EveryBOTy Counts: Examining Human-Machine Teams in Open Source Software Development

Olivia B. Newton[1,2], Samaneh Saadat[3], Jihye Song[1,2],

Stephen M. Fiore[2,4], and Gita Sukthankar[3]

[1]College of Engineering and Computer Science

[2]Institute for Simulation and Training

[3]Department of Computer Science

[4]Department of Philosophy

University of Central Florida, Orlando, FL, USA

Keywords: Collaboration; Cognition; Software Development; Bots; Sociotechnical Systems

Corresponding Authors: Stephen M. Fiore (sfiore@ucf.edu)

and Olivia B. Newton (olivia.newton@ucf.edu)

3100 Technology Pkwy, Orlando, FL 32826

EveryBOTy Counts: Examining Human-Machine Teams in Open Source Software Development

Abstract

In this study we explore the future of work by examining differences in productivity when teams are composed of only humans or both humans and machine agents. Our objective was to characterize the similarities and differences between human and human-machine teams as they work to coordinate across their specialized roles. This form of research is increasingly important given that machine agents are becoming commonplace in sociotechnical systems and playing a more active role in collaborative work. One particular class of machine agents, bots, is being introduced to these systems to facilitate both taskwork and teamwork. We investigated the association between bots and productivity outcomes in open source software development through an analysis of hundreds of project teams. The presence of bots in teams was associated with higher levels of productivity and higher work centralization in addition to greater amounts of observed communication. The adoption of bots in software teams may have tradeoffs, in that doing so may increase productivity, but could also increase workload. We discuss the theoretical and practical implications of these findings for advancing human-machine teaming research.

EveryBOTy Counts: Examining Human-Machine Teams in Open Source Software Development

Collaboration continues to influence the future of work as problems become more complex and innovation becomes a primary driver for change, increasing the need for complementary forms of knowledge and skills (Fiore, 2008; Fiore et al., 2010; M. McNeese et al., 2020; N. J. McNeese et al., 2018). Concomitant with this, though, is a need to develop collaborative technologies that are able to scaffold work (Fiore & Wiltshire, 2016). To support modern work, a number of online collaborative support tools have emerged, some of which include artificial intelligence (AI) to create new forms of human-machine teaming. From a practical standpoint, it is unclear to what degree these intelligent technologies enable or hinder collaboration. From a theoretical standpoint, researchers must address how these technologies alter the nature of collaborative work. The field of cognitive science, grounded in interdisciplinarity, is well positioned to address these questions.

In this paper, we examine how open source software (OSS) development, a popular paradigm in modern software development, is altered when machine agents are members of the team. The creation and adoption of AI to support human work, referred to as bots, was made possible by advances in automation for continuous activities and repetitive tasks in this work domain. In this context, bots are autonomous software agents that complete critical work activities with humans in software development projects. These machine agents have communication and decision capabilities that enable them to interact with their human teammates to support their activities in addition to completing their own tasks (Golzadeh et al., 2021) in service of a shared goal, improved software.

We focus our research on open source projects hosted on GitHub, an online platform for collaborative software development, to explore how the presence of bots in teams is related to team productivity. Our goal is to set the stage for cognitive science research on the future of work via an interdisciplinary analysis that combines theory from the study of teams (e.g., Claggett & Karahanna, 2018; Rico et al., 2008), team cognition (Cooke et al., 2013), and macrocognition in teams (Fiore et al., 2010; Fiore & Wiltshire, 2016) with computational social science (e.g., Goggins, Mascaro, & Valetto, 2013; Lazer et al., 2009). We first briefly review sociotechnical systems theory and its relevance to understanding human-machine teaming. We then focus on open source development, a particular work domain where human-machine teaming is on the rise. We describe a study comparing human-only teams to human-machine teams and discuss how the inclusion of a machine agent as a member of a team alters collaborative processes and outcomes.

Sociotechnical Systems and Team Cognition

The future of work has been designated as one of the most significant challenges for research across disciplines. Academia, industry, and government are focusing on how to understand the changing nature of the sociotechnical systems, how to develop technologies for improving the nature of work, as well as how to design technologies to support human workers (e.g., *NSF's 10 Big Ideas: Future of Work at the Human-Technology Frontier*, 2020). The growing presence of intelligent machines altering the labor landscape in a number of organizational domains makes this even more challenging for researchers and practitioners (Erickson et al., 2018). Little is known about the effects of the inclusion of intelligent machines as team members. Although various forms of intelligent technologies have been studied for decades, much of the related research focuses on technology as part of a work system rather than a full-fledged member of a team. It is therefore imperative that researchers examine how socially intelligent machine teammates alter the behavior of humans engaged in collaborative work (Wiltshire et al., 2017).

