Effect of Time Size on Team Productivity

Sara Gholami
Department of Electrical and Computer Engineering
University of Alberta, Edmonton, Canada
Email: sgholami@ualberta.ca

I. Introduction

The main goal of this study is to find the effect of team size on the productivity of the development team. The reason for conducting this study is to give some insight to project managers as at the beginning of any projects there are many different factors to be considered by the project managers and choosing the correct combination is a daunting task. One of these factors is the size of the development team were choosing a small number may result in the high pressure of workload on the development team and choosing a large team can result in having many developers without enough task and result in the waste of money of the company and the developers' time. So, the project managers want to select a team size which leads to an increase in the team productivity and revenue of the company. After selecting the number of team members, the project managers need to choose the project language where there are a wide variety of languages available. This study tries to find if there is any relation between team size and the language of the project. Consequently, there are two main research questions for this study as follows:

- **RQ1** What is the relation between team size and productivity of the development team?
- **RQ2** What is the relation between team size and the language of the project?

In order to answer these questions, firstly I performed some preprocessing on the data. Then, I examined the distribution of the variables to select the appropriate statistical test to find the correlation between team size and the rest of the variables. Afterward, I tested each of the team sizes as a threshold to figure out if any of can divide teams such that teams on one side of the threshold are better than the other teams with respect to all of the variables.

The contributions of this study are finding the effect of team size on some of the productivity factors and overall its effect on the productivity itself, finding a team size (7 based on finding) as a threshold which makes us able to compare with sizes below and above it. The source code of this study is available on GitHub[1] which can be easily reproduced.

II. Related Work

There have been many studies on software projects and how different factors are affecting each other. Rodrguez et. al [1] studied the relationship between software project team size and the productivity of the team. They performed an empirical study based on the International Software Benchmarking Standards Group (ISBSG) repository. They applied some statistical approaches and found that there is a correlation between productivity, effort, time and team size. Islam et.al [2] performed an empirical study on *travistorrent_8_2_2017.csv.gz* [3] dataset where they aimed to answer three research questions. Firstly, finding if there is a relationship between the complexity of a task and CI build failure. Secondly, if build strategies and developers' contribution model have an impact on CI build failure. Moreover, finding if there is any correlation between the size of teams and CI build failure .Also, some empirical studies evaluated the effect of team size on the software development effort, software size in function points and programming language type [4]. Another study estimated the impact of software team size on the software development cost [5].

III. Background and Terminology

This section provides a brief overview of some of the concepts used in this study as follows:

A. TravisTorrent

TravisTorrent is a dataset which is the GHTorrent partner, and its dataset contains 3.7 million rows of data where each row represents a build job executed on Travis CI. Every row in TravisTorrent combines data from three resources, the projects git repository, data from Traviss API and an analysis of the build log, and data extracted from GitHub through the GHTorrent with prefixes $git\_,\ tr\_,\ gh\_$ respectively.

B. Statistical Tests

There are various kinds of statistical tests where their usage depends on the variables' type and distribution.

- **Spearman's Rank-Order Correlation** is a non-parametric test. It measures the strength and direction of the association between two ranked variables.
- **Contingency Table with Chi-Square Test of Independence** a contingency table or matrix displays the frequency distribution of variables. In other words, it shows the observed frequency of two variables. The chi-square test of independence is a way to test if two categorical data are independent. First, we get the contingency table of two variables and then calculate the chi-square test based on that table.

[1]https://github.com/cmput402-w19/project-saragholami

IV. Methodology

A. Preprocessing

travistorrent_8_2_2017.csv.gz dataset is used to obtain data required to answer the research questions of this study. TravisTorrent gathered information about 1283 projects, where the team size of these projects varies between 0 and 288. As Figure 1 displays there exists only one sample project per team size for team sizes higher than 60. Therefore, I decided not to include projects with team sizes higher than 60 in my study as there cannot be a fair judgment based on only one sample. For example, if I concluded that teams with size 200 are more productive, then it will be based on just one sample which is not enough for this study. After removing these teams, it leaves 1277 projects available.

