2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) | 978-1-6654-1924-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/WIIAT50758.2020.00027

Analyzing the Productivity of GitHub Teams based on Formation Phase Activity

Samaneh Saadat
Department of Computer Science
University of Central Florida
Orlando, FL US
ssaadat@cs.ucf.edu

Olivia B. Newton
School of Modeling, Simulation, and Training
University of Central Florida
Orlando, FL US
olivianewton@knights.ucf.edu

Gita Sukthankar
Department of Computer Science
University of Central Florida
Orlando, FL US
gitars@eecs.ucf.edu

Stephen M. Fiore
Department of Philosophy and
School of Modeling, Simulation, and Training
University of Central Florida
Orlando, FL US
sfiore@ist.ucf.edu

***Abstract*—Our goal is to understand the characteristics of high-performing teams on GitHub. Towards this end, we collect data from software repositories and evaluate teams by examining differences in productivity. Our study focuses on the team formation phase, the first six months after repository creation. To better understand team activity, we clustered repositories based on the proportion of their work activities and discovered three work styles in teams: *toilers*, *communicators*, and *collaborators*. Based on our results, we contend that early activities in software development repositories on GitHub establish coordination processes that enable effective collaborations over time.**

***Index Terms*—GitHub, team formation, software engineering productivity**

I. Introduction

GitHub[1], a social coding platform used by self-organized teams for open source software development (OSSD), is increasingly popular among researchers as a source for data mining of software repositories. Recent work in this research domain includes the prediction of repository popularity based on historical data [1], [2], contribution choices of developers [3], and file changes based on communication networks [4]. Many of these studies have collected data from mature, large software repositories to predict outcomes of interest. To add to this body of work, in the present study we examine a variety of repositories from the early stages of creation to better understand the relationship between processes in team formation and subsequent performance. The paper addresses the following research questions:

- **RQ1:** how does team size and work centralization differ during the formation period?
- **RQ2:** how is work style related to team performance?
- **RQ3:** which team activity features are closely related to performance?

In group research, team formation describes the foundational processes that support subsequent team interactions. Our conceptualization of team formation is based on Kozlowski et al.'s normative model of team development [5]. This model defines four continuous and overlapping phases: team formation, task compilation, role compilation, and team compilation. In the team formation phase, team members begin to learn about each other and establish norms and shared goals through information-seeking behaviors, and thus develop rudimentary shared knowledge structures about the team. In the context of social coding, and in line with traditional team performance studies, some research suggests that team formation in GitHub is related to previous collaborations and social ties [6].

As described in research on teamwork in science and technology, collaboration draws from both social or team factors, as well as technical or task related factors to meet objectives. More specifically, such work requires an effective integration of both teamwork and taskwork to achieve team goals [7], [8]. Less research has examined the importance of the taskwork processes established during team formation and its links to team productivity over time. To redress this gap, we analyze features characterizing the team formation phase in GitHub repositories. We examine differences in developer activity-based features extracted during the team formation phase in groups of high- and low-performing software development teams.

II. Related Work

Several threads of research on software development in GitHub are most relevant to the present work. In particular, we survey research on coordination processes, cognitive artifacts, and individual and group performance evaluation in social coding platforms. Research on collaborative work in software repositories on GitHub typically examines the activity profiles of developers and/or their responses to surveys and semi-structured interviews to better understand important processes and outcomes of interest. This body of work examines differences between contributors, including their varying influence

[1]https://github.com/

978-1-6654-1924-6/20/$31.00 ©2020 IEEE
DOI 10.1109/WIIAT50758.2020.00027

on each other [9], and their level and length of participation [10]–[12].

1) Coordination Processes: Coordination can be either implicit or explicit but it generally describes the means through which team members organize their activities towards a shared goal [13]. These processes shape the team's understanding of interdependencies for individual and collaborative output, leading them to develop strategies for the effective integration of multiple contributions. In GitHub, implicit coordination takes the form of information-gathering behaviors for the maintenance of task and developer awareness, for example, by reviewing code artifacts or following a developer to receive notifications about their activity [14]. Explicit coordination is observed in discussions in comments linked to commits, pull requests, and issues. In a study of highly-watched repositories, research finds that, as team size increases, explicit coordination also increases [15]. Other work has similarly found that team size influences coordination structure such that small teams can function without the aid of a designated coordinator while larger teams require top-down management for the coordination of their contributions [10]. Indeed, centralization and delegation of work in software projects is critical for success [16] and has been linked to improved issue support quality in software repositories on GitHub [17].

