SAND2021-9378C

A Data-driven Approach to Rethinking Open Source Software Organizations as a Team of Teams

Elaine M. Raybourn[1⋆], Reed Milewicz[1], David M. Rogers[2], Benjamin H. Sims[3], Greg Watson[2], Elsa Gonsiorowski[4], and Jim Willenbring[1]

[1] Sandia National Laboratories, P.O. Box 5800 MS 0639
Albuquerque, New Mexico 87185 USA
{emraybo,rmilewi,jmwille}@sandia.gov
[2] Oak Ridge National Laboratory, 1 Bethel Valley Road
Oak Ridge, Tennessee 37380 USA
{rogersdm,watsongr}@ornl.gov
[3] Los Alamos National Laboratory, P.O. Box 1663
Los Alamos, New Mexico 87545 USA
bsims@lanl.gov
[4] Lawrence Livermore National Laboratory, 7000 East Ave.
Livermore, California 94550 USA
gonsie@llnl.gov

Abstract. This paper introduces a data-driven approach toward rethinking the characterization of a virtual organization as an aggregate, *team of teams*. We developed a repository mining toolkit, Reposcanner, to assist software teams in identifying opportunities for software process improvement and connected collaborations with other teams, thus characterizing efforts as team of teams. We use the example of continuous integration (CI) testing as an indicator of software development team productivity that signals potential best practices in which team of teams collaboration exists. Our preliminary findings indicate that the degree to which continuous integration testing is present can also signal the likelihood of related best practices, and a team of teams organizational structure.

Keywords: software productivity, repository mining, team of teams

1 Introduction

The Exascale Computing Project (ECP) is a virtual organization comprised of 6 co-design centers, 24 computational science applications, and 71 software technology products developed by teams from national laboratories, universities, and industry. Over 1000 researchers have contributed to an interconnected

⋆ Contact Author

web of software projects since its inception in 2017. It is thus a large collection of open-source scientific software projects, and in a sense, one example among many of the ongoing live experiments in coordinating codes, disciplines, and institutions at scale. We define a *virtual organization* as one that is composed of geographically dispersed members who mostly rely on electronic communication technology as a principle means of executing work. For these reasons, the ECP provides an ideal opportunity to apply tools and techniques of repository mining, backed by decades of academic scholarship, to deepen our understanding of what it takes to make scientific software teams successful. Drawing upon a well-supported foundation of software improvement theory and practice, the authors are developing a toolkit to automate the workflow of the Productivity and Sustainability Improvement Planning (PSIP) methodology. The PSIP toolkit includes a software repository mining framework which characterizes collaboration and software development testing practices (Reposcanner), interactive self-assessment (RateYourProject [RYP]), and progress tracking cards (PTC). Together, these tools are designed to assist development teams in achieving software process maturity.

In the present paper, we provide preliminary findings from ongoing efforts by the PSIP team to model software development practices found in aggregate within the ECP that may generalize to open source software organizations. Using Reposcanner as a tool for data-driven exploration, we examine continuous integration practices across teams as a lens from which to describe collaborative scientific software development at scale.

2 Data-driven Approach to Characterizing ECP as a Team of Teams

The ECP has drawn geographically dispersed researchers to work together as members of the virtual organization that is advancing the United States' first exascale software stack (see exascaleproject.org). While the ECP is organized to integrate and interoperate the work of many teams, an examination of solely an official organizational structure does not provide adequate insights into how software teams work together, coordinate their actions in reality, and whether they form a collection of independent teams or are truly interdependent. Understanding the extent to which the ECP functions as a collection of aggregate teams is the subject of the research articulated in the present paper. Our research seeks to identify opportunities for open source software process improvement at scale. In this research we use the public-facing GitHub repositories of the ECP as a use case to understand whether, and to what extent, interconnectedness among projects, code repositories, individuals, and teams impact computational science practices, risk mitigation, and the overall culture of sustainability/productivity across the virtual organization. We intend to apply this information to inform the development of tools aimed at automating the software process improvement method known as Productivity and Sustainability Improvement Planning (PSIP) [1].