The Macrocognition in Teams Model (MITM) was developed as a theoretical approach capable of addressing complex cognition when humans and machines interact (Fiore et al., 2010) and provides an appropriate foundation for this shift in research. It encompasses individual and collaborative processes as well as internal and external cognition and the processes associated with each (Figure 1). Relevant to the present context, the MITM has been applied to study how the integration of technology alters teams and augments their cognitive processes (Fiore et al., 2014). It can be used to examine how technology as a teammate influences coordinative activities across areas like individual and team knowledge building.

Researchers have also developed foundational work for the application of the MITM to the development of intelligent machines to support team cognition and improve collaborative processes for problem solving (Newton, Wiltshire, et al., 2018).

In initial work studying the communication-technology linkage, Fiore et al. (2014) drew from the MITM in a study of NASA mission control teams. They specifically focused on collaborative problem solving dealing with a failure of technology on the International Space Station. In that analysis, they found that macrocognitive functions, such as knowledge building activities, often extended cognition from individual to team, using technological scaffolds co-created by problem-solving teams. Fiore and Wiltshire (2016) elaborated on these and related findings to develop a theoretical framework for studying intelligent technology as a teammate by more clearly linking elements of the team cognition theory to technological scaffolds. In the MITM, humans and technology are seen as functioning as an integrated unit in which cognition is amplified.

[Figure 1 Placeholder]

Figure 1. The Macrocognition in Teams Model

More recent work has studied the deliberative processes that underlie major macrocognitive functions associated with collaborative problem solving in a lab setting. In a study of communication patterns and phases, Wiltshire et al. (2018) showed that the MITM was able to capture changes in collaboration processes while working to solve a problem. This ranged from variations in information requests and provisions to situation updates and affirmations of information. The proportion of differing macrocognitive processes shifted as teams solved problems (e.g., moving from knowledge constructions to deliberations around solution options).

Prior work on the effect of machine agents on coordination and team cognition has been limited by the availability and prevalence of intelligence technology in work domains. Advances in machine intelligence have, however, led to the introduction of software agents in online spaces (e.g., communication channels and social media). In the current context, we build on this to consider how machines alter human work when the technology itself exhibits a form of intelligent, agentic behavior in service of collaborative work. This introduces a set of research issues that must be understood from not only the computational standpoint, but also the psychological and organizational standpoint. Further, we add to newly developing areas of study where researchers are examining these forms of human-machine collaboration, including human-autonomy teaming, to understand how machine agents influence coordination and team performance.

In human-autonomy teaming, a human teammate collaborates with an autonomous agent to achieve a shared task goal and the autonomous agent has the ability to "take initiative and give orders to both human and automated counterparts" (Demir et al., 2019, p. 150). The literature describes an autonomous agent as a member of a team if it holds a unique role, but maintains interdependence with human activities, and completes tasks that would otherwise be assigned to a human, exhibiting independence and proactivity to alter process and performance (Fiore & Wiltshire, 2016; Larson & DeChurch, 2020; O'Neill et al., 2020). From that theoretical perspective, recent studies have examined differences between human-only teams and human-machine teams through the lens of interactive team cognition theory. This draws from complex systems theory, defining teams as a particular type of nonlinear dynamical system in which team cognition is an emergent property of team interactions.

Informed by the aforementioned theories, Demir and colleagues (2019) varied team composition in a controlled experiment, assigning participants to an all-human team, a human-machine team, or a human-confederate team where an experimenter played the role of an autonomous agent. When looking at teams interacting over time, and how perturbations altered performance, they found that coordination in human-machine teams was more stable than in all-human teams. The human-machine teams did not however achieve metastability in coordination—characterized by agility and responsiveness—as observed in the human-confederate teams. The rigidity of the human-machine teams resulted in lower levels of effectiveness in terms of both team performance and situation awareness. This finding supports recent theorizing that metastable coordination underpins optimal levels of team performance in demanding environments (Demir et al., 2018). Demir et al. (2019) theorize that their findings serve as evidence for an "inverted U-shaped relationship between team stability and team effectiveness" where optimal performance requires teams to "strike a balance between stability and instability" (p. 157). These findings add to prior research by Demir et al. (2018) and McNeese et al. (2018), who similarly showed, in different task contexts, that teams who were stable rather than metastable in their coordination demonstrated a lack of adaptability and less effectiveness in dynamic environments.

Researchers have also found that team structure and team building interventions influence social dynamics (e.g., communication patterns) in human-machine teaming, but only the latter factor contributes to changes in performance (Walliser et al., 2019). Such research finds that the use of team building activities, like mutual goal setting and role clarification, were associated with higher performance compared to informal cooperative gameplay between human and machine collaborators prior to the completion of the experimental task.