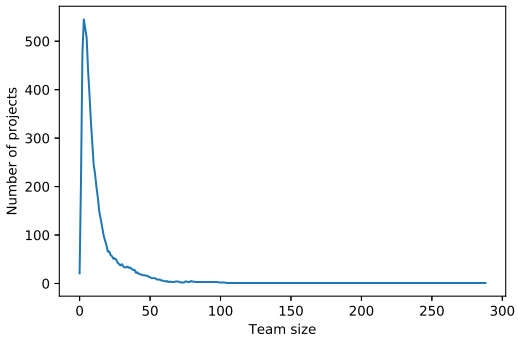


Fig. 1: Number of projects with different team sizes

From the TravisTorrent dataset, I selected some of the features which could be related to the productivity and the scope of this study such as the number of lines of the source or test code changes, success or failure of build job, etc. The following list represents this list of features:

- **Project Name** Project name on GitHub.
- **Team Size** Number of developers that committed directly or merged PRs from the moment the build was triggered and 3 months back.
- **Source Code Changes** Number of lines of production code changed (added, deleted or modified) in all commits that were built for this build.
- **Test Code Changes** Number of lines of test code changed (added, deleted or modified) in all commits that were built for this build.
- **Sloc** Number of executable production source lines of code, in the entire repository.
- **Test Lines in Test Cases per Kloc** Test density. Number of lines in test cases per 1000 Sloc.
- **Test Cases per Kloc** Test density. Test density. Number of test cases per 1000 Sloc.
- **Travis Status** The build status (such as passed, failed,) as returned from the Travis CI API.
- **Number of Tests Run** Number of tests that ran in total, extracted by build log analysis.
- **Number of Tests OK** Number of tests that succeeded, extracted by build log analysis.
- **Number of Tests Failed** Number of tests that failed, extracted by build log analysis.
- **First Commit Created at** Timestamp of first commit in the push that triggered the build, in UTC. In rare cases, GHTorrent has not recorded a push event for the commit that created the build in which case *first_commit_created_at* is nil.
- **Pushed at** Timestamp of the push that triggered the build (GitHub provided), in UTC.
- **Lang** Dominant repository language, according to GitHub.

Travis Status is a categorical data where I am mapped passed status to 1 and failed or errored status to 0. Afterward, I removed the rest of the builds with other statuses which were canceled and started. The table I represents the type and percentage of missing values for the selected features of the TravisTorrent dataset. As it can be seen the Number of Tests Run, Number of Tests Failed and Number of Tests OK have a high volume of missing values. Therefore, I removed the lines which contain a missing value in these features. This process reduced the number of data point to 1.5 million rows and left 1034 projects in the dataset.

TABLE I: Some of the TravisTorrent features

Variable	Type [Range]	Missing Values (%)
Team Size	Numeric[0, 288]	0
Source Code Changes	Numeric[0, 5,278,263]	0
Test Code Changes	Numeric[0, 393,495]	0
Sloc	Numeric[3, 1,379,466]	0
Test Lines in Test Cases per Kloc	Numeric[0, 414,076.92]	
Test Cases per Kloc	Numeric[0, 6,192.307]	
Travis Status	Categorical[passed, failed, errored, canceled, started]	0
Number of Tests Run	Numeric[1, 733,960]	48.9
Number of Tests OK	Numeric[0, 733,960]	45.8
Number of Tests Failed	Numeric[0, 733,960]	43
First Commit Created at	DateTime	20.4
Pushed at	DateTime	20.4
Lang	Categorical [Ruby, Java, Javascript]	0

Afterward, I normalized some of the selected features to remove the effect of different project sizes on the results which produced some new features as follows:

- **Source Code Changes Ratio** The changes of source code over the total size of source code.
- **Test Code Changes Ratio** The changes of test code over the total size of test code.
- **Time Interval** The time between creating the first commit and the push time.
- **Source Code Changes Ratio Over Time** Source code changes ratio normalized by time spent on it.
- **Test Code Changes Ratio Over Time** Test code changes ratio normalized by time spent on it.
- **OK Tests Ratio** The number of passed tests over the total number of tests.
- **Failed Tests Ratio** The number of failed tests over the total number of tests.
- **Travis Status Passed Ratio** The number of passed builds over the total number of builds.
- **Travis Status Failed Ratio** The number of failed builds over the total number of builds.

