ass. Soc. Inf. Sci. Tech., vol. 14, no. 4, pp. 90–103, 2012.

[15] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik, "On negative results when using sentiment analysis tools for software engineering research," Empir. Softw. Eng., pp. 1-42, 2017.

[16] R. Zhang, K. Aloisio, and N. J. Horton, "The Statistical Sleuth in R : Chapter 1," pp. 1–10, 2012.

[17] M. Vakilian, N. Chen, S. Negara, B. A. Rajkumar, B. P. Bailey, and R. E. Johnson, "Use, disuse, and misuse of automated refactorings," In Proc. of Int. Conference of Software Engineering, pp. 233–243, 2012.

[18] T. Lesiuk, "The effect of music listening on work performance," Psychology of music, vol. 33, no. 2, pp. 173-191, 2005.

[19] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," In Proc. of Work. Conf. MSR 2014, pp. 262–271, 2014.