Analysis of Test Driven Development on sentiment and coding activities in GitHub repositories

This paper studies the relationship between Test Driven Development (TDD), productivity and developer sentiment in order to assess the impact of TDD on software development. We used a set of 256572 Java repositories archived from GitHub in September 2015 and made available through the Boa language and infrastructure. This research found that of these repositories, 9537 could be classified as using the TDD methodology. After obtaining these repositories we compared them to an equivalently sized set of control repositories. In general those repositories practicing TDD had fewer commits and a faster median rate of committing than did their control counterparts. We also found that TDD repositories generally contained fewer bug fixing commits. Finally, sentiment analysis was performed on both sets of repositories and it was determined that TDD repositories have a significantly (p-value 3.857e-12) more positive sentiment in comparison to the control repositories.

Analysis of Test Driven Development on Sentiment and Coding Activities in GitHub Repositories

Neil C. Borle
Department of Computing Science
University of Alberta
Edmonton, Canada
nborle@ualberta.ca

Meysam Feghhi
Department of Computing Science
University of Alberta
Edmonton, Canada
feghhi@ualberta.ca

Abram Hindle
Department of Computing Science
University of Alberta
Edmonton, Canada
hindle1@ualberta.ca

ABSTRACT

This paper studies the relationship between Test Driven Development (TDD), productivity and developer sentiment in order to assess the impact of TDD on software development. We used a set of 256572 Java repositories archived from GitHub in September 2015 and made available through the Boa language and infrastructure. This research found that of these repositories, 9537 could be classified as using the TDD methodology. After obtaining these repositories we compared them to an equivalently sized set of control repositories. In general those repositories practicing TDD had fewer commits and a faster median rate of committing than did their control counterparts. We also found that TDD repositories generally contained fewer bug fixing commits. Finally, sentiment analysis was performed on both sets of repositories and it was determined that TDD repositories have a significantly (p-value 3.857e-12) more positive sentiment in comparison to the control repositories.

CCS Concepts

•Software and its engineering → Software creation and management; Software development process management; •Information systems → *Miscellaneous;*

Keywords

Human Factors in Software Engineering; Opinion Mining; Sentiment Analysis; Test Driven Development

1. INTRODUCTION

GitHub is an online developer resource that is based on the Git version-control system (VCS). In GitHub, code repositories can be hosted publicly, encouraging open development and recording social interaction through software artifacts [3]. Because of GitHub's size and scale, a domain-specific language and infrastructure known as Boa has been developed to perform large scale queries over archived GitHub repositories to extract and summarize the software artifacts found in them[4].

Test Driven Development (TDD) is a software development methodology in which failing tests are written first, in advance of source code development, so that source code can be tested as it is developed[1].The provision of immediate feedback may have an emotional effect on the code developers, an effect on code quality or an effect on the rate at which code is developed.

In this paper we use Boa to gain insight into the software development process. In particular we study the prevalence and influence of TDD on software development processes and opinion in Java repositories hosted on GitHub in 2015. In our research we address the following questions:

RQ1: How many repositories use TDD?

RQ2: Does TDD affect the volume of commits in a repository?

RQ3: Does TDD improve commit velocity?

RQ4: Does TDD reduce the required number of bug fixes?

RQ5: Does TDD improve developer satisfaction?

In the process of addressing these questions we consider software commits/revisions as our primary source of information. Sentiment analysis is applied to revision logs in *RQ5* while log time stamps, changed files and key words are used for the other questions.

2. RELATED WORK

In 2007, Hindle *et al.* described a taxonomy for classifying revisions based on the type of files being changed in a revision. The classes of this taxonomy include source revisions, test revisions, build revisions and documentation revisions (STBD)[6]. In the context of Hindle *et al.* this study considers source code revisions and test revisions, the revision classes most relevant to TDD.

In the work of Zaidman *et al.*, two different open source repositories are studied to determine if testing strategies such as TDD are detectable[8]. Zaidman *et al.* develop a method of associating source code with test code by relying on a naming convention where the word "Test" is added as a postfix to a test file corresponding to a similarly named source file[8]. Like Zaidman *et al.* we also took advantage of this convention for associating test files to source files when detecting TDD. We differ from Zaidman *et al.* in that we do not only consider "Test" as a postfix, we match "Test" with case insensitivity, and we impose a minimum limit on the ratio of test files.

To establish the credibility of extracting sentiment from software artifacts, Murgia *et al.* conducted a study to determine if software developers left opinions behind during the development process. They concluded that developers do leave emotional content in their software artifacts. From this they suggest that the automated mining of developers' emotions from their software artifacts is feasible[7].