2) Cognitive Artifacts: In the cognitive and social sciences, cognitive artifacts are argued to be integral to effective cognitive and collaborative processes. Cognitive artifacts are "artificial device[s] designed to maintain, display, or operate upon information in order to serve a representational function" [18] and can provide valuable insights about processes and outcomes in distributed cognitive systems [19]. Fiore and colleagues argue that these forms of externalized cognition are central to team cognition, particularly when collaboration occurs through a hybrid human-machine team [8]. In the present context, software repositories contain a multitude of artifact types, and whether they be code- (i.e., work output) or communication-based (processes), they are devised to support elements of team cognition. Issue trackers in social coding platforms are official documentation artifacts. Issue comments, in particular, are artifacts of collaborative problem-solving processes. Furthermore, through the use of features like issue labels, developers can organize these artifacts. These features, when consistently used, may support information-gathering activities of current and potential contributors as developers report that they prefer to evaluate task-relevant information in favor of disrupting another working developer [14].

3) Team Performance Evaluation in Social Coding Platforms: Performance measurement of software development teams in GitHub is challenging as there are a number of indicators that are related to performance outcomes. For example, developers report that they measure the success of a software project in terms of the number of contributors and contributor growth [20]. Accordingly, studies of software development on GitHub often emphasize the importance of a project's popularity and growth as indicators of its contributors' performance. Indicators of repository popularity and growth include its number of stars and forks (copies), respectively. Other research identifies outcomes that can serve as indicators of team performance, including integrator productivity and code quality [21], and issue support quality [17]. We focus on overall work activity as the primary indicator of team performance. In this, we examine several different aspects of collaborative work in GitHub: work centralization and work style.

III. Method

Based the prior work described in the previous section, we apply the following criteria to the data collection procedure. The *team formation phase* was defined to be the first six months following the creation of the repository. We observed the activities of the repositories for a 13 month period after the repositories' creation to assess team productivity; this time period is termed the *evaluation period*.

A user is identified as a *team member* based on their participation in the repository during a specific time period (formation phase or evaluation period). A user is considered a member of the team if they have completed at least one of the following during the team formation phase: one push event; five accepted pull requests; ten issue comments; or ten pull request review comments.

All of the following are treated as work events: (1) push; (2) merged pull request; (3) issue comment; and (4) pull request review comment. Furthermore, we consider push events as work output, issue comments and pull request review comments as explicit coordination, and merged pull requests as collaborative work.

We examined performance or productivity of teams by aggregating work of the evaluation period team at the repository-level. To view this as a measure of team effectiveness, we averaged productivity over the number of members within a repository. In other words, we assessed the average amount of work completed by individual team members 13 months after the creation of the repository.

A. Data Set

The data set used in this study contains all GitHub events from January 2016 to June 2017. From this data set, we selected software repositories created in January 2016 and included only those that had more than 20 work events and at least two members in the team formation phase. We measured size of the teams, once more, in the evaluation period and removed the repositories with less than two members. Our final data set included 20,370 active repositories and the 59,178 unique GitHub users contributing to those repositories in the evaluation period.

B. Team Feature Extraction

We extracted team features based on event data in the team formation phase.

- **Relative frequency of work events** is the ratio of the number of each work event to the total number of work events. This feature allows us to evaluate the type(s) of work undertaken during team formation.

- **Work events per person** is the number of each work event divided by the number of team members. We used this feature to evaluate the amount of work, or productivity of contributors.
- **Burstiness** measures the temporal correlation of activities within a team and defined as equation 1; where μ_τ and σ_τ are mean and standard deviation wait times $P(\tau)$.

$$Burstiness = \frac{\sigma_\tau - \mu_\tau}{\sigma_\tau + \mu_\tau} \quad (1)$$

  An analysis of burstiness reveals the presence of increased, synchronized activity in a team and is linked to effective collaborations and improved team performance [22].
- **Issue labeled proportion** is the proportion of issues in the repository that are labeled. Because issue labels are one way that developers can organize their work and communication on GitHub, we use this feature to evaluate the use and organization of cognitive artifacts by the team.
- **Team size** is the number of team members in the repository during the team formation phase. This feature was used to assess the importance of the team's size early on in its development.

IV. Results

A. Team Performance Evaluation

First, we examine the characteristics of the work event data during the performance evaluation period. Performance is defined as the amount of the work teams completed per person in the evaluation period. Figure 1A shows that the distribution of work per person is a heavy-tailed distribution. A log transformation on work per person was applied to decrease the variability of this measurement. The transformed distribution is a normal distribution with a mean and variance of 2.05 and 0.37, respectively (Figure 1B). We considered the log transformed values of work per person representative of team productivity. To categorize teams based on their productivity, we used the median of team productivity (= 2.0) as our threshold and labeled teams as high-performing if their productivity was greater than the threshold and low-performing otherwise.

Our unit of analysis is based, in part, on the size of the team during the performance evaluation period. Figure 2 provides the histogram of team sizes in our data during the performance evaluation period.

