2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)

An Empirical Study of Activity, Popularity, Size, Testing, and Stability in Continuous Integration

Aakash Gautam, Saket Vishwasrao, and Francisco Servant
Virginia Tech
{aakashg, saketv02, fservant}@vt.edu

***Abstract*—A good understanding of the practices followed by software development projects can positively impact their success — particularly for attracting talent and on-boarding new members. In this paper, we perform a cluster analysis to classify software projects that follow continuous integration in terms of their activity, popularity, size, testing, and stability. Based on this analysis, we identify and discuss four different groups of repositories that have distinct characteristics that separates them from the other groups. With this new understanding, we encourage open source projects to acknowledge and advertise their preferences according to these defining characteristics, so that they can recruit developers who share similar values.**

***Index Terms*—data mining; clustering; characterizing; continuous integration**

I. Introduction

The identification of the practice and culture of software projects can play an important role in their success — particularly for open source software projects, and for newcomers.

For example, Von Krogh *et al.* [1] posit that the level of activity (lurkers vs. contributors) and the types of activity involved within a repository are important factors that newcomer developers consider to decide whether to start contributing to an open source project. Furthermore, multiple studies (*e.g.,* [2], [3]) identify barriers that may hinder new developers to contribute to a project, such as the difficulty to contribute, lack of information, and complexity of the project modules.

In this paper, we aim to provide empirical evidence for the extent to which different open source projects show a set of characteristics that are normally considered as highly desirable in software development: activity, popularity, size, testing, and stability. Newcomers may assume that, when software projects show some signs of sophistication, they will also show many other desirable characteristics. We posit that such assumption may not necessarily always hold. As such, we focus our analysis in open source projects that show a particular sign of sophistication and know-how in terms of software practices: following continuous integration (CI) practices. While there are some empirical studies of open source projects (e.g., [4], [5]) or hurdles in migrating and maintaining CI systems (e.g., [6], [7], [8]), there are none (to our knowledge) that have described the practices in open source projects that use CI.

We perform a cluster analysis over a collection of software repositories from TravisTorrent dataset [9], which we complement with GitHub Archive [10]. The combination of datasets allows us to obtain information about each project's individual activities as well as its popularity, size and test setup.

Our analysis reveals a set of software project *profiles* that show different levels of activity, popularity, size, testing, and stability. By providing this set of project profiles taken from a large set of real-world projects, we aim to help open source project creators and maintainers to be explicit about these characteristics, and to help new developers become aware of the culture and expectations of their repositories of interest.

II. Data Management

In this study, we use 05-03-2016 TravisTorrent CSV [9] as the main source of data. This dataset had information about build jobs that were triggered through Travis CI system and so, we cleaned records such that each instance in our dataset was a unique commit that triggered a build through the Travis CI system. There could be multiple commits within the build, but in our analysis, we consider the commit that triggered the build. We augmented this dataset with repository-related information such as the number of watchers, number of open issues, and the repository size from GitHub Archive [10].

Table I lists all the properties that were used in the experiment. We focused on using attributes that represented activities related to the build job, the outcome of the build job, and public information about the repositories.

De-duplicating records based on the commit ID, and augmenting it through Github Archives resulted in a total of 94,391 records. Our dataset is available online [11].

III. Clustering

Since we wanted to know the types and characteristics of open source repositories, we conducted a K-means clustering using Weka's [12] SimpleKMeans algorithm, which uses the Euclidean distance measure. Weka internally normalizes the attributes before computing the distances. Furthermore, Weka automatically handles a mixture of categorical and numerical attributes. To find an optimal number of clusters, we measured the sum of squared errors (SSE) at each cluster count. We found that the mean SSE out of 10 tests decreased more slowly after 4 clusters. Thus, we clustered the data in 4 groups. As K-Means is sensitive to initial cluster centers, we tried initializing the clustering with 5–15 random seed centers and chose using 10 centers since it provided the lowest SSE.

IV. Analysis of the clusters

Table I shows the mean values of the attributes within the clusters. We highlight the values that are higher or lower than

978-1-5386-1544-7/17 $31.00 © 2017 IEEE
DOI 10.1109/MSR.2017.38

the mean values for all projects by more than one median absolute deviation (MAD). We use MAD as a robust measure that is more resilient to outliers than the standard deviation. Comparing the values of each cluster, we can identify some distinct characteristics.

A. Cluster 1: The test-priority repositories

These repositories have a test-first attitude with high test density. We hypothesize that their priority on tests help them to identify issues, achieving fair stability in builds — despite having smaller teams and significantly larger code churn rates.