Across these studies, machine teammates seemingly contributed to the achievement of stability, but the development of machine agents with higher levels of flexibility and adaptability is paramount for the achievement of metastability. This gap in machine capabilities is evident, for example, in the observation that machine teammates failed to "anticipate information needs" at the same level of human-only teams (N. J. McNeese et al., 2018, p. 272). In addition to altering communication dynamics, low language skills on the part of the machine agents may have increased the workload of human teammates as they had to ensure their communications were unambiguous and without error lest they be incomprehensible to their machine teammate (e.g., human teammates could not use abbreviations). Indeed, Demir et al. (2018, 2019) note that the machine agents used in their experiments had limited language capabilities, imposing constraints on communication used by the humans on the team.

In sum, these studies provide insights into team processes and performance when intelligent machines are both a component of, and contributing members in, collaborations. This body of work, however, has relied on experiments using artificial tasks in addition to samples made up of primarily naive/novice participants (e.g., university students). Few studies have examined collaboration and coordination in human-machine teams in real-world work contexts. We next discuss how we addressed this gap and describe the particular context in which we study human-machine teaming.

Open Source Software Development

In the past two decades, the OSS development paradigm has become increasingly common in software development. This trend emerged, to some extent, as a result of changes in organizational practices (e.g., adoption of agile and distributed development approaches; Abrahamsson, Salo, Ronkainen, & Warsta, 2002; Dingsøyr, Nerur, Balijepally, & Moe, 2012) but was also driven by the popularity of software freedom among developers and self-described hackers (Coleman, 2013). Furthermore, online platforms for "social coding" have gained popularity among both organizations and individuals, promoting the temporal and geographic distribution of collaborative work in open source projects (Yu et al., 2014). For example, GitHub, a widely-adopted social coding platform, is used to host a variety of software projects, including those associated with major companies and those without direct or indirect corporate support (*Octoverse 2020*, 2020). Its community is made up of salaried and independent developers (Reyes López, 2017; Yu et al., 2014), resulting in collaborations that blend crowdsourcing with traditional teams (McDonald & Goggins, 2013).

In social coding platforms, developers may integrate commercially developed apps or create their own forms of intelligent automation to streamline and distribute elements of their work (Lebeuf, 2018). These technologies are commonly referred to as bots and their inclusion creates human-machine hybrids in which work is distributed across people and intelligent technology. We contend that the combination of the distributed nature of work in online platforms, the varied complexity of open source projects, and the inclusion of bots as members of the team, provides an ideal real-world context to study the future of work and the increasingly prevalent sociotechnical system resulting from humans collaborating with intelligent machines.

GitHub as a Sociotechnical System

Social coding platforms are a relatively new and significant means of supporting social interaction and technological advancement (Newton, Fiore, et al., 2018). In line with studies of collaboration in science and technology, collaborative software development relies on both social and task factors for goal completion. More specifically, this work requires an effective integration of both teamwork and taskwork (Fiore et al., 2015; Fiore & Wiltshire, 2016; Mathieu et al., 2000) and, as a collaborative platform for software development, GitHub aims to support these needs. In support of taskwork, for example, it purports to help users maintain awareness of their collaborators' activity, while in support of teamwork, it provides an infrastructure for communication and other forms of coordination. On the social side, the platform relies on features characteristic of social networking sites (e.g., following users, favoriting content, and threaded discussions) to connect users with shared interests and support resulting collaborations.

Despite the intent of platform designers, a number of challenges have been identified by its users and linked to direct or indirect effects on teamwork and problem solving in open source projects. For example, GitHub users have reported not using certain features (e.g., following users) because they are not particularly useful in terms of staying up to date with projects. Instead, these features negatively affect their productivity by introducing noise and information overload (Blincoe & Damian, 2015). Nonetheless, other features, such as the inclusion of bots to reduce developer workload, afford the possibility of increased productivity. This context, then, provides both theoretical and practical research directions. From a theoretical standpoint, there is a need to understand how this complex system is altered by these newer forms of collaboration. From a practical perspective, studying this context can contribute to the design of more intelligent machine teammates, and offer guidance on improving human-machine teaming requirements.

The diversity and complexity of the work observed in GitHub makes an analysis of team processes challenging. Despite this, we can examine work in a social coding platform from a task analytic perspective to identify the sociotechnical features supporting team functioning. In the extant literature, the structure of the platform is typically broken down into users, repositories, and features (Reyes López, 2017). The combination of these, summarized in Table 1, forms the foundation for GitHub as a sociotechnical system for OSS development.

