Understanding the Effect of Developer Sentiment on Fix-Inducing Changes: An Exploratory Study on GitHub Pull Requests

Syed Fatiul Huq, Ali Zafar Sadiq, Kazi Sakib
Institute of Information Technology
University of Dhaka
Dhaka, Bangladesh
bsse0732@iit.du.ac.bd, zafarsadiq120@gmail.com, sakib@iit.du.ac.bd

***Abstract*—Developer emotion or sentiment in a software development environment has the potential to affect performance, and consequently, the software itself. Sentiment analysis, conducted to analyze online collaborative artifacts, can derive effects of developer sentiment. This study aims to understand how developer sentiment is related to bugs, by analyzing the difference of sentiment between regular and Fix-Inducing Changes (FIC) – changes to code that introduce bugs in the system. To do so, sentiment is extracted from Pull Requests of 6 well known GitHub repositories, which contain both code and contributor discussion. Sentiment is calculated using a tool specializing in the software engineering domain: SentiStrength-SE. Next, FICs are detected from Commits by filtering the ones that fix bugs and tracking the origin of the code these remove. Commits are categorized based on FICs and assigned separate sentiment scores (-4 to +4) based on different preceding artifacts – Commits, Comments and Reviews from Pull Requests. The statistical result shows that FICs, compared to regular Commits, contain more positive Comments and Reviews. Commits that precede an FIC have more negative messages. Similarly, all the Pull Request artifacts combined are more negative for FICs than regular Commits.**

***Index Terms*—Software Engineering, Sentiment Analysis, Data Mining**

I. Introduction

Sentiment analysis – the process of extracting emotions from natural text [1] – has garnered significant interest in the field of software engineering, relating emotions with developer performance. Emotions have been observed to affect not only personal lives, but professional ones as well [2]. Understanding the extent of how employees' productivity differs based on their emotions has proven to change the perception of workplace culture [3]. In the domain of software engineering, software development processes have thus far not considered the impact of emotion. Nevertheless, existence of emotion has been observed in contributor submissions related to development [4] [5] [6]. The next step is to understand how this emotion relates to the performance of software engineers.

Software development processes differ from general organizational operations in the full adaptation and utilization of the online space. Online version control systems have become a key aspect of almost every software projects. GitHub is one of the most popular online version control and project management systems [7] that provide contributors with multiple collaborative artifacts e.g., Commits, Issues, Pull Requests etc. Among these, Pull Requests are not only directly linked to a developer's series of codes but also contain discussions from other contributors. A Pull Request encapsulates its context to a single task – fixing a bug or adding a feature. Hence, analysis on a Pull Request can help obtain an understanding on how the discussions regarding a task can influence its outcome.

This study, after extracting developer emotion, correlates these to Fix-Inducing Changes [8] – code that accidentally introduces bugs to the system, inducing its fix in the future. FICs are used for discovering and analyzing the introduction of bugs to a software. These can be detected from developer Commits [9], a component of Pull Requests, that document the changes a developer makes to the system. The introduction of bugs can be regarded as a metric for quantifying contributor performance. Therefore, observing whether sentiment in Pull Request components is related to FICs can yield an understanding of the effect of emotions on developer performance.

The study answers 4 Research Questions (RQ) which are divided based on the utilization of different artifacts. All the RQs aim to observe whether sentiment is related to FICs.

RQ1: How does sentiment of Commits in Pull Requests relate to Fix-Inducing Changes? Pull Requests contain multiple Commits from a contributor to complete a task. Previous Commits can contain indication of the outcome of a certain change. Emotion in Commits can be extracted from its message, in which the developers describe their changes.

RQ2: How does sentiment of Comments in Pull Requests relate to Fix-Inducing Changes? Comments contain the discussion among contributors about the task at hand. The discussions include recommendations and criticism on the code from a high level perspective by dedicated or concerned contributors. Therefore, it can influence future changes.

RQ3: How does sentiment of Reviews in Pull Requests relate to Fix-Inducing Changes? Reviews are posted directly with lines of code, as a suggestion for further modifications. These are more direct than the Comments. New Commits are then posted based on these Reviews. Hence, Reviews can directly affect the changes that follow it.

2640-0715/19/$31.00 ©2019 IEEE
DOI 10.1109/APSEC48747.2019.00075

RQ4: How does sentiment of all components in Pull Requests relate to Fix-Inducing Changes? All prior components together provide the general sentiment of a Pull Request and can predict a problematic change.

Previous studies on sentiment analysis in software engineering analyzed online repositories to derive relations between developer emotion and project properties. GitHub Commits were mined [4] [5] to observe how properties like day of Commit, project language and personnel diversity are related to sentiment. Issue comments from GitHub [6] and Jira [10] [11] worked with the effect of emotions in issue resolution activities. Other researches studied specific projects and analyzed the change of sentiment in relation to performance degradation [12] and core contributor's retirement [13]. Despite these researches, no study have yet been conducted to observe the relation between introducing bugs and developer sentiment.