1) Evaluation Period Team Sizes: For additional analyses, we grouped teams based on their size: teams with three or fewer members were considered small, teams with between four and six members were considered medium, and teams with seven or more members were categorized large. Vasilescu et al. [11] used higher thresholds to categorize teams in software repositories on GitHub. However, given the short life span of the repositories we studied, we determined these thresholds were more appropriate to capture differences in processes and outcomes across team sizes. The number and the proportion of teams in each group is provided in Table I.

Group	Size	Proportion	Frequency
Small	[2, 3]	70%	14261
Medium	[4, 6]	24%	4951
Large	[7, inf)	6%	1157

TABLE I
PROPORTION OF TEAMS BY TEAM SIZE.

In the performance evaluation period, more than half of the repositories in our data set were maintained by small teams and less than a quarter were maintained by medium-sized teams. Large-team repositories made up less than five percent of the data.

Figure 3A shows the performance of teams of different sizes; teams with two members have slightly higher performance compared to teams with three, four, or five members. Given that most contributors do very little work in GitHub repositories [23], this finding is not particularly surprising. Figure 3B shows the number of high- and low-performing teams in each team size category. Interestingly, there are more high performers than low performers in the large team size group. This suggests that, in our data set, large teams are likely engaged in more frequent interactions and potentially more effective collaborations leading to the production of more communications and code artifacts (i.e., more work events per person).

B. Work Centralization

To examine the distribution of work underlying the performance differences observed across team sizes, we calculated the amount of work centralization in teams using the Gini coefficient. The Gini coefficient is a measure of inequality, where 0 represents perfect equality and 1 represents maximum inequality. We use it to analyze the inequality of work events per person. More specifically, a higher Gini coefficient suggests that a smaller set of team members do most of the work; that is, work is centralized to a fewer number of members in the repository. Figure 4 provides distributions for the Gini coefficient values of teams in the performance evaluation period. High-performing teams have a higher Gini coefficient compared to low-performing teams, regardless of team size. The difference between the Gini coefficient of low- and high-performing teams is greater in large teams compared to small and medium teams. This finding provides support for the importance of work centralization for performance in OSSD in GitHub [17]. Our results suggest that work centralization is linked to increased overall productivity in software repositories and is particularly important for collaborations in large teams.

C. Work Style Clusters

GitHub users and teams behave differently and clustering them can express their diversity [24]. To discover various work styles on GitHub, we clustered teams using the *k-means* clustering algorithm applied to the proportion of different types of work events. We tested various k values for the *k-means* clustering and observed that $k = 3$ generates the minimum number of clusters that are meaningfully distinct

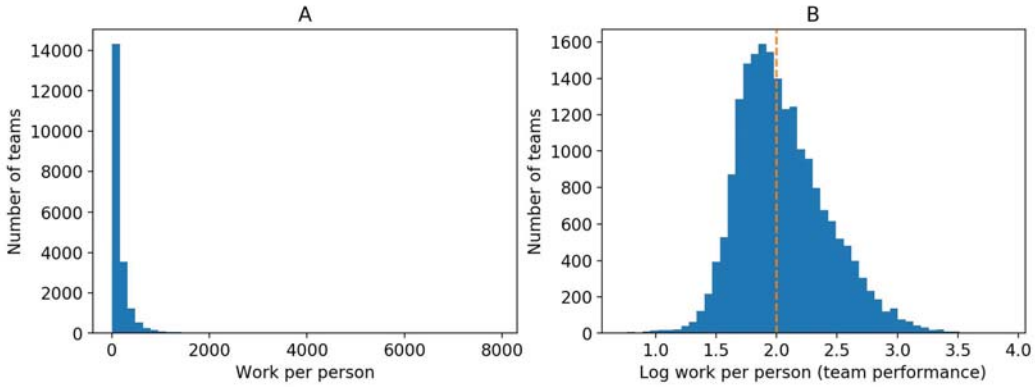

Fig. 1. Distribution of work per person per month (A) and transformed distribution of work per person per month (B).

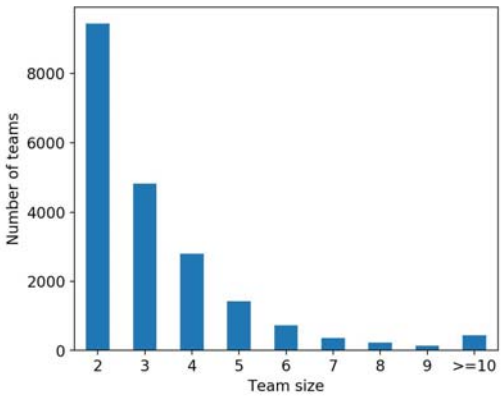

Fig. 2. Team sizes in the performance evaluation period.

from each other. The number of teams in each of the three work style clusters is provided in Table II, and the cluster centroids are plotted in Figure 5. Based on the proportion of events in work style clusters, we labeled them as: *toilers*, *communicators*, and *collaborators*. Figure 6 illustrates work style of users and demonstrate their differences.