The ECP has been described as a team of teams [2]. A *team of teams*, as described by McChrystal et al. [3] and Fussell and Goodyear [4], is a distributed, coordinated organization that works toward a common goal, and is made up of a number of smaller, close-knit, semi-independent teams. The challenge for organizations such as ECP, composed of aggregate teams, is to preserve some of the advantages of small teams (e.g. ease of communication, interpersonal trust, shared consciousness, and agile decision-making) and scale these advantages to work across the larger, coordinated organization as a team of teams [2]. It is difficult to do this in a traditional, hierarchical organization, so [3, 4] propose a more network-oriented organizational structure that enables the team of teams to more effectively coordinate and share information. Teams of teams interoperate through collaboration around an aligning narrative [4]. One coordination mechanism [2–4] suggested is to appoint liaisons between teams: key members of one team who embed with another team for a period of time in order to build trust and serve as a steward of cross-team communication. Additionally, McInnes et al. [5] and others [2] assert that the most successful ECP teams collaborate through software ecosystems—as developers, users, and testers.

In the subsequent section we discuss early stage data analyses which appear promising toward a description of team of teams collaboration at scale. Our analyses suggest ECP, a virtual organization, may have some key characteristics of a team of teams organization in which the associated software ecosystem of projects is linked by a network of shared contributors.

2.1 Methodology

To match the scale of the ECP and keep pace with its evolution, we leveraged solutions for automated, scalable data collection and analysis using software repository mining. Software repository mining is the subdiscipline of empirical software engineering concerned with collecting and interpreting artifacts from code development platforms to uncover insights and actionable information about software systems and projects. Historically speaking, most scientific software has been the product of individuals and small teams working in isolation to create bespoke scripts for their own research [1]. However, the cultural shift towards open, reproducible science and open-source, collaborative community software projects has now made it possible to rigorously study the character and evolution of scientific software.

We conducted a systematic, rapid review of the literature on data mining software development team repositories, which identified 742 relevant published articles as of December 2020. Through further review, we narrowed these down to 47 papers that were most directly relevant to our research questions, including three key references on research design [6–8]. These references point to GitHub as presenting a unique and insightful data source for all aspects of development, including internal and external team behaviors. However, each also cautioned that data obtained from these sources do not represent comprehensive activity logs, can be revised over time, and have inherent identifiability problems.

2.2 Research Questions

By studying ECP development teams and their practices at scale using repository mining and social network analysis, we were able to draw general conclusions about observable behaviors and characteristics expressed in the artifacts of code that could contribute to a snapshot of a team's software productivity practice. It is important to note that in the context of repository mining research, establishing and disambiguating software contributor identities is a significant challenge [6]. Accurate determination of developer identity is a necessary step towards connecting individuals and teams to measures of activity and impact. However, studies of large-scale open-source software projects often must contend with incomplete or inaccurate data that limits the ability of researchers to draw reliable conclusions.

Therefore, we decided to defer the employment of targeted interviews and/or surveys to the future, as we refine and evaluate our data analytics approach and findings. We were especially interested in understanding whether ECP teams that have public accessible GitHub repositories could be identified as leveraging each other's software. Therefore our initial investigations were guided by the following research questions (in order of exploration):

RQ1: How many contributors[5] participate in more than one ECP project?

RQ2: How many ECP projects have explicit code dependencies upon other ECP projects?

RQ3: How many ECP project teams have implemented some form of tests, code examples, and continuous integration–specifically, can we infer from their practice of testing code that they function as a team of teams?

RQ4: Does ECP have a social network structure that would indicate it functions as a team of teams – specifically, can we identify ECP members who function as liaisons[6] between teams?

2.3 Datasets

We bound our initial analyses to relevant, publicly available GitHub artifacts associated with ECP software teams, such as lists of project members, code repositories, and metadata.

Due to overlapping software technologies among projects, some repositories were difficult to attribute entirely to a single ECP project. Therefore, to validate the repository list, we manually curated the list of project repositories, and cross-validated the repositories with project assignments by verifying the presence of known project personnel in GitHub developer lists. This step allowed us to base the dataset of relevant repositories on ECP project "ground truth" and address risks to data integrity identified in [6–9]. Through this process, we compiled a list of 296 ECP software repositories. Of those, 259 were hosted by

[5] For analysis purposes, we define a *contributor* as a unique individual who committed a revision to 1 or more repositories (within a project or cross-project).