This research is conducted in the following steps. First, taking six repositories from GitHub, all merged Pull Requests and Commits are extracted and stored. The contents of the Pull Requests are classified and divided based on the RQs. The contents are assigned sentiment scores, ranging from -4 to +4, using SentiStrength-SE [14]. Each Commit is subsequently given a score by averaging the scores of prior contents. Commits are then categorized as FIC or regular, after computationally tracking changes that induce fixes. The two categories of Commits, with their separate scores, are statistically analyzed to derive correlation between developer emotion and FICs.

The result yielded from this exploratory research show that relations between emotion and FICs exist for all the four RQs.

- Sentiment of Commits before an FIC is more negative than that of a regular Commit.
- FICs contain more positive Comments than regular ones.
- FICs are generated from more positive Reviews than regular Commits.
- Overall, FICs exist in Pull Requests more negative than regular ones.

II. Background

The aim of this research is to understand whether the sentiment in software development artifacts can indicate erroneous code. The study combines three fields of research: sentiment analysis, collaboration in GitHub and Fix-inducing changes.

A. Sentiment Analysis

Sentiment analysis, or opinion mining, is the process of quantifying the sentiment or emotion of natural text [1]. Calculating emotion provide insight into the speaker's opinion on the topic in question. It is one of the branches of data mining [15] that serves to give meaning to natural text computationally, where the size of the data makes manual reviewing impossible. Sentiment analysis has proven to be an effective tool in summarizing emotions from a large body of text from multiple types of sources, e.g., social media posts [16], movie reviews [17], app reviews [18] etc. With textual data populating the online space, the need to summarize and categorize this seemingly infinite data has increased exponentially.

As the technology for Natural Language Processing (NLP) improved, varying approaches for conducting sentiment analysis have been developed. Application of the different approaches have a common sequence of processes:

1) Text is taken as input and processed (stemmed, filtered etc.) for more efficient analysis.
2) The text is divided into specific chunks e.g., paragraphs, sentences or tokens.
3) The chunks are given individual sentiment scores based on a classification model. Scores usually have a range covering negative, positive and neutral emotions. The data used for training the classification model determines the context which the approach specializes on.
4) The whole text is given a single sentiment score by combining the individual chunks' scores, using a predefined merging algorithm.

For this study, SentiStrength-SE [14] is used as the tool to quantify the sentiment in software development artifacts. A tool derived from SentiStrength [19], SentiStrength-SE has been specialized for software engineering artifacts. Studies have shown its improved accuracy over other off-the-shelf sentiment analysis tools when run on software development related text [20] [21]. The tool calculates sentiment polarities – negative and positive emotions on a scale of -1 to -5 and +1 to +5 respectively. The scoring is conducted based on a sentiment dictionary that contains predetermined polarity scores of emotional words and phrases.

B. Pull Requests in GitHub

GitHub is one of the most popular version control platforms that also works as an online software development and management system. Its management aspect is built on the inclusion of collaborative artifacts such as Commits, Issues, Pull Requests, Comments, Reviews etc. These artifacts are directly linked to the source code of the software and are populated by contributor discussion. Since GitHub also contains all versions of the source code, its progression based on contributor discussions can be tracked using the artifacts.

Among the many features and artifacts GitHub provides for project management, this study is concerned with Pull Requests. Pull Requests are a feature of GitHub that contain code submissions by contributors, to be reviewed and merged into the system. The submission contains a set of Commits, and a description of the task conducted or problem fixed. Moderators, who are in charge of the system's main code, review the Commits. New sets of Commits may be submitted based on the reviews. If the Commits are satisfactory, the Pull Request is merged with the main code. Otherwise, it is rejected and closed.

This study utilizes three artifacts from Pull Requests:

1) Commits: A Commit is a contributor's documented modification to the source code. Each Commit is associated with a Commit message where the contributor describes the changes made.

2) Comments: Contributors can post comments on a Pull Request's feed to describe issues or suggestions regarding the changes. These discussions influence the contributor's consequent changes or Commits.
3) Review Comments: Reviewers can leave comments directly on a changed code. While Comments can be descriptive and usually elaborate on the task as a whole, Review Comments are used for commenting on a specific line of code. Newer Commits followed by Review Comments usually contain changes directly based on those Review Comments[1].

Pull Requests not only contain all the changes made regarding a specific task by a contributor, but also all the discussion and conversations related to those changes. The motivation behind each changed code is retained in the Comments and Reviews. Since Pull Requests construct a direct link between Commits and the Comments and Reviews, it is a useful feature to extract a correlation between discussion and code changes.

C. Fix-Inducing Changes

Fix-Inducing Changes (FIC) are the changes to code that unintentionally create problems in the system, requiring (inducing) a fix in the future. Changes are regarded as FICs once it is reverted or modified in a future change, under the context of fixing a bug. Bugs in the system originate in these FICs [8] and hinder the performance of a system. Hence FICs can be regarded as a coder's direct damage to the software and a negative metric for their performance.