Toilers produce a higher proportion of push events compared to *communicators* and *collaborators* but have limited, if any, communication taking place in GitHub. This may reflect a failure to engage in explicit coordination but it is also possible that these teams use alternative communication channels, like Slack [2] and Discord [3], to coordinate their contributions, as past research has shown [25]. Although *toilers* focus on code contributions, they generally do not accept external contributions, unlike the other two work style groups. *Communicators* produce a higher proportion of comment events, and *collaborators* produce a higher proportion of pushes and merged pull requests.

For additional analysis, we calculated the average number of each event type for different work styles (Table II). Overall, *collaborators* have the highest level of productivity in terms of internal and external code contributions (pushes and merged pull requests, respectively) whereas *communicators* have the highest level of productivity in terms of explicit coordination. Although *toilers* devote the majority of their effort to code contributions, they still have, on average, a lower number of pushes compared to *collaborators*.

1) Work Style and Performance: Figure 7 shows the number of high- and low- performing teams in each work style cluster. Nearly 75% of the teams in our data are in the *toilers* cluster. Of this subset, the majority of them were low-performing teams. This suggests that a lack of communication within GitHub is indicative of poor performance. Although *toilers* could be communicating outside the GitHub ecosystem, the fact that they did not perform relatively well suggests that direct communication within GitHub (e.g., issue comments) may help overall team performance. This is illustrated when we look at the *communicators* and *collaborators* clusters. For the *communicators*, we see a relatively more equal distribution of comments proportional to the amount of work done. For the *collaborators*, there is relatively more communication compared to the toilers. And, proportionate to the amount of other activity, they also accept more pull requests than any of the other clusters. In the *communicators* and *collaborators* clusters, the majority of teams are high performers. This effect is more pronounced for large teams and, to a lesser extent, medium teams compared to small teams.

D. Team Features

Our goal is to understand the impact of team formation phase on the evaluation period. Figure 8 illustrate the team formation phase features that are linked to performance of teams in the evaluation period; that is, the features predictive of success. Across each of these, t-tests show that the *p-value* is less than 10^{-10} in all features represented in figure 8. More specifically, these features identified during the formation phase, and indicative of the type of work norms created during formation, are significantly related to higher performance during the evaluation phase.

[2]https://slack.com/
[3]https://discordapp.com/

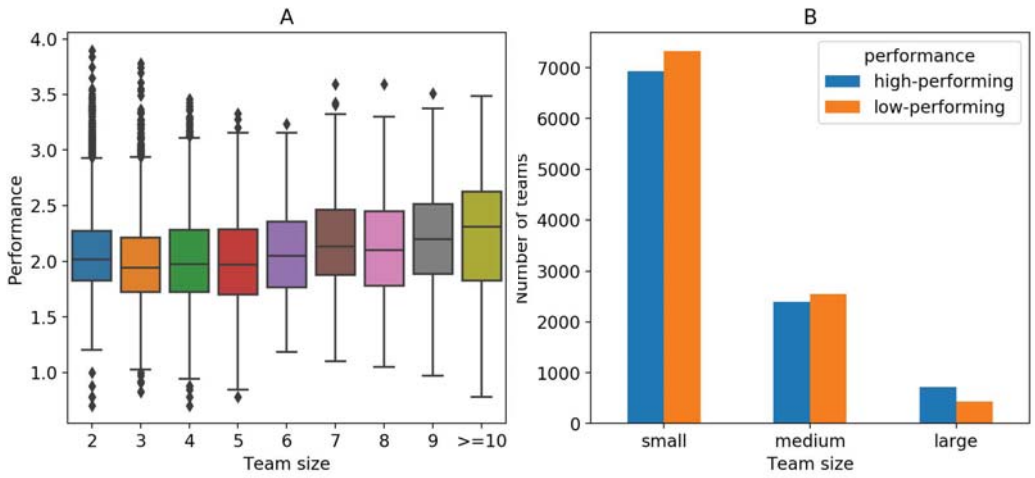

Fig. 3. Performance of teams by size (A) and high- and low-performing teams by team size groups (B).