[6] For analysis purposes, we define a *liaison* as a unique individual who committed a revision to 2 or more cross-project repositories.

GitHub, 16 by Bitbucket, 18 by a GitLab server, and 3 by other types of version control. Based on this list, only 53 of 82 total ECP projects (as defined by the official work breakdown structure as of August 2020) were represented in our list of publicly hosted repositories. The remainder of the projects likely have private, or other source code repositories unknown to the PSIP team. Approximately half of the projects had four or more repositories on GitHub. Datasets of open source software developer code commits, issues, pull requests, and comments were generated using the Reposcanner toolkit (described below) and analyzed in Python using the Pandas and NetworkX libraries. Commit history was analyzed from 277 of the repositories above, while issues, pull requests, and comments were only analyzed from GitHub. While this is not by any means a complete representation of aggregate ECP teams, the dataset enabled us to test our hypotheses.

3 Reposcanner Data Mining Software Toolkit

We developed Reposcanner, a novel software repository mining toolkit, to aid in our research to characterize team of teams collaboration and software development testing practices. Written in Python, Reposcanner provides a highly modular, extensible framework for defining routines for mining data from software repositories and performing analyses on that data to yield insights about team behaviors. Reposcanner provides several attractive features not normally seen in repository mining research codes, such as seamless support for different version control platforms like GitHub, GitLab, and Bitbucket, smart parsing of URLs, intelligent credential management capabilities, and a comprehensive test suite. Reposcanner includes in-depth provenance logging, end-to-end analysis support from raw data to graphs and tables, and a powerful data management layer with rich metadata capabilities. The rich data that can be gleaned through repository mining with Reposcanner spans from the very granular, like quantitative metrics and qualitative instances of individual developer activity, to developer networks and patterns of collaboration, and then on to ecosystem-wide studies on software co-evolution across dependency networks. These advances enable reproducible research and provide traceability for artifacts used in team decision making.

Figure 1 presents the overall architecture of the Reposcanner toolkit. The tool is built around a request-response architecture model where user inputs are translated to a set of tasks which are consumed by data mining routines. The resulting data is held in a communal data store which can be leveraged by downstream analyses to derive human-readable reports that are meant to inform team decision making about their projects. It is our intention that Reposcanner be packaged as a toolkit to assist software teams in identifying opportunities for software process improvement, and in summarizing the state of their practice, as in step 1 of the PSIP methodology [1].

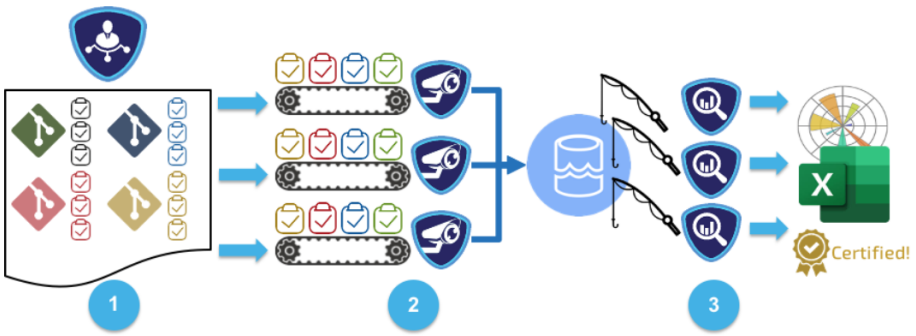

Fig. 1. An overview of the architecture of Reposcanner. Data collection operations are represented as task objects which are consumed by mining routines (1-2). The resulting data is held in a communal data store which can be leveraged by downstream analyses responsible for graphs, summaries, and other artifacts (3).