FICs are detected using Commits since these formally document and store all changes. The detection process contains the following steps:

1) Commits that fix bugs are filtered and selected.
2) The code changes, specifically deletions, made in that Commit are listed.
3) The Commits that originated those deleted lines (buggy code) are tracked. These Commits are labeled as FICs.

III. Methodology

This study conducts sentiment analysis to find a correlation between Fix-Inducing Changes (FIC) and developer sentiment in project artifacts. The methodology to conduct this research is divided into four parts. First, the artifacts are collected from GitHub repositories. Then, those are classified based on the four Research Questions mentioned above. Next, sentiment analysis is conducted on the artifacts using SentiStrength-SE. Lastly, FICs are detected from all the Commits of that repository which are used to categorize the Commits. An overview of the methodology is provided in Fig 1 and descriptions in the following subsections.

A. Artifact Collection

Pull Requests are collected from the remote repository using the GitHub API[2]. Jcabi[3], a Java adapter for the API, is used. Jcabi provides helpful data containers for the different artifacts in GitHub as well as functions to extract their many properties. Using the API, all the 'closed' Pull Requests of a repository are traversed. 'Open' Pull Requests are excluded because these are still under process. Among the 'closed' Pull Requests, only the merged ones are stored, because rejected ones do not contain code that exists in the system's base code. Each Pull Request links to three different contents: Commits, Comments and Reviews. The three contents are stored separately with sets named CM, CN and RV respectively. These are also combined under a common class, Pull Content. All four sets are sorted based on their creation dates. The collection process results in a set of sorted Pull Contents under every single Pull Request. This set can be denoted as, $PC = \{PC_1, PC_2, ..., PC_{n-1}, C_n\}$, where n is the total number of Pull Contents for that Pull Request.

Furthermore, all the Commits of the repository, regardless of their inclusion in Pull Requests, are fetched. The remote repository is cloned locally using Jgit[4]. All the Commits stored within the local repository are traversed and stored. The full list of Commits are necessary for tracking FICs.

B. Artifact Classification

After the collection process is completed, the contents are classified based on the requirements of the Research Questions (RQ). Each RQ requires its own uniquely processed set of sentiment values which will be used for differentiating between regular Commits and FICs. Each sentiment value (snt) belongs to a Commit and the unique processes differ based on the selection of Pull Contents that were posted prior to that Commit. Sentiment scores of the three types of Pull Contents are quantified through the $senti()$ function.

1) ***RQ1****:* This RQ tries to find out how a Commit in a Pull Request is affected by sentiment existing in all the previous Commits. So, for each Commit in a Pull Request, only the Commits that precede it are considered for sentiment analysis. The sentiment score processed for this RQ is,

$$snt_{PC_x} = \frac{\sum_{i=1}^{x-1} senti(PC_i)}{x-1}, \tag{1}$$

where $PC_x, PC_i \in CM$ and $1 < x \leq n$. The first Commit of a Pull Request is disregarded since it does not have a preceding Commit to calculate sentiment from.

2) ***RQ2****:* This RQ tries to understand the emotional effect of Comments in a Pull Request on the outcome of a Commit. Usually a Comment contains discussion on the whole task instead of a specific part of code. Therefore, a Comment can influence any of the consequent Commits in a Pull Request. For this RQ, all preceding Comments of a Commit are considered to calculate the sentiment score,

$$snt_{PC_x} = \frac{\sum_{i=1}^{x-1} senti(PC_i)}{x-1}, \tag{2}$$

where $PC_x \in CM$, $PC_i \in CN$ and $1 \leq x \leq n$. Commits that have no preceding Comment are ignored from the final set of sentiment scores.

[1]this paper refers to "Review Comments" as simply "Reviews"
[2]https://developer.github.com/v3/
[3]https://github.jcabi.com/
[4]https://github.com/eclipse/jgit