Then I needed to check the distribution of each of the variables in order to choose the correct statistical test to find whether there is any correlation between each of them and team size. Figure 2 illustrate the distribution of some of the features. As it can be seen, none of them follows a normal distribution. So, to test the correlation of these features with team size, we cannot consider tests with normal distribution assumption.

B. Statistical Analyses

In order to answer the first research question and to find whether team size has any effect of team productivity I tested the correlation between team size and each of the features using the Spearman's rank correlation test.

Then I conducted the following test to find if there exists a team size which can divide teams such that teams on one side are on average better than the other teams with respect to all of the parameters related to productivity. As a result, I sampled the dataset 100 times with replacement to be able to find any trend in the data. Afterward, for each of the sample sets, I tested each of the team sizes ranging between 0 and 60. The following code displays this procedure.

```
sample_sets = 100
team_sizes = 60
for i in range(sample_sets):
    for threshold in range(team_sizes):
        avg_below = Average of parameter x of
         ↪ below threshold team sizes
        avg_above = Average of parameter x of
         ↪ above threshold team sizes
        difference = avg_below - avg_above
```

, Where x is one of the parameters including source code changes ratio over time, test code changes ratio over time, test failure ratio, tests' success ratio, Travis status passed ratio and Travis status failed ratio. Then, I check if there was any team size which separated the teams such that teams on one side of the threshold are always better than the others with respect to all of the parameters.

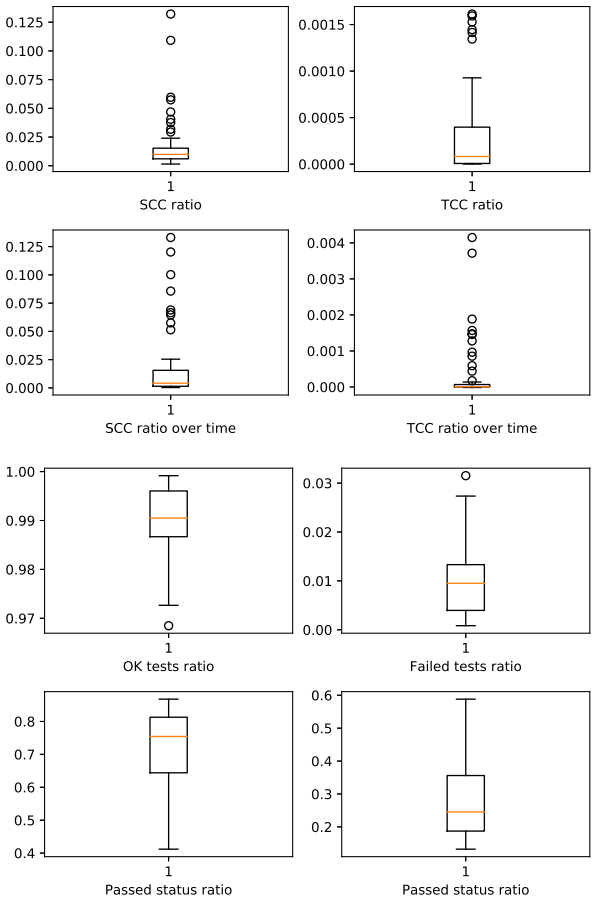


Fig. 2: Distribution of all of the variables

Afterward, to answer the second research question which is the relation between team size and primary language of the project I checked the list of different languages in the dataset which are only Ruby, Java and JavaScript. Then, I checked how many projects are in each of the languages and the distribution of these languages with respect to team size. After that, I ran the contingency table with the chi-square test of independence to find if there is any association between team size and language of the project. The results are available in the following section.