4 Discussion of Findings

To address RQ1, regarding contributors to more than one project, we applied Reposcanner to the GitHub database as of July 13, 2021. We found that 52 out of the 53 ECP projects in our dataset were connected to one another via at least one contributor in common. Although limited to ECP projects with public-facing source code repositories, our findings nevertheless indicate that a significant subset of the organization is connected via interactions across project teams.

Other conclusions drawn from querying this dataset include: (1) there were 4247 distinct contributors; (2) 617 individuals (14.5 percent of the population) crossed project lines, meaning that they contributed to repositories belonging to more than one ECP project; and (3) 71 individuals (1.7 percent) contributed across 4 or more projects (See Table 1.) In other words, we found a limited but substantial contingent of developers who work across domains and software layers, therefore strengthening our hypothesis that these teams may function as a team of teams. These findings are also relevant to RQ4, which is discussed further below.

To address RQ2, we used a curated subset of an official ECP project dependency database to construct a dependency graph of ECP projects. The dependency database, which was based on code dependencies entered by the project teams themselves, indicated that all categories of ECP software depend on programming models, development tools, and software ecosystem infrastructure packages. Thus, by querying a database generated by Reposcanner and curating additional lists, we were able to answer RQ1 and RQ2: namely that at the time of this research ECP had at least 240 individuals who contributed to one or more projects, and that all ECP projects self-reported dependencies. More

Table 1. Contributors to ECP projects. The left column indicates the number of projects a person contributed to, the center column indicates how many contributed to that number of projects, and the right column breaks this down into percentages.

# of ECP projects	# of contributors	% of contributors
≥ 7	7	0.2
6	6	0.1
5	22	0.5
4	36	0.8
3	122	2.9
2	424	10.0
1	3630	85.5

specifically, ECP application codebases are reported as dependent on software technology (ST) codes, tools, and libraries [9] as expected.

To address RQ3, we identified ECP teams that had adopted at least one of several best practices in software development such as continuous integration (which is of interest to related research [10]). Our findings relating to RQ3 on adoption of tests, code examples, CI, and documentation practices are illustrated by the plot in Figure 2. Here we computed the conditional probabilities of observing each practice within GitHub repositories belonging to a project team. For the purposes of our analysis, key project personnel are the minimum number of core developers on half the project's repositories, and core developers are defined as the set of developers needed to account for 80% of the lines of code added to a repository over the previous 12 months. Community contributors are defined as non-core developers who created an issue or comment over the previous 3 months. Meanwhile, the presence of tests, examples, tutorials, and CI were determined based on file names; lines of documentation per lines of code were based on file names and classifications made by the Linguist tool offered by GitHub [7]. While this approach leaves some ambiguity on how the files are used in practices, it does allow us to compare, in a coarse-grained way, the co-occurrence of different key practices and team structures (e.g. how many repositories with eight or more community contributors also use continuous integration services?).

The statistics in Figure 2 are aggregated at the project level. Even though a majority of the projects have some repository which implements these practices, there are still some teams that have not adopted CI. Further analysis of our data also suggests connections between software development practices (RQ3) and social network structure (RQ4). We found that teams implementing examples, tutorials, and CI also have high levels of community involvement, as defined by cross-project contributions. In fact, having more developers is the best indicator of whether tests exist. 87% of projects with tests have examples. Examples are the best predictor of whether a project has CI (at 71%). Finally, CI is the best indicator for whether a project has a large user community. Our analyses to