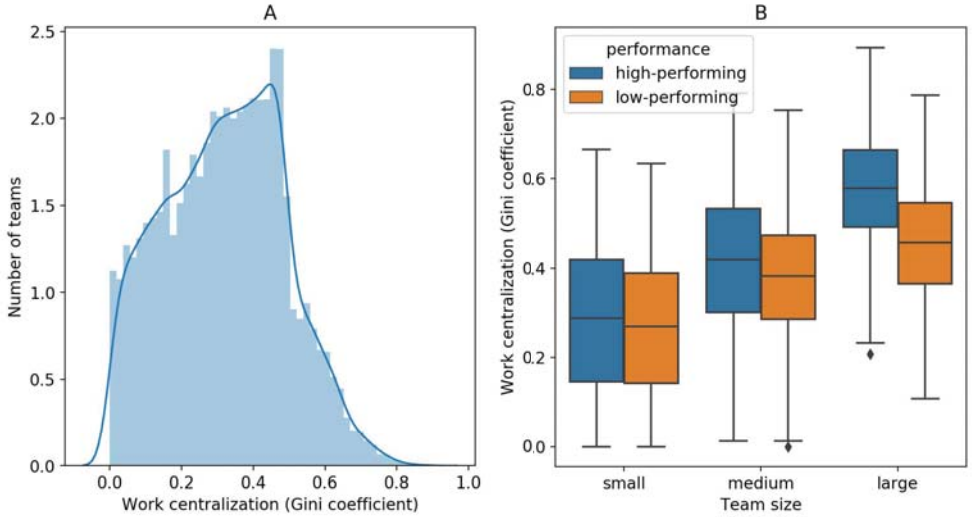

Fig. 4. Distribution of Gini coefficients for all teams (A) and distribution of Gini coefficients by team size and performance (B).

Work style cluster		*Toilers*	*Communicators*	*Collaborators*
Number of teams		15264	3219	1886
Push	mean	79.61	68.61	91.60
	std	118.19	143.40	128.98
Merged PR	mean	0.55	14.93	23.12
	std	4.53	47.79	45.15
Issue comment	mean	0.86	109.72	17.31
	std	4.61	477.73	35.62
PR review comment	mean	0.09	41.12	5.65
	std	1.51	142.60	19.54

TABLE II
THE MEAN AND STANDARD DEVIATION OF DIFFERENT WORK EVENTS FOR TOILERS, COMMUNICATORS, AND COLLABORATORS.

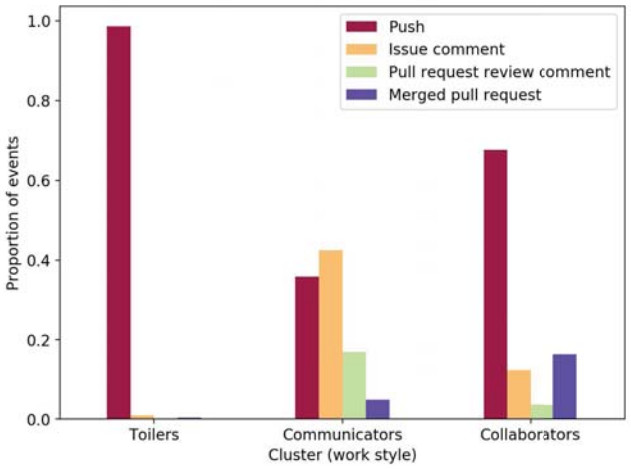

Fig. 5. Proportions of events by work style cluster.

1) Types of Activity: Large-team repositories have a higher proportion of coordination events (i.e., issue comments in figure 8B) and collaborative work events (merged pull requests in figure 8C). Conversely, these repositories have a lower proportion of internal contribution events (pushes in figure 8A). High-performing teams have a lower proportion of push events and a higher percentage of coordination events. In other words, high-performing teams exhibit higher levels of coordination. Issues in GitHub repositories can be used to develop an understanding of the problem at hand and the work needed to resolve the problem, and also offer a space for the discussion and evaluation of potential solutions and strategies to address needs of users and the project. The larger proportion of issue comments in high-performing teams shows that dissections of issues are helpful in improving the productivity of the teams. In contrast, low-performing teams have a lower proportion of coordination events and a higher proportion of contribution events. This suggests that these teams are primarily focused on their work output and spend less time on coordination and communication. Fitting with the team cognition literature [26], this may lead to a failure to evaluate alternatives and the premature selection of solutions.

2) Burstiness: Overall, we observe a higher amount of burstiness in high-performing teams and a lower amount of burstiness in low-performing teams. Interestingly, this difference in high- and low-performing teams is consistent across team sizes. This suggests that high-performing teams, in general, interact more frequently and their activities are highly-synchronized.

3) Issue Labels: Figure 8F shows that proportion of labeled issues is generally higher for high-performing teams than it is for low-performing teams. This is particularly true for medium and large teams and less noticeable in small teams. Again, this finding supports the claim that larger teams require more coordination mechanisms to function effectively. This suggests that, as a classification system for artifacts, the consistent use of issue labels may scaffold collaborative problem-solving processes [8] and thus support the productivity of software development teams in GitHub. More specifically, in line with team cognition theory on complex problem solving [27], issue labels provide support for information-gathering and knowledge-building activities of current and prospective team members.

V. Limitations and Future Work

Given the complicated nature of OSSD, software repositories have different development models. As described, in the fork & pull model, developers fork the main repository, push their changes to the forked repository, and when their task is completed, the developer sends a pull request to the maintainers of the main repository and ask them to accept their changes. In this study, we did not consider the additional push events performed on the forked repositories. This could lead to a lack of accounting for a portion of push events in fork & pull-based repositories.