V. Results

In this section will present the results of the applied methods.

Table II displays the results for the Spearman's rank correlation test between team size and other features. Based on these results, team size and Source Code Changes Ratio are highly correlated (-0.9069) but when we get this changes over the unit of time the correlation between them becomes weaker and is equal to -0.7769. While there is no apparent correlation between team size and test code changes ratio. Similarly, it can be seen that team size, and the ratio of tests passing or failing are highly not-correlated with a correlation value of 0.0994 and -0.0994 respectively. The same this happened for

Travis status ratio for passing and failing build jobs. As a result, the only features correlated with team size are source code changes ratio and its ratio over time, which mean smaller teams change more source code with respect to the size of the project, but it does not indicate if the changes were more successful and passed more tests.

TABLE II: Spearman's rank correlations between each of the variables and team size

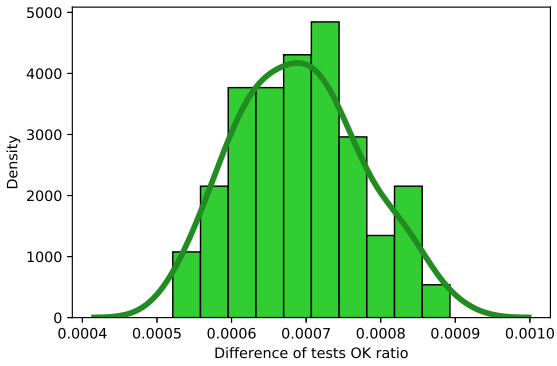
Variable	Spearman Correlation
Source Code Changes Ratio	-0.9069
Test Code Changes Ratio	0.4255
Time Interval	-0.3368
Source Code Changes Ratio Over Time	-0.7769
Test Code Changes Ratio Over Time	-0.2189
OK Tests Ratio	0.0994
Failed Tests Ratio	-0.0994
Travis Status Failed Ratio	0.5741
Travis Status Passed Ratio	-0.5741

After testing each of the team sizes for all 100 sample sets the smallest threshold which separated teams based on our criteria was team size 7. Figure 3a displays the difference of OK test ratio of teams smaller and larger than 7, where smaller teams had a higher ratio. Similarly, figure 3b display the difference of ratio of the failed tests where smaller teams had a less failed team. So, teams smaller than 7 had a higher OK ratio of tests and less failure in tests.

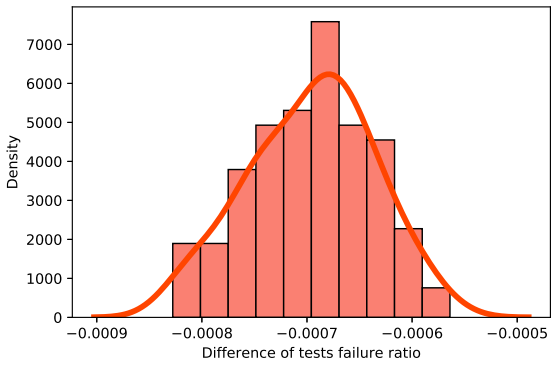
In the same way, figures 4a, 4b illustrate the difference of source and test code changes ratio over time respectively, where teams with size less than 7 had more changes which mean they produced more code. Similarly, according to figures 5a, 5b teams smaller than 7 had more passed build jobs with respect to the total number of builds and less failed build job ratio. Based on the results we can drive the answer to RQ1 as follows:

Answer to RQ1: Teams with smaller sizes- specifically 7 based on our findings- tend to be more productive as they produced more lines of code and their code was less error-prone and passed more tests.