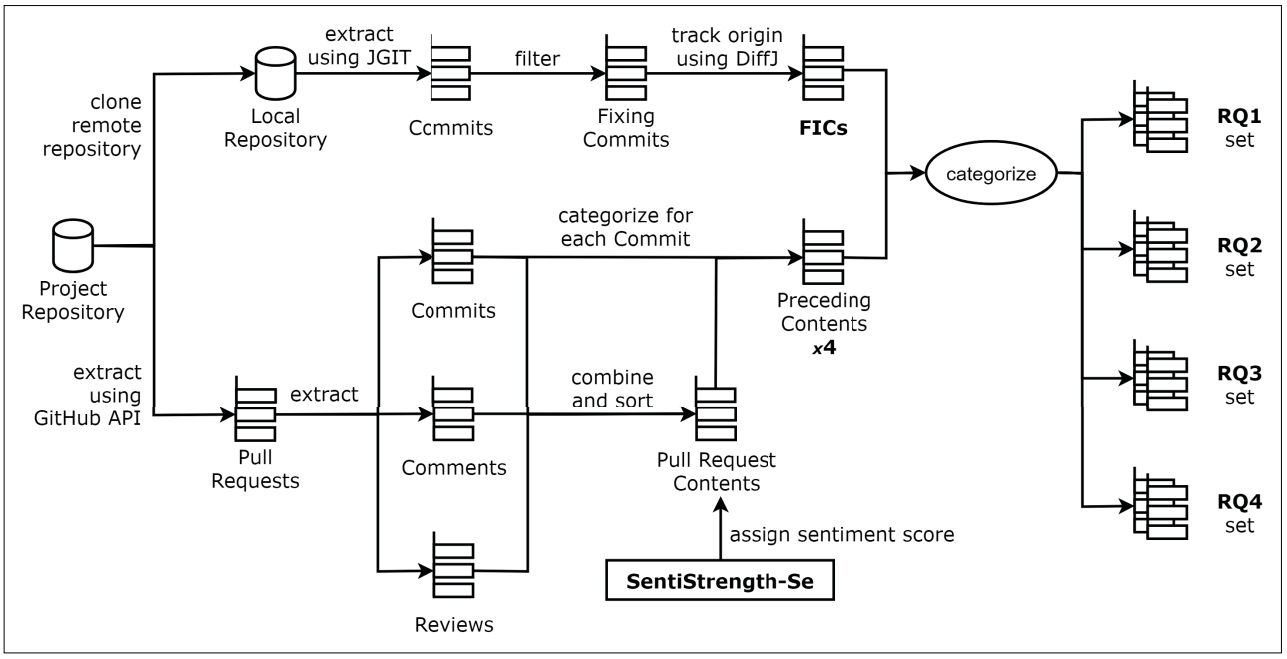

Fig. 1: Diagram of the methodology

3) ***RQ3****:* This RQ experiments upon the correlation between Reviews and a Commit in a Pull Request. A Review contains instructions on how to rectify a specific code in a Commit. This prompts new Commits that change code according to the Review's instructions. The effect of a Review only spans in Commits that directly follow it. So the sentiment score for a Commit (CM_x) is the average of all the Reviews that were posted since the previous Commit (CM_p). If no Reviews exist in this time-frame, CM_x is given the same score as CM_p since both are influenced by the same set of Reviews. Sentiment score of a Commit for both these cases is,

$$snt_{PC_x} = \begin{cases} \frac{\sum_{i=p}^{x-1} senti(PC_i)}{x-p}, & \text{if } x-p \geq 1 \\ snt_{PC_p}, & \text{otherwise,} \end{cases} \quad (3)$$

where $PC_x, PC_p \in CM$, $PC_i \in RV$ and $1 \leq p \leq x \leq n$. Commits that have no preceding Review are discarded since no Review influences these.

4) ***RQ4****:* The last RQ takes all the Pull Contents into consideration. The general sentiment of a Pull Request can be observed through this. The sentiment score for this RQ is,

$$snt_{PC_x} = \frac{\sum_{i=1}^{x-1} senti(PC_i)}{x-1}, \quad (4)$$

where $1 \leq x \leq n$. Commits that have no preceding Pull Content are not considered.

C. Quantifying Sentiment

Sentiment polarity scores for the artifacts collected and classified previously are calculated using SentiStrength-SE. The tool first takes as input one or more sentences of text. The texts are then divided into individual sentences which are tokenized, filtered, stemmed and lemmatized. Scores for each of the tokens are assigned based on the sentiment dictionary. The dictionary consists of words or tokens and their corresponding sentiment polarity score from -5 to +5. The polarity indicates the type of sentiment – negative (< 0), positive (> 0) or neutral ($= 0$) – and the level of severity, based on the distance from 0.

Using the individual scores of each token, the sentence as a whole is assigned two polarity scores – a negative and a positive point. The two points consist of the lowest negative (neg_{min}) and highest positive (pos_{max}) token scores in the sentence respectively. If no negative or positive token exist, the default polarity point is -1 and +1 respectively.

Subsequently, the complete text is assigned two polarity scores, taking the highest ones from the individual sentences. This also follows the default -1 and +1 scoring rule. The final score of the text is the sum of the two points,

$$senti(text) = neg_{min} + pos_{max} \quad (5)$$

The final score has a range of -4 to +4. -4 indicates a strong negative emotion and +4 a strong positive one. The severity decreases as the score nears 0 with 0 being a neutral emotion.

D. Fix-Inducing Change Detection

FICs are the erroneous changes to code that threaten the correctness of the system. Tracking these changes helps provide insight into the causes of introducing bugs – both technological and behavioural. FICs are detected in the following steps:

1) For each Commit extracted using Jgit, its message is analyzed to check whether it is a 'fixing Commit'. Fixing Commits are the changes to code that remove bugs or other errors. Therefore, by identifying the codes that fixing Commit removed, the exact lines of code that possibly created the bug can be found. The Commits are filtered by the inclusion of these keywords in the message: "bug", "fix" and "patch" [9]. The filtered Commits are regarded to be related to bug fixing activities, hence are labeled as fixing Commits.
2) In the next step, modifications conducted in the fixing Commits are extracted. Modification of source code in a single Commit can be found by analyzing the difference between that Commit and its parent Commit. A source code can consist of format change as well as code change. To find the changes introduced in a fixing Commit, all changed files are compared using DiffJ[5] tool. The tool compares changes between the two versions of that file existing in the fixing Commit and its parent Commit. DiffJ also disregards format changes, considering only source code modifications.
3) In the last step, the FIC is detected using the output of DiffJ tool[6]. The tool's output is the lines of code that were modified, which can be interpreted as errors that were mitigated. Finding the source of these lines can lead to the Commit that introduced errors, hence the FIC. The sources are tracked using Jgit's "blame" function, which finds the exact Commit where a line of code is added. This generates the list of FICs while the rest are considered regular Commits.