Another issue is how bots influence team collaborative activities. Bots are increasingly common in OSSD [28]. We did not identify and remove bots from our data set. It is possible that some differences in productivity may be influenced, in part, by bot activity. Some developers also use multiple accounts on GitHub; we did not apply any aliasing to usernames to identify multiple accounts used by a single user.

Finally, we did not consider specifics of work centralization across users. In future work, we plan to examine performance differences in core and periphery contributors as it relates to work centralization and overall productivity. The core and periphery are qualitatively distinct in GitHub, contributing at different rates and exhibiting different levels of interdependency [10], [29]. Additionally, we plan to develop and evaluate a combinatory metric for team performance in GitHub based upon factors such as work centralization and other factors indicative of efficiency and effectiveness. This will allow us to holistically examine the multidimensionality of team performance in social coding platforms.

VI. Conclusion

We studied the impact of team formation phase activities on the performance of the team in the future. Our results show that coordination processes manifest themselves as particular GitHub features, and that the diversity of work in GitHub repositories can be linked to productivity. This work makes both theoretical and practical contributions. At the theoretical level, our findings contribute to the body of work in team cognition by showing how features relating to teamwork and taskwork, and their occurrence during team formation, influence performance at later development times. It also documents the importance of coordinating mechanisms, like cognitive artifacts in GitHub, and how they are related to team effectiveness. At the practical level, understanding the features we extracted, and how they relate to team performance, could help repository maintainers and developers lead their efforts more efficiently. When creating new projects, developers who

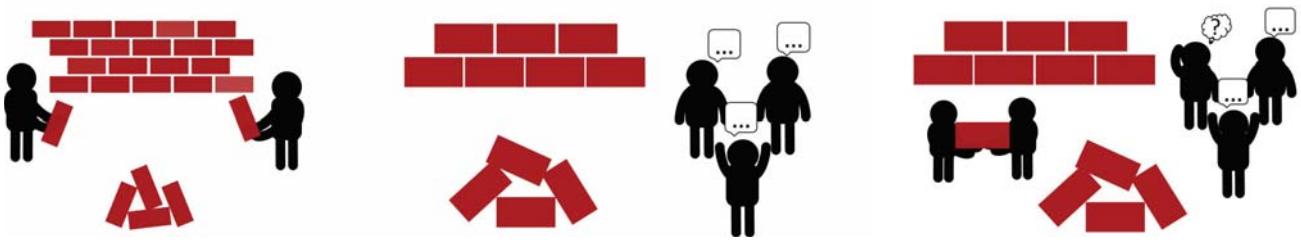

Fig. 6. From left to right: Toilers, Communicators, and Collaborators

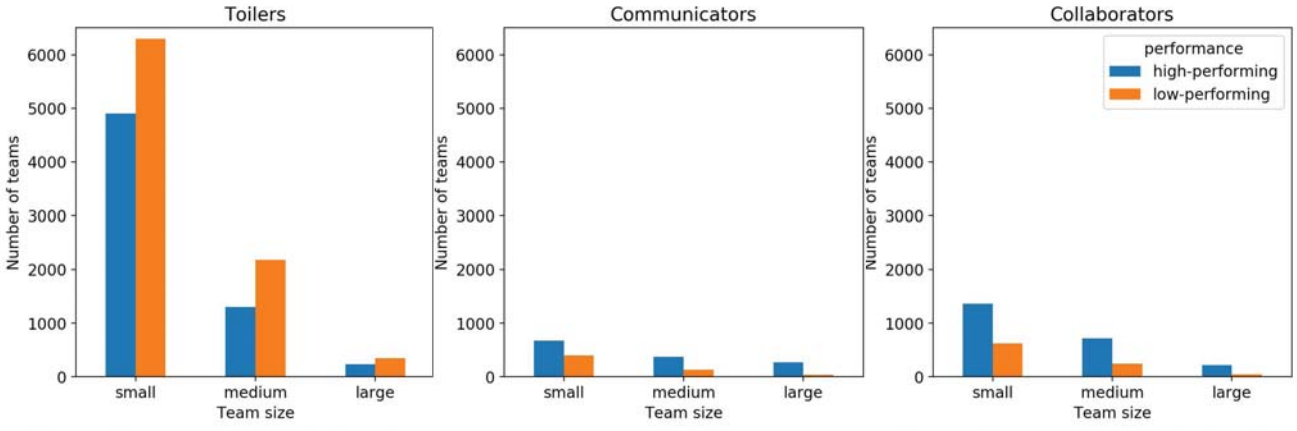

Fig. 7. The performance of different work styles and team size groups.