To answer **RQ2**, I check the different languages available in the dataset and number of projects in each of languages. Table III display that there are only three languages in the TravisTorrent dataset which are Ruby, Java, and JavaScript with 776, 254 and 4 projects using each language respectively. As the numbers suggest a high volume of projects are written in Ruby and with lower proportion in Java. Afterward, I checked the distribution of the projects using each of the languages with respect to their team size. Based on figure 6 none of the languages follow a normal distribution. As this



(a) Tests OK ratio



(b) Tests failed ratio

Fig. 3: Comparison of tests ratio for team sizes below and above 28

data is a categorical data type, I used the contingency table with chi-square test of independence and based on that the p-value was 0.9989 which is very close to 1 and indicates that team size and primary language of projects do not have any association. So, our conclusion for QR2 is as follows:

Answer to RQ2: Based on the findings from TravisTorrent dataset we cannot make any judgment about the relation between team size and the primary language of the project.

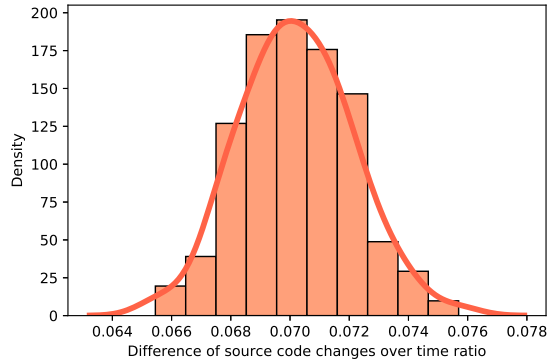
TABLE III: Different languages in TravisTorrent

Language	Number of Projects
Ruby	776
Java	254
JavaScript	4

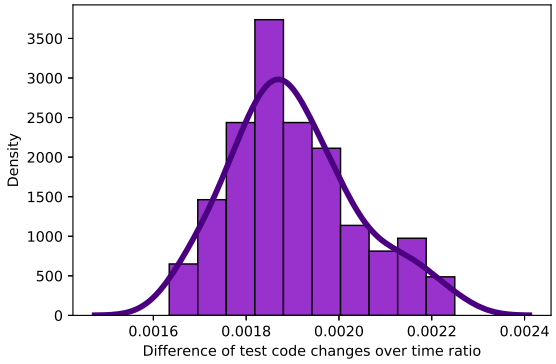
VI. Threats to Validity

There exist multiple threats to the validity of this work.

- The TravisTorrent dataset is very skewed where %87 of the teams have team size smaller than 10. As a result,

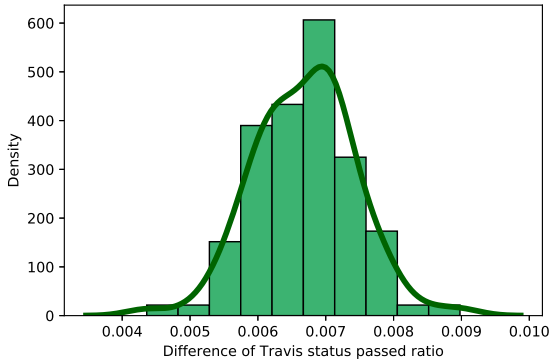


(a) Source code change ratio over time

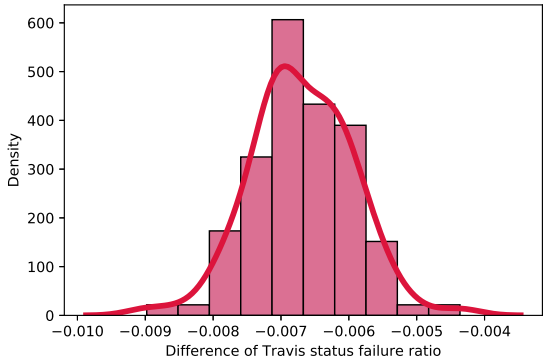


(b) Test code change ratio over time

Fig. 4: Comparison of code change ratio with respect to time for team sizes below and above 28



(a) Travis status passed ratio



(b) Travis status failed ratio

Fig. 5: Comparison of Travis status ratio for team sizes below and above 28

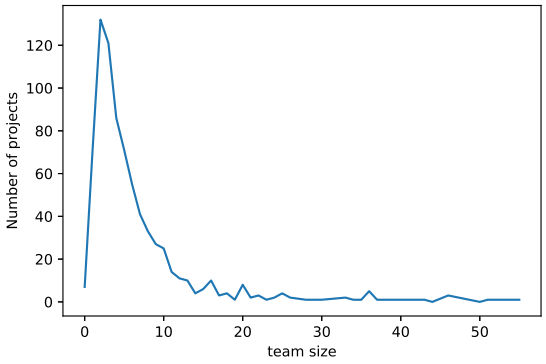
this study may miss enough data from teams with a larger number of team members.