[7] https://github.com/github/linguist

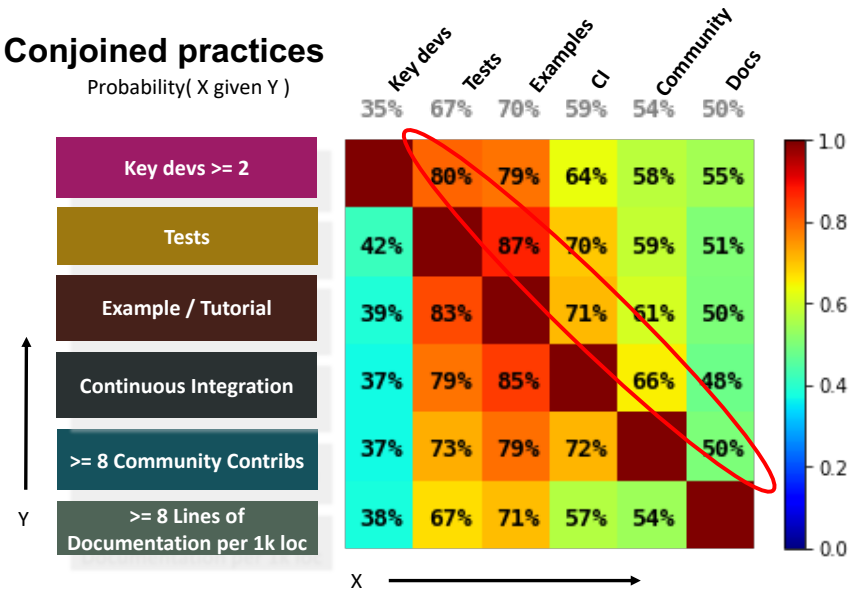

Fig. 2. Conditional probabilities of observing each practice within GitHub repositories belonging to a project team.

gain understanding about ECP project team implementation of CI yielded a few key observations. First, a highly interconnected ecosystem of project teams actively makes use of pull requests and issues. Second, larger teams, code tests, and examples are all related. Third, code examples attract collaborators, and are associated with implementation of automated build and test systems.

Finally, to address RQ4 we used an individual and their commits (revisions or changes to files) to GitHub project repositories as a rough indicator of whether they are associated with a given ECP team/project. Using a dataset collected with Reposcanner, we learned that the vast majority (86 percent) of contributors only committed to a single ECP project. However, there also appears to be a small but relevant number of contributors who made commits to multiple ECP projects. We noted that approximately 14% of ECP commit contributors have made at least one commit to more than one project, suggesting that these contributors might be considered liaisons between teams, and allowing us to infer that the ECP may be an example of how open source software organizations can function as a team of teams.

Another preliminary result from analyzing team repositories, relevant to RQ4, is that many GitHub issues in ECP repositories are associated with significant cross-team discussions. The average number of unique GitHub users involved in issues and discussions is 28.5 per project. This average is taken over user activity between January 2017 and February 2021. Of those 28.5 users, 9.1 are associated with an ECP project roster or a core developer on an ECP project,

but only 5.5 come from within the project itself. That makes for a cross-team discussion involving, on average, 3.6 people crossing over per project.

Finally, our findings in support of RQ4 are tentative at best. Nevertheless, we are encouraged that we have been able to identify common contributors among projects, suggesting that these contributors might serve as liaisons between teams, and that the ECP may indeed function as a team of teams.

5 Threats to Validity

While the majority of ECP repositories are hosted on GitHub, certain repositories are hosted outside of GitHub (Gitlab and Bitbucket), and at the time of writing, did not offer full support for those platforms. Likewise, a minority of repositories were private and therefore inaccessible. This means that we could not gather information for every ECP repository, resulting in an under-estimation of key personnel. Finally, our approach to identifying project practices is rudimentary, involving only detecting the presence of filenames starting with "test". However, the CI checks were more robust, since CI involves a set of files with a standard naming scheme. All of these issues are resolvable, however, and we hope to provide a more comprehensive treatment of the ECP in future work.

6 Conclusions and Future Work

By studying ECP development teams and their practices at scale, the authors were able to draw general conclusions about the observable behaviors and characteristics that contribute to the characterization of open source software teams as being part of a virtual team of teams organization.

Our preliminary data analyses with Reposcanner suggest that the ECP is not only large and diverse but also notably interconnected in that its projects are linked by a network of shared contributors, a smaller group of contributors potentially serving as liaisons among projects with common, or shared, motivations. These contributors represent individuals and teams working together across codes, disciplines, and institutions to produce software for the exascale ecosystem. We plan to build on this work by: (1) continuing to develop methods for verifying and validating Reposcanner data; (2) collecting additional data on number and frequency of code and text commits in order to identify different categories of developers and liaisons; and (3) integrating a team of teams network analysis with analysis of code development practices to create a comprehensive picture of team relationships and activities signaling productivity and a cultural propensity for incremental software process improvements.