Guzman *et al.* used sentiment analysis to extract opinions from commit messages belonging to GitHub projects. Their research identified relationships between positive and negative sentiment and factors such as the programming language used, team members' geographic locations, day of the week and project approval[5]. Like Guzman *et al.* we sought to apply sentiment analysis to the messages left behind during the development process, but focused our approach on the comparisons of projects using TDD process with those not using TDD process.

3. METHODS

3.1 The Data Set

We used the Boa language and infrastructure to obtain 256572 Java repositories from a copy of GitHub archived September, 2015. From each repository, we obtained the repository URL, commit logs, commit time stamps, revisions, and file names associated with revisions where Java files were created. A Boa Output Parser[1] was used to convert Boa output into Python objects.

From these 256572 Java repositories we identified 9537 TDD repositories and extracted 97300 commit messages. From the remaining repositories containing at least one commit, we randomly sub-sampled a set of 9537 control Java repositories for comparison purposes.The controls had a total of 321051 commit messages. The set of control and TDD repositories were used for answering RQ2 through RQ5. It is important to note that the TDD set and the control sets were disjoint and that each repository from both sets had at least one commit and commit message.

3.2 Approach

3.2.1 Approach to RQ1: How many repositories use TDD?

To determine whether or not a repository was practicing TDD, we reviewed all the revisions and all associated Java files created in each revision. Here, we excluded all repositories where there were no Java files that could be identified as test files. Below is the step by step logic employed by the authors:

Step 1. We partitioned the Java files into two sets for each of the repositories. The first set of files were those that were identifiably test files, where the file name matched the regular expression shown below. The second set contained the remaining Java files.

```
/.*test.*\.java/i
```

Step 2. For each file in each set we identified the creation time by obtaining it from the revision in which it was created.

Step 3. For each of the test files we found matching or similarly named files. This assumed that Java programmers typically name their test files according to their source code files. For example, "someFile.java" might have a corresponding test file called "TestSomeFile.java" or "someFileTest.java". If no matching file could be found, we found similar files. Similar file names were identified using the Python Standard Library for string comparison. A similarity threshold of 0.8 was used, where 0 is completely dissimilar and 1 is exactly the same. This threshold was chosen because it could match, for example, words like "search" with "searching". Here the file "searchTest.java" would be similar to "searching.java". Finally, if no matching or similar files were found for a test file, that test file would be ignored in this step.

Step 4. Once we had obtained the mapping from a test file to the source code files that matched it or were similar, we then ensured that the time stamp associated with the test file was either older or the same as the source file(s). This ensured that the test file was either committed before or at the same time as the source file(s).

Step 5. As a last step, if at least one out of every four Java files in the repository were test files (25%), the repository was declared to be practicing TDD. This was an arbitrary threshold selected by the authors to reduce the number of false positives being included in the TDD set. After the completion of this step we now had our 9537 TDD repositories with a total of 97300 commit messages.

3.2.2 Approach to RQ2, RQ3, RQ4, RQ5

Having obtained a set of TDD repositories to answer RQ1, we then used this set and the corresponding sub-sampled control set of equal cardinality to answer the remaining research questions.

To address RQ2, we counted the number of commits associated with each repository from the two sets and observed the differences between them.

To address RQ3, we extracted the time stamps associated with all the the commits collected and then noted the average of the time stamp differences (deltas) between commits in each repository

To address RQ4, we used the regular expression shown below (continued on two lines) to identify bug fixing commits in the repositories. We then used this count as an approximation of the number of bugs that had existed in that repository.

```
/.*((solv(ed|es|e|ing))|(fix(s|es|ing|ed)?)
|((error|bug|issue)(s)?)).*/i
```

To address RQ5 we applied the methodology described by Guzman *et. al.*[5] using the software SentiStrength[2]. SentiStrength is described in detail in their work and so we refer the reader to that publication[5]. Once sentiment scores had been obtained for each of the 418351 commits, we then averaged the scores for each repository to obtain repository level sentiment scores.

4. ANALYSIS AND FINDINGS

4.1 How many repositories use TDD?

Of the 256572 Java repositories available from Boa, we identified 9537 (3.7%) as practicing TDD according to our criteria.