This process of detecting FICs is similar to [9]. Manual check for each repository shows that the Fixing Commits are indeed intended for bug fixes.

The end result of conducting all the four processes of the methodology are 4 paired sets of sentiment scores. Each paired set consists of one set of sentiment scores for FICs and another for regular Commits. The 4 pairs are based on the classification for each of the RQs.

IV. The Experiment

The processes described in the previous section are conducted on six different GitHub repositories. The description of the repositories and the outcome of each of the processes are described in the following subsections.

A. Repository Description

To conduct this research, six well known[7] Java projects are chosen from GitHub's repositories. Java is the language of choice because it provides necessary resources for extracting Fix-Inducing Changes (FIC) from the code e.g., DiffJ. These repositories are of open source projects, which ensure that the artifacts are publicly available. These also house contributors who do not have a central workplace and need to communicate through online collaborative artifacts.

The six projects with their repository details are displayed in Table I. The projects show maturity with an average project life of 7 years and average (median) 189 contributors. All the projects have a significant number of Commits to work with ranging from 4,798 to 23,550, covering projects of different sizes. With an average of 1,115 Pull Requests, the repositories together hold a sizeable number of contents to analyze on.

B. Content Description

Applying this study's methodology on the six projects produced 4 paired sets of sentiment scores. All pairs contain a set of FICs and a set of regular Commits. Fig 2 provides numeric comparison between the two types of Commits. The four sets represent the distinct scenarios of the four RQs.

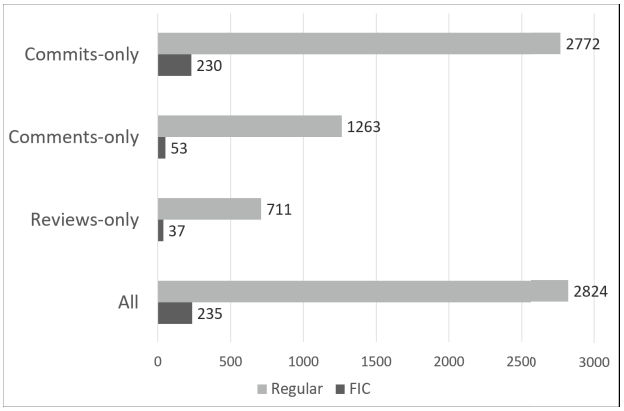

Fig. 2: Comparison of FICs and regular Commits in different scenarios

The Figure shows a large disparity in numbers between the two types of Commits. On an average only 6% Commits are FICs. This indicates the effectiveness of the review processes in Pull Requests, where each code is scrutinized methodically to remove any shortcoming.

V. Result Analysis

Each of the four paired sets are analyzed individually to answer the four Research Questions (RQ). Analysis and discussion of each RQ is provided as follows.

RQ1: How does sentiment of Commits in Pull Requests relate to Fix-Inducing Changes?

For this RQ, for every Commit in a Pull Request, all the previous Commits of that Pull Request are considered. Sentiment scores are calculated from the messages of those Commits. These are averaged and assigned to the Commit being analyzed. The process is repeated for all the Commits of all Pull Requests from all the projects.

Table II demonstrates the difference of sentiment for FICs and Regular Commits. It is observed that, on an average, the Commit messages before an FIC have less positive sentiment than a regular Commit's. A p-value less than 0.05 calculated

[5]https://github.com/jpace/diffj
[6]DiffJ was used Instead of Jgit as it omits format and whitespace changes.
[7]https://medium.com/issuehunt/top-11-popular-java-projects-on-github-48aaad5b4e0a

TABLE I: INFORMATION OF REPOSITORIES

Project Name	https://github.com/	Lifetime (Years)	Contributors	Commit Number	Pull Requests
Google Guava	google/guava	5.0	187	4798	469
Mockito	mockito/mockito	6.7	155	5019	826
Apache Commons-lang	apache/commons-lang	10.0	115	5396	431
Apache Hadoop	apache/hadoop	4.8	191	21435	1073
Selenium	SeleniumHQ/selenium	9.3	435	23550	1561
Spring Framework	spring-projects/spring-framework	6.4	378	18022	2331
Average		7.0	189	13036	1115

TABLE II: RESULT OF THE SENTIMENT ANALYSIS

Artifacts	Fix-Inducing Commits				Regular Commits				p-value	Decision
	Neutral	Negative	Positive	Mean	Neutral	Negative	Positive	Mean		
Commits	50%	31.7%	18.3%	8.7E-05	71.4%	16.3%	12.2%	0.004	0.019	Effect exists
Comments	9.4%	3.8%	86.8%	0.37	26.2%	9.9%	63.9%	0.26	0.031	Effect exists
Reviews	29.7%	10.8%	59.5%	0.11	53.4%	15.3%	31.2%	0.09	0.022	Effect exists
All	41.3%	28.5%	30.2%	0.03	48.9%	17.9%	33.2%	0.04	0.047	Effect exists

from a Wilcoxon Rank Sum test tells that difference in the mean is statistically significant.