1) Activity: A small number of highly prolific developers contribute to high code churn. They also describe their changes more succinctly than other cluster's developers. The pull requests in this cluster are relatively less verbose (averaging 15.5 words) in the title and description of the pull requests than the overall mean (34.8 words). This cluster actually lies on the lower quarter in terms of verbosity, since the 25th percentile is 16.26. A possible explanation for these short change descriptions may involve the cluster's small team sizes: committers in this cluster possibly assume that their colleagues understand the unexplained changes in pull requests.

A typical build in a repository in this cluster showed a high churn rate (100.2), which lies at the top 12% of all the repositories in the dataset. Correspondingly, builds in this cluster also showed a large number of modified files. Additionally, since these repositories have small team sizes, their number of contributions per team member is highest among clusters — they have highly prolific developers. A typical repository in this cluster gets 351.7 commits from each contributor — the mean for all repositories was 154.

2) Popularity: Surprisingly, despite the strong emphasis that these repositories put on testing, they showed the lowest values in popularity metrics. When compared to the overall mean, they have far fewer forks (29.3 compared to 117.4), fewer watchers (167.4 compared to 533.2), and smaller team sizes (6.2 compared to 10.4).

3) Size: Despite having small team sizes, these number of open issues does not deviate much from the overall mean. In the dataset, we notice a weak positive correlation between the number of members in a team and the number of open issues (Spearman's rho= 0.329; $p < 2.2e\text{-}16$). We posit that the high test density in these repositories facilitates identifying issues in the code. Another unique property of these repositories is that they have a large size (in KB) — among the top 10%.

4) Testing: These repositories have about 3 times the mean test density. The centroid point in the cluster has 8769 lines in the test cases per 1000 executable production source lines of code. The 95th percentile of test lines is 2683.43 so, this group of repositories fall in the top 5% of the entire population in terms of number of test lines. This group also has larger number of test cases and test asserts per 1000 executable production source lines of code both of which lie above the 95th percentile. A previous study had found weak correlation (Spearman's rho = 0.207) between the number of developers and test cases [13] but this group is anomalous to that finding.

5) Stability: We find that the builds triggered in these repositories have a success rate of 0.73, which is comparable to the overall mean of 0.72. We hypothesize that despite having largest source code churn rate, the builds are fairly stable because of the high test density.

B. Cluster 2: The popular, high-flying repositories in the block

This group of repositories lies on the other extreme of the testing priority spectrum where they are large and popular, but tend to set low-priority to testing.

1) Activity: The mean churn rate within the cluster is 80.4 whereas the population mean is 88.4 from which we can infer that they make smaller changes in a typical commit. They also tend to have pull requests that have most words in the title and message (45.8 words compared to 34.8 overall mean). We posit that this verbosity is necessary to helps others to understand because they tend to have a large team size.

Furthermore, the commits on these repositories are on a subset of files that are frequently updated whereas other files are not modified often. The files that are changed have a mean of 28 unique commits over the last three months whereas across the repositories the mean is 23 and the 75th percentile is 25. Since developers are mostly working on a fixed subset of the code, they are not adding tests cases. This can be seen from "tests added" attribute in Table I where the number of tests added is 0.00 in contrast to the 0.04 mean across repositories.

2) Popularity: The repositories within this cluster have the largest number of watchers. They also have the largest number of forks with an mean of 453.75 forks. The overall mean is 117.41 and the 90th percentile is 314 which shows that these repositories are extremely popular in GitHub.

The group of repositories in this cluster also have the largest team size with mean size of 22.6. The overall mean team size of the population is 10.4 and 75th percentile is 12.0. Probably because they have a large team size, this group of repository has the lowest per user commits among the entire population (72.1 versus the overall mean of 154.0) signaling that at when team sizes are large, some developers may contribute less. Similarly, this group of repositories have the least amount of source code line changes in a typical commit.

3) Size: The repositories within this cluster lie in the top 10% of the largest code bases in Github that uses Travis CI system. They have a mean of 64802.13 executable production source lines of code whereas the overall mean is 24762.68 and the 90th percentile is 57356 lines.

These repositories have a large number of open issues that lie among the top 15% among all the repositories. Since they have the largest code base of all the groups of repositories, it is intuitive to expect larger number of issues in the code. In addition, we hypothesize that having a large number of watchers and forks helps in identifying and reporting issues in the project despite having a low number of test cases.

4) Testing: We see that the repositories within this cluster do not prioritize tests like the repositories in cluster 1. The mean for test lines per 1000 production lines of code in this

TABLE I

MEAN VALUES FOR THE 25 ATTRIBUTES OF THE 4 CLUSTERS IDENTIFIED IN OUR K-MEANS CLUSTER ANALYSIS. WE HIGHLIGHT VALUES LOWER AND HIGHER THAN THE OVERALL (FULL DATA) MEAN VALUES BY ONE MEDIAN ABSOLUTE DEVIATION (MAD).