- In addition, some of the columns such as number of OK tests, number of failed tests and number of run tests had about %50 missing value. This volume of missing values may affect the final result as it further reduces the dataset size.
- Furthermore, the projects in TravisTorrent are developed only by Ruby, Java or JavaScript. This set of languages is very much limited and does not let to derive any general conclusion.

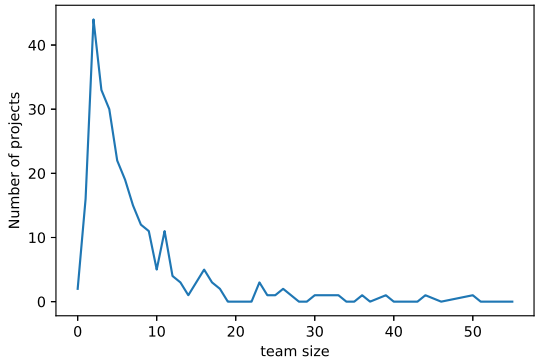
VII. Conclusion and Future Work

In this study, I used the TravisTorrent dataset which is a dataset containing information about projects on GitHub which are using Travis CI. This dataset encloses information about each build job of these projects and gathers its information from the GitHub repository of the project, the Travis CI and GHTorrent. I tried to answer two questions in this research, whether team size affects the productivity of the team and whether the team size and language of the project have any relation. According to this study, I concluded that teams with smaller team size - specifically 7 - would have better productivity as they produced more lines of codes in each build job and their code had a higher quality based on the number of tests were passed or failed and the overall result of their build job. However, I was not able to drive any conclusion about the relation between team size and the primary language of the project.

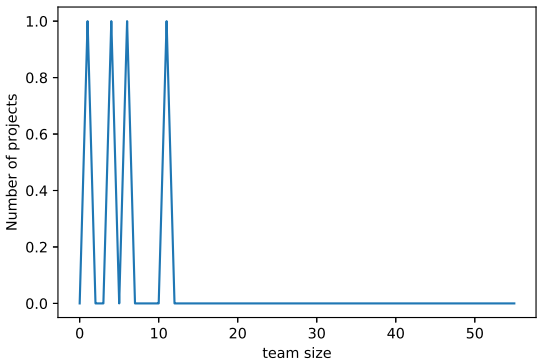
An extension of this work can be conducted where we use a richer dataset. It is possible to manually get the project information from their repositories as we had many missing values and a highly skewed dataset.



(a) Distribution of Ruby projects with respect to team size



(b) Distribution of Java projects with respect to team size



(c) Distribution of JavaScript projects with respect to team size

Fig. 6: Distribution of projects with different languages with respect to team size

REFERENCES

[1] D. Rodríguez, M. Sicilia, E. García, and R. Harrison, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, no. 3, pp. 562–570, 2012.

[2] M. R. Islam and M. F. Zibran, "Insights into continuous integration build failures," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 467–470.

[3] M. Beller, G. Gousios, and A. Zaidman, "Travistorrent: Synthesizing travis ci and github for full-stack research on continuous integration," in *Proceedings of the 14th working conference on mining software repositories*, 2017.

[4] P. C. Pendharkar and J. A. Rodger, "An empirical study of the impact of team size on software development effort," *Information Technology and Management*, vol. 8, no. 4, pp. 253–262, 2007.

[5] ——, "The relationship between software development team size and software development cost," *Communications of the ACM*, vol. 52, no. 1, pp. 141–144, 2009.