Furthermore, FICs contain more negative Commits than positive (31% to 18.3%). This difference is larger than that of regular Commits (16.3% to 12.2%). In terms of neutrality, Commits before FICs are 21% less neutral than regular ones, meaning Commits before FICs show more emotion.

These results suggest that negative emotions in contributor Commits can indicate an FIC to come. If a contributor, when coding for a task, submits Commits that contain, in majority, negative sentiment, then their next Commits are prone to be FICs. This information can help in prioritizing or focusing reviews. Commits after a series of negative ones should be reviewed more intensely for possible introduction of bugs.

RQ2: How does sentiment of Comments in Pull Requests relate to Fix-Inducing Changes?

Pull Requests contain Comments as a medium for discussion among multiple contributors and for this RQ, the effect of these discussions is observed. FICs and regular Commits are given sentiment scores by averaging the scores of all the Comments that precede it. Comments consist of queries on the problem and solution, high level criticism of the code's approach, and suggestions on how to solve and improve. Their sentiment can alter developer emotion and thus affect their work. The results align with this assumption.

Table II shows that, based on the p-value from Wilcoxon Rank Sum test, Comments before FICs are significantly more positive than that of regular ones. Positive Comments outnumber both neutral and negative Comments by a large margin. FICs have almost 23% more positive Comments posted than regular ones. Negativity for FICs is very low, ranging around 4%. Lastly, neutrality is more common for regular Commits, containing 16.8% more neutral Comments than FICs.

The results indicate that too much positive emotions in discussion may lead to buggy code. Positive emotion in Comments are caused by gratitude or praise. These are needed as positive reinforcement for the developer. But, according to the results, these can cause the opposite outcome. Positive sentiment can turn developers overconfident and careless, reducing their ability to scrutinize their own code. Furthermore, in case of multiple contributors reviewing a task, one contributor's positive sentiment can bias the others' judgement.

RQ3: How does sentiment of Reviews in Pull Requests relate to Fix-Inducing Changes?

Reviews are an essential component in Pull Requests where reviewers leave comments linked to the code. These contain requests, suggestions and questions to the developer regarding specific lines of codes. Reviews, contextually, are nearer to the code than Comments. Hence their area of effect is also smaller and more immediate. New Commits abide by Reviews posted directly before them. The sentiment of Commits for this RQ, therefore, is calculated based on the nearest Reviews.

Statistical analysis shows that Reviews, like Comments, are more positive for FICs than regular Commits. As Table II demonstrates, based on the Wilcoxon Rank Sum test, the difference of sentiment is significant. FICs contain almost 30% more positively labeled Reviews than regular Commits. The ratio of positive Reviews is more than half of all sentiments (almost 60%). Although both types of Commits have almost similar percentage of negative Reviews (11% and 15%), neutrality is more dominant for regular Commits (53% to 30%).

This result is similar to that of RQ2, but of lesser numerical extent. Reviews that provide positive reinforcement can also distort a coder's ability to be careful with the code. The presence of emotion can confuse the coder in understanding the impact of their shortcomings. Reviews being directly linked to code, should be worded as formally as possible, omitting tangential discussions and being to the point.

RQ4: How does sentiment of all components in Pull Requests relate to Fix-Inducing Changes?

All components of a Pull Request determine the general sentiment conveyed regarding a task. While Comments and Reviews show contributor reception to code, Commit messages hold contributor reaction to feedback. Together these set the tone of a virtual atmosphere, which in turn can influence the quality of code. A negative atmosphere has the potential to cause adverse effect in a developer's performance. The statistical analysis show that that is possible.

Results of the statistical analysis are provided in Table II. It shows that, on an average, the Pull Requests of an FIC

are more negative than that of regular Commits. Wilcoxon Rank Sum test solidifies the difference. The difference in ratio between the two categories of Commits is minimal. The internal ratios of the different sentiment polarities are also similar in both cases. Regardless, neutrality is higher for Pull Requests of regular Commits. This means that more emotion is displayed in Pull Requests that influence FICs.

The first three RQs dealt with the components separately. All three observed that polar emotions play an adverse effect on the code. While Comments and Reviews are more positive for FICs, previous Commits are more negative. Comments hold the least amount of neutral sentiment, owing to emotional discussions. Reviews also show a similar pattern, but at a smaller quantity. Commits, as well, show that lack of neutrality results in FICs. The results of the last RQ reinstates this effect of neutrality.