[1]https://github.com/eddieantonio/bop

[2]http://sentistrength.wlv.ac.uk/

4.2 Does TDD affect commit volume?

	TDD	Controls
Mean	10.20	33.66
STD	32.59	231.47
Quantile 0%	1	1
Quantile 25%	2	2
Quantile 50%	4	6
Quantile 75%	10	17
Quantile 100%	2526	8851

Table 1: Number of Commits per Repo

In Table 1 it can be seen that both measures of centre, mean and median (Quantile 50%), indicate that the control repositories have a larger number of commits in comparison to the TDD repositories. We can also observe that the largest control repository (8851 commits) is much larger than the largest TDD repository in terms of commits. A two sample Mann–Whitney–Wilcoxon test for this data returns "p-value < 2.2e-16", showing there is extremely strong evidence that these samples come from different populations and that control repositories have larger numbers of commits in general. We have omitted the the CDF curves in this section for brevity, noting that Table 1 provides us with a good understanding of the data.

4.3 Does TDD improve commit velocity?

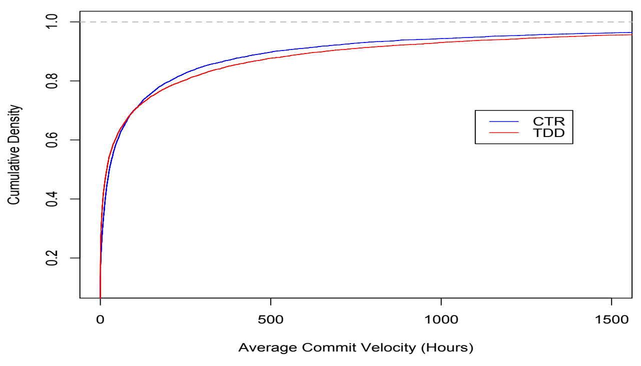

Figure 1: CDF of Average Commit Velocity per Repo

As shown in Figure 1 the TDD density curve sits above the control density curve until approximately 0.7, after which point the the control density curve becomes the upper curve. This indicates that most of the TDD repositories have a smaller average time difference between their commits as compared to the controls. Due to the right skewed in this data, the graph is truncated at 1500.

	TDD	Controls
Mean	297.92	263.19
STD	1099.73	1050.18
Quantile 0%	0	0
Quantile 25%	0.99	3.71
Quantile 50%	19.01	26.55
Quantile 75%	153.69	141.26
Quantile 100%	34072.11	31997.48

Table 2: Average Commit Time Difference per Repo (Hours)

As seen in Table 2, control repositories have a larger median commit time than TDD repositories, the more important measure of centre due to skew. A two sample Mann–Whitney–Wilcoxon test for this data returns a p-value of 3.174e-10, showing there is extremely strong evidence that control repositories generally have larger elapsed times between commits.

It is also interesting to note here that the TDD set has larger values for the 75% and 100% quantiles. This suggests that the TDD repositories with the slowest commit velocity tend to be slower than the controls with the slowest commit velocity.

4.4 Does TDD reduce the number of bug fixes?

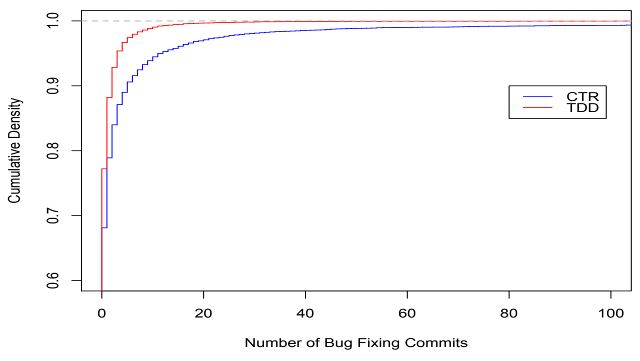

Figure 2: CDF of Number of Bug Fixing Commits per Repo

As shown in Figure 2, the density curve for the TDD repositories always sit above the density curve for the control repositories. Due to the right skew in this data, the graph is truncated at 100.

	TDD	Controls
Mean	0.75	4.79
STD	5.34	48.91
Quantile 0%	0	0
Quantile 25%	0	0
Quantile 50%	0	0
Quantile 75%	0	1
Quantile 100%	416	1894

Table 3: Number of Bug Fixing Commits per Repo

As shown in Table 3, most quantile values are 0 and so the data are very skewed. A two sample Mann–Whitney–Wilcoxon test for this data returns "p-value < 2.2e-16", showing there is extremely strong evidence that control repositories generally have larger number of bug fixing commits.