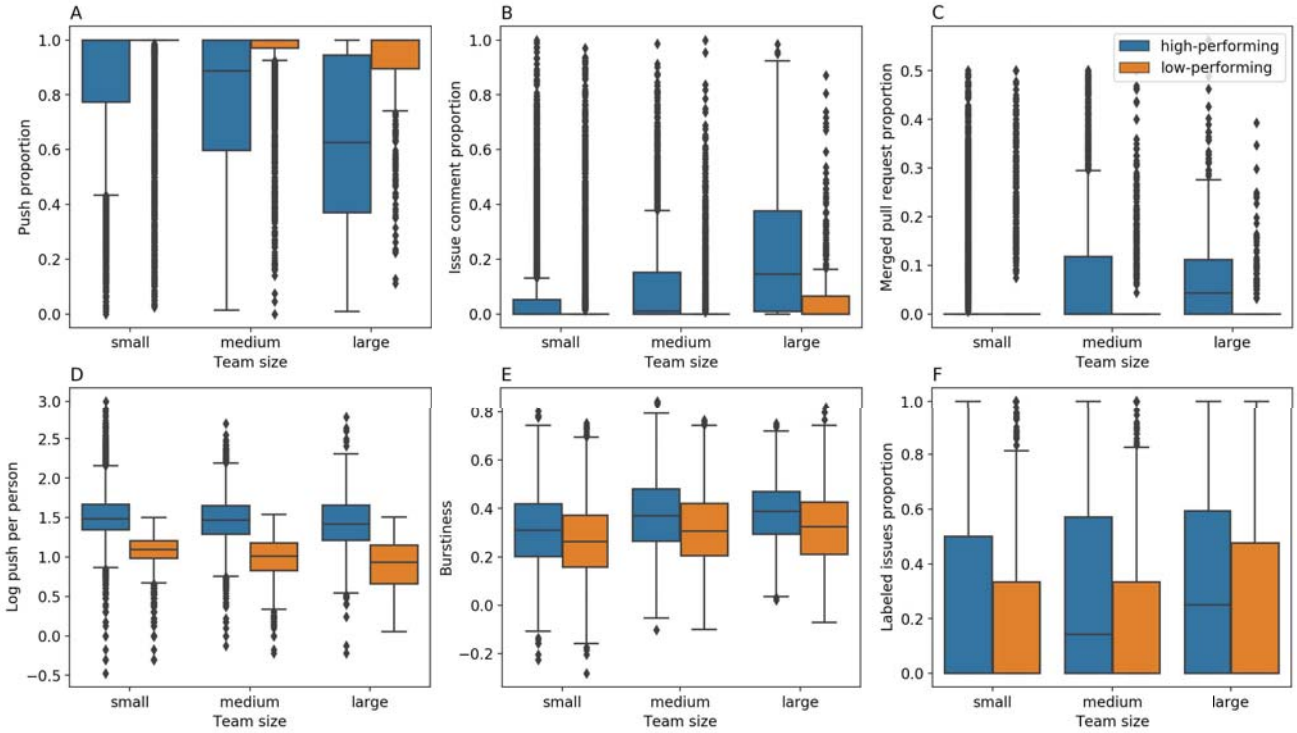

Fig. 8. Team formation phase features by team size and performance group.

would like to attract productive contributors could consider how their use of the platform for coordination purposes will influence the type and levels of work that will emerge over time. When identifying new projects to contribute to, developers may consider the amount of explicit coordination they observe in the repository to gauge the work style and productivity levels of the developers in the repository.

VII. Acknowledgements

This work was supported by grants FA8650-18-C-7823 and W911NF-20-1-0008 from the Defense Advanced Research Projects Agency (DARPA). The views and opinions contained in this article are the authors and should not be construed as official or as reflecting the views of the University of Central Florida, DARPA, or the U.S. Department of Defense.

References

[1] H. Borges, A. Hora, and M. T. Valente, "Predicting the popularity of GitHub repositories," in *Proceedings of the International Conference on Predictive Models and Data Analytics in Software Engineering*, ser. PROMISE 2016. New York, NY, USA: ACM, September 2016, pp. 9:1–9:10.

[2] A. Al-Rubaye and G. Sukthankar, "A popularity-based model of the diffusion of innovation on github," in *Conference of the Computational Social Science Society of the Americas*. Springer, 2018, pp. 165–178.

[3] R. Nielek, O. Jarczyk, K. Pawlak, L. Bukowski, R. Bartusiak, and A. Wierzbicki, "Choose a job you love: Predicting choices of GitHub developers," in *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, October 2016, pp. 200–207.

[4] I. S. Wiese, R. Takashi Kuroda, and G. Ansaldi Oliva, "Do historical metrics and developers communication aid to predict change couplings?" *IEEE Latin America Transactions*, vol. 13, no. 6, pp. 1979–1988, June 2015.