Despite FICs containing more negative artifacts on an average, the larger disparity between FICs and regular Commits are in the neutral artifacts. Similar to individual effects, together the components are less neutral for FICs. This demonstrates a pattern where predominance of polar emotion in discussions lead to changes that introduce bugs.

Based on the inference of the statistical outcome, the information can be applied in the following two ways:

- **Anticipating FICs by analyzing sentiment:** The results show that FICs are preceded by: (1) negative Commits, (2) positive Comments and (3) positive Reviews. These can help in anticipating a buggy code. Moderators and reviewers can analyze the sentiment of the components of a Pull Request, and bolster reviewing processes if the mentioned patterns are found. Future studies can develop prediction models based on the individual and combined effects of the components.
- **Moderating the language of reviews:** Since emotions affect FICs, discussions in Pull Requests can be curated accordingly. The results warn of the detrimental effects of polar emotions. Neutrality was constantly larger for regular Commits compared to FICs. Furthermore, the opposite effects of Comments and Reviews, and Commits show that, a balance between positive and negative emotions is required. Based on these instructions, moderators and managers can administer how collaborative artifacts are used and how reviews should be worded.

VI. Threats to Validity

The results of this study has been produced from Pull Requests of 6 Java projects from GitHub. All 6 projects are live, and adding, merging and closing Pull Requests on a regular basis. These can also be reopened for further activities. Hence, the dataset cannot be recreated by following the processes mentioned alone. Table III provides the date the repositories were last accessed and the latest Pull Request extracted. A correct replica of the dataset can be created by filtering based on the dates mentioned and checking the ID of the highest Pull Request extracted.

TABLE III: Project Last Access Information

Project	Last Access Date	ID of Latest PR
Google Guava	2-Jul-19	3519
Mockito	4-Jul-19	1727
Apache Commons-lang	3-Jul-19	433
Apache Hadoop	6-Jul-19	1049
Selenium	4-Jul-19	7347
Spring Framework	3-Jul-19	23191

For sentiment analysis, the dictionary provided by SentiStrength-SE[8] is used, but with a single modification. The dictionary gives "mess*" a sentiment score of -2, to reflect the negative connotations of words like "mess", "messy", "messed" etc. But the prefix also includes "message" which is a neutral word. Hence, the dictionary is updated for this study, replacing "mess*" with the individual variants of "mess".

VII. Related Work

With the exponential increase of textual data over the internet, data analysis techniques have become popular for analyzing the data and deriving useful information. Sentiment analysis is one of these data analysis techniques that has been used to understand trends in social media [16] and user reception of products [22]. It has also been applied in organizations, to study the influence of employee emotions on their quality of work. Choudhury et al. [2] found relations between employee performance and emotion, which indicates the importance and applicability of sentiment in the workplace.

Private software development endeavors follow a similar environment to other organizational workplaces. Guzman et al. [12] extracted the flow of developer sentiment on three projects using collaborative artifacts like Commit messages, bug reports etc. The results show that the sentiment in the artifacts are positively correlated with real life developer behavior. Real life events have also been observed to effect public software development i.e., open source projects. Garcia et al. [13] analyzed the Gentoo project and found developer turnover to be related with sentiment.

Open source projects have been intensively mined to extract patterns related to developer sentiment. Issues are an important aspect of software development, consisting of developer discussions rich with emotions [6]. Mining efforts on Jira issues have shown that positive emotions are linked to shorter issue resolution time [10] [11]. GitHub, a leading version control and source code management system, have also been explored to discover emotional patterns. Mining the Commit messages in numerous open source projects yielded relations between sentiment and various properties of developers and the project [4] [5]. These properties include time and day of Commit, number of changed files, project language, geographical distribution of the team and project rating. These relations indicate that developer emotions vary based on the environment and characteristics of the project. Impact of these patterns can be better understood if sentiment is correlated to a developer performance metric.

[8]https://laser.cs.uno.edu/Projects/Projects.html

An important metric for developer performance is Fix-Inducing Changes (FIC). FICs indicate Commits that create bugs in the software that are later fixed [8]. Studies have analyzed FICs to derive useful information on coding, such as, how fixes themselves can create new bugs [23] and its relation with code coupling [24]. Apart from programmatic cause and effects, FICs can also be analyzed for the behavioural aspects of developers.

Sentiment analysis studies on software engineering so far have observed relation of emotions with assigned tasks or real life events. Contributory artifacts have been analyzed to discover emotional patterns of developers. Researches to better utilize these information, by linking sentiment with appropriate performance metric like FIC, have yet been conducted.

VIII. Conclusion

This study observes whether developer sentiment in software development artifacts can indicate the introduction of bugs in the system. Fix-Inducing Changes (FIC) are used as the metric to determine the origin of bugs. Sentiment is derived from GitHub's Pull Requests, which contain collaborative and contributory artifacts like Commits, Comments and Reviews. SentiStrength-SE is used for calculating the sentiment polarity of the artifacts. FICs are detected through an automated approach of filtering Commits that fix bugs and finding the origin of the code these remove. Finally, Commits are categorized as FICs and regular Commits, and assigned sentiment scores based on preceding artifacts.

