IC²S² 2018 Submission: Predicting the Performance of Software Development Teams on GitHub

4th International Conference on Computational Social Science IC²S²
July 12-15, 2018
Northwestern University’s Kellogg School of Management
Evanston, IL. USA

Anonymous

keywords: GitHub, team performance, supervised learning

1 Extended Abstract

Although there are many prescriptive models of effective teamwork, accurately predicting team performance remains an open research challenge. In particular, modeling the behavior of virtual teams in which the members are dispersed geographically is of great importance in the modern workplace. The aim of our research is to use machine learning to predict the performance of virtual teams over long time periods. Our approach leverages features from the Espinosa et al. research on teams [1] in which they show that familiarity with team and task is beneficial to team performance.

To study the performance of virtual teams, we use data from software developers who collaborate using GitHub\(^1\), a popular software repository and collaborative development tool [2]. Software development teams can store their open source software as a repository on GitHub and use the platform to support team communications such as reporting bugs and requesting new features. GitHub repositories typically have several collaborators working on the project as a virtual team. Collaborators make changes to the repository by committing their content. Any GitHub user can contribute to a repository through sending a pull request. Collaborators review pull requests and decide whether to accept or reject the requests [2]. Contributors can attach a comment to their commit or pull request to communicate a message. Unlike other communication channels, the list of public events is available to everyone; this transparency makes GitHub a unique platform to study team behavior [3].

In our project, we use team familiarity, task familiarity, and team complexity in order to predict the performance of teams, as measured by the popularity of their projects. Team familiarity is the knowledge of team members about each other; task familiarity is the knowledge about the task itself [1]. To measure team familiarity among collaborators of a GitHub repository, we investigate the past projects and communications of every pair of collaborators working on that repository. For example, if two collaborators have shared project history, or they have mentioned each other in comments, we score them as being more familiar with one another. To infer task familiarity on GitHub, we compare the past experiences of developers with characteristics of the repository such as programming language and topic. For instance, the number

\(^1\)https://github.com/
of lines of code a developer has written in a particular programming language demonstrates
the fluency of the developer in that language. Thus, task familiarity is higher for a developer
who is fluent in the programming languages of the repository. Moreover, repositories can be
assigned a topic in GitHub (e.g. machine learning); developers are likely to be more familiar
with topics that they have past experience in. Task and team complexity also affects the per-
formance of teams. Task complexity is the complexity of the task itself independent of team, for
example, the number of lines of code in GitHub repositories. Team complexity is derived from
challenges of communicating and coordinating a team: larger teams are more complex because
of the increased coordination demands.

In this study, we assume the popularity of a repository can be used to measure the perfor-
mance of the virtual team of software developers. In GitHub, users show their interest in a
project by *starring* it. The number of *stars* of a project is the most conventional way to mea-
sure its popularity. In addition, GitHub users can *fork* a repository to own a copy of it or to
make changes to the content and submit their contributions via a *pull request*. Thus, popular
repositories have a high number of *forks* and *pull requests*. In this study, we predict the number
of *stars*, *forks*, and *pull requests* to evaluate team performance.

In order to predict popularity of GitHub repositories, we extract features of teams based on
team/task familiarity and complexity and then train a supervised learning model. Since GitHub
contains a wide variety of team structures, using a single classifier is less likely to have a high
predictive power. Therefore, we cluster the repositories to create a set of sub-problems and
then we train one classifier per cluster. Each cluster contains teams that are more similar to
each other but diverse enough to avoid overfitting classifiers.

The dataset that we are using for this study contains data of all users and repositories created
before September 2017 and chronicles event and activity data from January 2015 to August
2017. GitHub contains various types of repositories such as individual projects, and collab-
orative development of documents. For this research, we filter out repositories that do not
represent software development teams. The familiarity and complexity of teams and tasks have
been shown to play an important role in the performance of software engineering teams and
hence these features can be leveraged to train machine learning classifiers. We believe that
early prediction of team performance is potentially valuable for any organization that makes
use of virtual software engineering teams.

References

[1] J Alberto Espinosa, Sandra A Slaughter, Robert E Kraut, and James D Herbsleb. Familiar-
ity, complexity, and team performance in geographically distributed software development.

[2] Antonio Lima, Luca Rossi, and Mirco Musolesi. Coding together at scale: GitHub as a
collaborative social network. In *Proceedings of the International Conference on Web and
Social Media*, 2014.

[3] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in GitHub:
transparency and collaboration in an open software repository. In *Proceedings of the ACM