Attribute Type	Attribute	Full Data (94257)	Cluster-1 (12136)	Cluster-2 (16979)	Cluster-3 (51924)	Cluster-4 (13218)
Activity per Build	# files added	1.0	1.1	0.8	0.9	1.2
	# files deleted	0.5	0.7	0.8	0.3	0.5
	# files modified	4.5	5.1	4.1	4.5	4.4
	# src files changed	4.4	4.7	4.5	4.3	4.1
	# doc files changed	0.5	0.6	0.4	0.4	0.7
	# other files changed	1.0	1.6	0.8	1.0	1.1
	# code lines changed	88.4	100.2	80.4	88.6	87.1
	# tests added	0.04	0.02	0.00	0.06	0.01
	# tests deleted	0.2	0.2	0.1	0.2	0.2
	# test lines changed	32.4	30.9	31.3	33.1	32.5
	# commits in the build	2.5	2.6	2.5	2.4	2.6
	# commits on files touched	23.2	23.5	28.6	22.3	19.4
	# contributions per member	154.0	351.7	72.1	147.0	105.0
	# words in pull request	34.8	15.5	45.8	35.8	34.6
	% by core team member	0.82	0.56	0.80	0.88	0.81
Popularity	# watchers*	533.2	167.4	1,852.1	180.4	561.0
	# forks*	117.4	29.3	453.8	33.9	94.4
	# team members	10.4	6.2	22.6	8.2	7.5
Size	sloc	24,762.7	20,179.1	64,802.1	16,180.7	11,251.3
	# open issues*	22.5	23.4	52.3	10.5	30.1
	repository size (KB)*	24,503.9	42,921.2	26,256.2	18,778.5	27,834.2
Test Suite Size	# test cases per kloc	230.2	484.8	120.3	215.3	196.4
	# test lines per kloc	3,202.1	8,769.8	1,547.5	2,650.5	2,382.0
	# assert cases per kloc	506.4	1,086.7	270.3	470.3	418.2
Stability	% of build success	0.72	0.73	0.74	0.76	0.51
Example projects			'jruby/warbler', 'rspec/rspec-rails'	'resque/resque', 'mitchellh/vagrant'	'rdoc/rdoc', 'alecgorge/jsonapi'	'opal/opal', 'lsegal/yard'

Attributes marked with * were obtained from GitHub Archive. The remaining attributes were obtained from TravisTorrent.

cluster is 1547.5 whereas the overall mean is 3202.06. Likewise, compared to the overall mean, this cluster of repositories have fewer test cases (120.32 compared to 230.22) and test asserts (270.3 compared to 506.4) per 1000 lines of code.

5) Stability: The builds were successful 74% of the time which is comparable to the overall mean of 72%.

C. Cluster 3: The average Joes

This is the largest of all four clusters with 55% of the instances falling under this cluster. Two interesting observations of the cluster are that the core team members tend to be more involved in the projects than volunteers and that these repositories have a significantly low number of open issues.

1) Activity: The repositories within this cluster have average (close to the overall mean) values across attributes related to activity: they have an average amount of source code churn rate in a typical commit and an average test code churn rate. They have typical pattern when it comes to number of words used in the pull request title and message. The contribution amount from each team member is also average when compared to the entire set.

2) Popularity: These repositories have an average number of watchers but a significantly lower number of forks signifying that these repositories may not be popular among volunteer contributors. An average repository in this cluster has 33.9 forks whereas the overall population mean is 117.4 forks per repository. Probably because these repositories are not popular among larger contributors, we notice that the commits made on these repositories tend to be more by core team members.

3) Size: These repositories tend to have below average number of open issues with 10.5 on an average whereas the overall mean for all the repositories is 22.5. As mentioned above, there was a weak positive correlation between number of team members and the number of open issues which we see in these set of repositories. Since these repositories have lower number of members in the team (8.2 compared to 10.4 overall), they have a lower count of open issues.

4) Testing: This group of repositories have an average number of test cases (215.3 compared to overall mean of 230.2) and asserts per 1000 production lines of code (470.3 compared to 506.4 overall), and an average number of test lines (2650.5 to overall mean of 3202.1).

5) Stability: The build jobs triggered in this clusters tends to be successful 76% of the time which a bit more than the overall mean. However, the difference is not significant.

D. Cluster 4: The risk-taking adventurers

On average, half of the builds triggered in this group of repositories tend to fail. The uniqueness of the repositories with regards to the activity is that the changes tend to be made on files that have been relatively less worked upon before. The repositories in this cluster also have fewer lines of code in production.