Analysis conducted on six Java projects shows that there are significant differences of sentiment scores between FICs and regular Commits. Comments and Reviews before FICs have a more positive sentiment than that of regular ones. Conversely, Commits and the overall discussions preceding FICs are more negative than ones before regular Commits. This information suggests that whether bugs are being introduced in the system can be indicated through the emotions of collaborative artifacts that precede a change.

Future studies can utilize this applicability of sentiment for understanding developer performance. Other artifacts, like Issues can be considered, where the participation of clients and other contributors tend to be more opinionated.

Acknowledgment

Virtual machine facilities for this research, necessary for mining repositories online, are provided by Bangladesh Research and Education Network (BdREN).

References

[1] B. Pang, and L. Lee. "Opinion mining and sentiment analysis." Foundations and Trends® in Information Retrieval 2.1–2 (2008): 1-135.

[2] M. De Choudhury, and S. Counts. "Understanding affect in the workplace via social media." Proceedings of the 2013 conference on Computer supported cooperative work. ACM, 2013.

[3] R. J. Jones, S. A. Woods, and Y. R. Guillaume. "The effectiveness of workplace coaching: A meta-analysis of learning and performance outcomes from coaching." Journal of Occupational and Organizational Psychology 89.2 (2016): 249-277.

[4] E. Guzman, D. Azócar, and Y. Li. "Sentiment analysis of Commit comments in GitHub: an empirical study." Proceedings of the 11th Working Conference on Mining Software Repositories. ACM, 2014.

[5] V. Sinha, A. Lazar, and B. Sharif. "Analyzing developer sentiment in Commit logs." Proceedings of the 13th International Conference on Mining Software Repositories. ACM, 2016.

[6] F. Jurado, and P. Rodriguez. "Sentiment Analysis in monitoring software development processes: An exploratory case study on GitHub's project issues." Journal of Systems and Software 104 (2015): 82-89.

[7] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. "Social coding in GitHub: transparency and collaboration in an open software repository." Proceedings of the ACM 2012 conference on computer supported cooperative work. ACM, 2012.

[8] J. Śliwerski, T. Zimmermann, and A. Zeller. "When do changes induce fixes?." ACM sigsoft software engineering notes. Vol. 30. No. 4. ACM, 2005.

[9] S. Kim, E. J. Whitehead Jr, and Y. Zhang. "Classifying software changes: Clean or buggy?." IEEE Transactions on Software Engineering 34.2 (2008): 181-196.

[10] M. Mäntylä, B. Adams, G. Destefanis, D. Graziotin, and M. Ortu. "Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity?." Proceedings of the 13th International Conference on Mining Software Repositories. ACM, 2016.

[11] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli. "Are bullies more productive?: empirical study of affectiveness vs. issue fixing time." Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, 2015.

[12] E. Guzman, and B. Bruegge. "Towards emotional awareness in software development teams." Proceedings of the 2013 9th joint meeting on foundations of software engineering. ACM, 2013.

[13] D. Garcia, M. S. Zanetti, and F. Schweitzer. "The role of emotions in contributors activity: A case study on the Gentoo community." 2013 International Conference on Cloud and Green Computing. IEEE, 2013.

[14] M. R. Islam, and M. F. Zibran. "SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text." Journal of Systems and Software 145 (2018): 125-146.

[15] D. J. Hand. "Data Mining." Encyclopedia of Environmetrics 2 (2006).

[16] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. "Sentiment analysis of twitter data." Proceedings of the Workshop on Language in Social Media (LSM 2011). 2011.

[17] T. T. Thet, J. C. Na, and C. S. Khoo. "Aspect-based sentiment analysis of movie reviews on discussion boards." Journal of information science 36.6 (2010): 823-848.

[18] E. Guzman, and W. Maalej. "How do users like this feature? a fine grained sentiment analysis of app reviews." 2014 IEEE 22nd international requirements engineering conference (RE). IEEE, 2014.

[19] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas. "Sentiment strength detection in short informal text." Journal of the American Society for Information Science and Technology 61.12 (2010): 2544-2558.

[20] N. Novielli, D. Girardi, and F. Lanubile. "A benchmark study on sentiment analysis for software engineering research." 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR). IEEE, 2018.

[21] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto. "Sentiment analysis for software engineering: How far can we go?." 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018.

[22] H. Cui, V. Mittal, and M. Datar. "Comparative experiments on sentiment classification for online product reviews." AAAI. Vol. 6. No. 1265-1270. 2006.

[23] H. Yang, C. Wang, Q. Shi, Y. Feng, and Z. Chen. "Bug Inducing Analysis to Prevent Fault Prone Bug Fixes." In SEKE, pp. 620-625. 2014.

[24] A. Z. Sadiq, M. J. I. Mostafa, and K. Sakib. "On the Evolutionary Relationship between Change Coupling and Fix-Inducing Changes." (2019).