[5] S. W. Kozlowski, S. M. Hully, E. R. Nason, and E. M. Smith, "Developing adaptive teams: A theory of compilation and performance across levels and time," in *The Changing Nature of Performance: Implications for Staffing, Motivation, and Development*, ser. Frontiers of Industrial and Organizational Psychology, D. R. Ilgen and E. D. Pulakos, Eds. San Francisco, CA: Jossey-Bass Publishers, 1999, p. 452.

[6] 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 Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*. ACM Press, August 2015, pp. 817–828.

[7] S. M. Fiore, D. R. Carter, and R. Asencio, *Conflict, Trust, and Cohesion: Examining Affective and Attitudinal Factors in Science Teams*, 2015, ch. 10, pp. 271–301.

[8] S. M. Fiore and T. J. Wiltshire, "Technology as teammate: Examining the role of external cognition in support of team cognitive processes," *Frontiers in Psychology*, vol. 7, p. 1531, 2016.

[9] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on GitHub," *Information and Software Technology*, vol. 70, pp. 30–39, Feb 2016.

[10] M. Joblin, S. Apel, and W. Mauerer, "Evolutionary trends of developer coordination: a network approach," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2050–2094, August 2017.

[11] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in GitHub teams," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 3789–3798.

[12] S. Onoue, H. Hata, and K. Matsumoto, "A study of the characteristics of developers' activities in GitHub," in *Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC)*. Piscataway, NJ: IEEE Press, 2013, pp. 202–211.

[13] R. Rico, M. Sanchez-Manzanares, F. Gil, and C. Gibson, "Team implicit coordination processes: A team knowledge-based approach," *Academy of Management Review*, vol. 33, no. 1, pp. 163–184, January 2008.

[14] K. Blincoe and D. Damian, "Implicit coordination: A case study of the Rails OSS project," in *IFIP International Conference on Open Source Systems*. Springer, April 2015, pp. 35–44.

[15] D. M. Romero, D. Huttenlocher, and J. M. Kleinberg, "Coordination and efficiency in decentralized collaboration," in *Proceedings of the 9th International AAAI Conference on Web and Social Media*. AAAI Press, March 2015, pp. 367–376.

[16] F. P. Brooks, *The mythical man-month: essays on software engineering*. Addison-Wesley Pub. Co, 1975.

[17] O. Jarczyk, S. Jaroszewicz, A. Wierzbicki, K. Pawlak, and M. Jankowski-Lorek, "Surgical teams on GitHub: Modeling performance of GitHub project development processes," *Information and Software Technology*, vol. 100, pp. 32–46, August 2018.

[18] D. A. Norman, *Cognitive artifacts*. Cambridge University Press, 1991, pp. 17–38.

[19] E. Hutchins, "How a cockpit remembers its speeds," *Cognitive Science*, vol. 19, no. 3, pp. 265–288, 1995.

[20] N. McDonald and S. Goggins, "Performance and participation in open source software on GitHub," in *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '13. New York, NY, USA: ACM, April 2013, pp. 139–144.

[21] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*. ACM, Sep. 2015, pp. 805–816.

[22] C. Riedl and A. W. Woolley, "Teams vs. crowds: A field test of the relative contribution of incentives, member ability, and emergent collaboration to crowd-based problem solving performance," *Academy of Management Discoveries*, vol. 3, no. 4, pp. 382–403, 2017.

[23] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload–a case study of the Gnome ecosystem community," *Empirical Software Engineering*, vol. 19, no. 4, pp. 955–1008, Aug. 2014.

[24] S. Saadat, C. Gunaratne, N. Baral, G. Sukthankar, and I. Garibay, "Initializing agent-based models with clustering archetypes," in *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 2018, pp. 233–239.

[25] M.-A. Storey, A. Zagalsky, F. F. Filho, L. Singer, and D. M. German, "How social and communication channels shape and challenge a participatory culture in software development," *IEEE Transactions on Software Engineering*, vol. 43, no. 2, pp. 185–204, February 2017.

[26] S. M. Fiore, K. A. Smith-Jentsch, E. Salas, N. Warner, and M. Letsky, "Towards an understanding of macrocognition in teams: developing and defining complex collaborative processes and products," *Theoretical Issues in Ergonomics Science*, vol. 11, no. 4, pp. 250–271, July 2010.

[27] S. M. Fiore, M. A. Rosen, K. A. Smith-Jentsch, E. Salas, M. Letsky, and N. Warner, "Toward an understanding of macrocognition in teams: predicting processes in complex collaborative contexts," *Human Factors*, vol. 52, no. 2, pp. 203–224, Apr 2010.

[28] C. Lebeuf, M.-A. Storey, and A. Zagalsky, "Software bots," *IEEE Software*, vol. 35, no. 1, pp. 18–23, January 2018.

[29] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: An empirical study on count and network metrics," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, May 2017, pp. 164–174.