1) Activity: We call them "adventurers" because the developers in this group of repositories tend to make changes on files that are relatively less frequently changed. Typical files they change in a commit has a mean of 19.4 unique commits over the last 3 months whereas the population mean for this is 23.2. This implies that developers in these repositories change files that have not been changed recently.

2) Popularity: These repositories have a number of watchers (561) higher than the overall dataset mean (533.2). However, they have a relatively lower number of active team members compared to other clusters. The mean team size is 7.51 whereas the overall mean is 10.4. The number of commits per contributor is below the overall mean (105.0 versus 154.0) and so is the amount of changes in a typical commit.

3) Size: Furthermore, these repositories have a relatively low number of executable production lines of code. The mean number of lines in the repositories of the cluster is 11251.3 lines of code whereas the overall population mean is 24762.7. However, despite having fewer lines of code in production, these repositories have a relatively larger number of open issues. They have a mean of 30.1 open issues whereas the overall mean is 22.5 and the 75^{th} percentile is 26.

They have smaller code base and team size, both of which were weak but positively correlated to number of open issues. Yet, they have the largest number of open issues. This suggests that these projects are either new projects or "hobby projects".

4) Testing: These repositories have close to the overall mean values with regards to the test suite. They have 2382.0 test lines per 1000 lines of production code which is comparable to the overall mean of 3202.1. Likewise, their values are close to the overall mean for test case density and assert statement density.

5) Stability: We call them the risk-taking ones because they are they only ones who are characterized to have frequent build failures. 51% of the builds that have been triggered in this cluster have succeeded whereas the overall mean lies at 72%. We posit that these repositories practice a "build fast, fail fast attitude" during development.

V. Conclusion

In this paper, we provided a characterization of open source software projects that follow continuous integration practices. We characterized these software projects based on their activity, popularity, size, testing, and stability. By using cluster analysis, we provided empirical evidence that improves our understanding of the practices followed by open source software projects in the aforementioned areas.

Such better understanding the practices that software projects may follow in terms of activity, popularity, size, testing, and stability can benefit practitioners in a number of ways. Newcomers could make a more informed decision about which software project to join. Older contributors and maintainers of open source projects could make it easier to know these characteristics by defining them more explicitly.

In particular, our analysis revealed practices that open source project creators and maintainers could choose to make explicit or to take actions about them. Some examples of such practices are: (1) Giving high priority to the creation and maintenance of test cases which could be made explicit for newcomers to abide to, (2) providing expected levels of verbosity in pull-request titles and messages which newcomers would benefit from knowing beforehand, (3) having high numbers of lurkers (those who watch but do not contribute) which maintainers may choose to put efforts in place to try to engage them, (4) having highly involved core team members so newcomers can adjust expectations for their chances to contribute, and (5) stating cultural values *e.g.,* "build fast, fail fast", which could be helpful in retaining newcomers [2].

Based on the work in this paper, we plan to study some of the observed characterizations in more depth. Some interesting avenues of future research involve the design of novel onboarding portals for newcomers that are customized to the specific culture of the repository, the study of project management styles for different project profiles, or the study of barriers for developer migration across project-profile borders.

References

[1] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[2] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "Social barriers faced by newcomers placing their first contribution in open source software projects," in *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*. ACM, 2015, pp. 1379–1392.

[3] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in github: the role of prior social links and language experience," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 817–828.

[4] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 3, pp. 309–346, Jul. 2002. [Online]. Available: http://doi.acm.org/10.1145/567793.567795

[5] A. Capiluppi, P. Lago, and M. Morisio, "Characteristics of open source projects," in *Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on*. IEEE, 2003, pp. 317–327.

[6] D. Ståhl and J. Bosch, "Modeling continuous integration practice differences in industry software development," *Journal of Systems and Software*, vol. 87, pp. 48–59, 2014.

[7] H. H. Olsson, H. Alahyari, and J. Bosch, "Climbing the" stairway to heaven"–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software," in *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*. IEEE, 2012, pp. 392–399.

[8] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Communications of the ACM*, vol. 48, no. 5, pp. 72–78, 2005.

[9] 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.

[10] I. Grigorik, "The github archive," https://www.githubarchive.org/.

[11] A. Gautam, S. Vishwasrao, and F. Servant, "An Empirical Study of Activity, Popularity, Size, Testing, and Stability in Continuous Integration," May 2017. [Online]. Available: https://doi.org/10.5281/zenodo.439362

[12] G. Holmes, A. Donkin, and I. H. Witten, "Weka: A machine learning workbench," in *Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems, 1994.* IEEE, 1994, pp. 357–361.

[13] P. S. Kochhar, T. F. Bissyandé, D. Lo, and L. Jiang, "An empirical study of adoption of software testing in open source projects," in *Proceedings of the 13th International Conference on Quality Software (QSIC 2013)*. IEEE, 2013, pp. 103–112.