Establishing a culture of software productivity that upgrades team practices is necessary for a project like the ECP, whose scientific challenges demand solutions that are frequently multi-scale, multi-platform, multi-physics, and multi-disciplinary. It also reflects the more general trend in scientific software towards open source community code development. In the future we hope to answer the following questions: How do ECP teams work together as a team of teams? What

kinds of contributions do they make to each other's projects? Are the tools and techniques they use sufficient for this purpose? How do the connections form and evolve over time? Are these connections indicative of a culture of productivity?

As comments and issues provide a rich dataset from which to investigate collaborations, our future research will continue to look more closely at these data. Timestamps also provide insight into the evolution of these activities over time. In the present research, source code itself was used to detect whether features like documentation, testing, and continuous integration were implemented. We implemented several Reposcanner routines that distilled data from these APIs into output files stamped with provenance metadata.

Finally, previous research has established that using data even from a very stable platform with a comprehensive API, like GitHub, poses considerable data collection and validation challenges [9]. A key part of our research is understanding these challenges and developing methods for ensuring our data collection is reliable, a process that is ongoing. Further, arriving at a deeper understanding of ECP culture and how to intervene to improve practices requires that we draw together data from heterogeneous quantitative and qualitative sources. We intend to augment our research with confirmatory qualitative project characterization in the future.

Acknowledgments. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Oak Ridge National Laboratory is a multiprogram research laboratory managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725. This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

1. Heroux M.A. et al. (2020) Lightweight Software Process Improvement Using Productivity and Sustainability Improvement Planning (PSIP). In: Juckeland G., Chandrasekaran S. (eds) Tools and Techniques for High Performance Computing. HUST 2019, SE-HER 2019, WIHPC 2019. Communications in Computer and Information Science, vol 1190. Springer, Cham. `https://doi.org/10.1007/978-3-030-44728-1\_6`.
2. Raybourn, E.M., Moulton, J.D., Hungerford, A. (2019) Scaling Productivity and Innovation on the Path to Exascale with a "Team of Teams" Approach. In: Nah FH., Siau K. (eds) HCI in Business, Government and Organizations. Information Systems and Analytics. HCII 2019. Lecture Notes in Computer Science, vol 11589. Springer, Cham. `https://doi.org/10.1007/978-3-030-22338-0\_33`.

3. McCrystal, S., Collins, T., Silverman, D., Fussell, C. (2015) Team of teams: New rules of engagement for a complex world. New York, NY: Penguin Random House LLC.
4. Fussell, C. and Goodyear, C.W. (2017) One mission: How leaders build a team of teams. New York, NY: Penguin Random House LLC.
5. McInnes, L.C., Heroux, M.A., Draeger, E.W. et al. (2021) How community software ecosystems can unlock the potential of exascale computing. Nat Comput Sci 1, 92–94. https://doi.org/10.1038/s43588-021-00033-y.
6. Amreen, Sadika, et al. (2020) "ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems." Empirical Software Engineering 25, 2. 1136-1167.
7. Serebrenik, A. and Mens, T. (2015) Challenges in Software Ecosystems Research. ECSAW '15: Proc. 2015 Eur. Conf. on Software Architecture Workshops September 2015, no. 40 https://doi.org/10.1145/2797433.2797475.
8. Ilo, N., Grabner, J., Artner, T., Bernhart, M., Grechenig, T. (2015) Combining software interrelationship data across heterogeneous software repositories. 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME) https://doi.org/10.1109/ICSM.2015.7332516.
9. Kalliamvakou, E., Gousios, G., Blincoe, K. Singer, L., German D. M., Damian, D. (2016) An in-depth study of the promises and perils of mining GitHub. Empirical Software Engineering 21, 2035–2071. https://doi.org/10.1007/s10664-015-9393-5.
10. Kothe, D., Diachin, L., and McInnes, L.C. (2020) Exascale update. ser. Winter ASCAC Meeting, Washington, DC, January 2020. Last retrieved on April 28, 2021 from https://science.osti.gov/-/media/ascr/ascac/pdf/meetings/202001/ECP_Update_0114202.