4.5 Does TDD improve developer satisfaction?

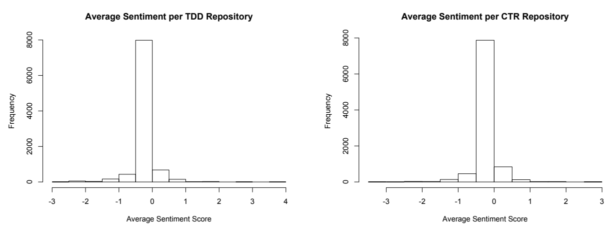

Figure 3: Average Sentiment Score per Repo

Figure 3 shows that sentiment expressed in these repositories is symmetric approximately around 0 and that in most cases no sentiment is expressed.

	TDD	Controls
Mean	-0.065	-0.075
STD	0.33	0.31
Quantile 0%	-3.0	-3.5
Quantile 25%	0	-0.12
Quantile 50%	0	0
Quantile 75%	0	0
Quantile 100%	4.0	3.0

Table 4: Average Sentiment Score per Repo

Table 4 show us that the mean sentiment of control repositories is slightly more negative than the mean sentiment of TDD repositories with approximately the same variance.

A two sample Mann–Whitney–Wilcoxon test for differences in sentiment returned a p-value of 3.857e-12. This suggests that there is very strong evidence to conclude that there are differences between the sentiments expressed in TDD repositories vs. non TDD repositories. In particular, that repositories using the TDD methodology generally have more positive commit messages.

To determine the degree to which the discrepancy in number of commits between control and TDD repositories might be affecting the sentiment scores, Spearman correlation was used on the union of both repository sets to determine the correlation between absolute sentiment score and number of commits. The resulting correlation was 0.5008541 which indicates that there is a relationship between these features. Therefore, the difference in number of commits between the control and TDD repositories could partially account for the stronger sentiment expressed in the control repositories.

5. THREATS TO VALIDITY

In this work, internal validity is threatened by our choice to use file names as our basis of TDD identification, and our selection of a 0.25 threshold for the proportion of Java files that had to be test files. In particular, this approach does not consider file contents and it is noted that not all developers use the source/test file naming convention we assumed, for example they may name files by their use case. It may be the case that repositories truly employing TDD were omitted from our count due to low test to source file ratio.

Another threat to internal validity was our use of hand crafted regular expressions for the identification of bugs fixing commits. This may be an over or underestimation of the true number of bugs.

External validity is threatened in this work by the authors' choice to work only with Java files. This was done out of convenience but means that this work may not generalize to other programming languages.

A final threat to external validity was our choice to not exclude small or personal projects from the study. While we have studied a set of repositories that are representative of GitHub, this work may not necessarily generalize to enterprise level software.

6. CONCLUSIONS AND FUTURE WORK

In this work we studied Java repositories on GitHub and compared those practicing Test Driven Development to those that did not. While our results are interesting, this study cannot claim that any of these results are the direct effect of implementing the TDD paradigm as there may be confounding factors in the data.

In our study we found that there were 9537 repositories on GitHub practicing TDD in September 2015, which corroborates evidence by Beller *et al.* that TDD in not commonly practiced[2]. When compared to a control set of non TDD repositories we found that generally, TDD repositories have fewer commits, have a faster commit velocity, require fewer bug fixes and have an overall more positive sentiment. Our results indicate that TDD might be an effective strategy for improving developer satisfaction and productivity.

Having discovered differences between repositories practicing TDD and those that do not, future work needs to be done to rigorously determine if these are the direct results of implementing a TDD methodology and to identify, if any, confounding factors that may have influenced these results. Extensions of this work will involve identifying TDD repositories by references to test classes instead of through file names, as well as investigating how source and test files correlate over time between TDD repositories and repositories using other development methodologies.

7. REFERENCES

[1] K. Beck. *Test-driven development: by example.* Addison-Wesley Professional, 2003.

[2] M. Beller, G. Gousios, A. Panichella, and A. Zaidman. When, how, and why developers (do not) test in their ides. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 179–190. ACM, 2015.

[3] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[4] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *35th International Conference on Software Engineering*, ICSE 2013, pages 422–431, May 2013.

[5] E. Guzman, D. Azócar, and Y. Li. Sentiment analysis of commit comments in github: an empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 352–355. ACM, 2014.

[6] A. Hindle, M. W. Godfrey, and R. C. Holt. Release pattern discovery via partitioning: Methodology and case study. In *Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on*, pages 19–19. IEEE, 2007.

[7] A. Murgia, P. Tourani, B. Adams, and M. Ortu. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 262–271. ACM, 2014.

[8] A. Zaidman, B. Van Rompaey, S. Demeyer, and A. Van Deursen. Mining software repositories to study co-evolution of production & test code. In *Software Testing, Verification, and Validation, 2008 1st International Conference on*, pages 220–229. IEEE, 2008.