Table 1. The structure of GitHub as it relates to taskwork. The third column provides examples of the macrocognitive processes and products that are associated with different aspects of the online platform.

GitHub Component	**Description**	**Link to MITM**
Users	Individual (personal) account	-
	Organizational account ● Can link people (individuals) to organizational account	-
Public Repositories (Repos)	Repos are a online project space used to host files, coordinate work, and engage in discussion around artifacts (e.g., source code) ● Multiple repos may form a larger project ecosystem	Externalized team knowledge Individual knowledge building (e.g., knowledge object development)
Interaction Modes: User-Repo	Create/delete a repo	Individual knowledge building Externalized team knowledge
	Fork: create a copy of an existing repo	Individual knowledge building (e.g., knowledge object development)
	Issue: report a problem, make a request, or initiate other discussion	Externalized team knowledge (e.g., artifact construction)
	Push: update a repo, changes recorded via commits	Individual knowledge building (e.g., individual information synthesis)
	Pull request (PR): suggest change(s) to a repo	Individual knowledge building (e.g., knowledge

		object development) Team knowledge building processes (e.g., team solution option generation)
	Comment: documentation and discussion linked to commits, issues, and PRs	Externalized team knowledge Team knowledge building processes (e.g., team evaluation and negotiation of alternatives)
	Watch: receive updates about activity in repo	Individual knowledge building (e.g., individual information gathering)
	Star: bookmark a repo	Individual knowledge building
Interaction Modes: User-User	Follow: receive updates about an individual user's activity	Individual knowledge building processes (e.g., individual knowledge gathering)
	Issue: communicate with specific user(s) via tagging/assigning mechanism, or broadcast message to project community	Externalized team knowledge (e.g., artifact construction) Team knowledge building processes (e.g., information exchange)
	Comment: communicate rationale, knowledge, and/or solutions related to project files to user(s), linked to commits, issues, and pull requests	Team knowledge building processes (e.g., team evaluation and negotiation of alternatives)

In GitHub and other similar platforms, repositories, or repos, are used to host project files (e.g., source code and documentation) and serve as a shared space for collaborative work. In these repositories, there is a significant amount of both collaborative activity and individual activity. Platform users can contribute to, track, and discuss changes to project files. Developers with 'write' privileges can make direct changes to files by pushing content from a local copy of the repository (e.g., on their personal machine) to the online repository. Any platform user can attempt to contribute to a repository by creating their own local copy, making changes, and submitting a pull request. Project integrators can then review the submission, discuss modifications through pull request review comments, and accept or reject the request to integrate suggested changes. Overall, the collective that makes up a repository are all contributing to a shared goal—that is, an improved software product.

In sum, platforms like GitHub represent an important development for the future of work. Decades of research in sociotechnical systems provide a foundation from which to build an understanding of these newer forms of collaboration. First, the work activity fits well with theories of teams. In the case of GitHub and similar online work, we refer to the collective as a team because their combined output is based on a significant amount of interdependencies with shared goals, and results in a unified product (Salas et al., 2008). Second, in line with sociotechnical systems theory, this work is distributed across people and machines, and increasingly, intelligent and autonomous technologies. Third, in line with cognitive science, this fits with the MITM by providing a space for integration of internalized and externalized knowledge in service of developing solutions to complex problems. We turn next to a discussion of cognition and coordination in these new forms of teams. We then discuss how the introduction of bots is potentially producing new interaction dynamics that alter collaboration and cognition.

In order to focus our analyses of GitHub differences, we considered specific elements of the MITM. Given the structure of GitHub, and how work is produced in open source projects, we consider the knowledge building processes and problem solving outcomes elements of MITM. We consider information sharing as it occurs through the use of issues and comments, and how these forms of externalized cognition, makes distributed cognition more concrete. In particular, these serve as important artifacts that support the communication and coordination within repositories. As such, they form the basis for how we examine differences in teams to study how the inclusion of bots may alter macrocognitive processes and overall team productivity.

Cognition and Coordination in Open Source Projects

In open source projects, a number of tools and communication channels are used to coordinate the work of developers. Some of these tools and channels are embedded in social coding platforms (e.g., issue tracking and user tagging) while others originate from these platforms (e.g., email notifications) or outside of them (e.g., in communication channels like Slack and social media like Reddit). Bots are a newer and increasingly common means through which developers coordinate and delegate their work (Hukal et al., 2019). Importantly, these technologies, including bots, vary in the degree to which they support taskwork (i.e., behaviors for meeting objectives) and teamwork (behaviors for working well with team members). We note here that this theoretical distinction is particularly important for cognitive science and the introduction of intelligent technologies. Understanding how bots can differentially contribute to taskwork and/or teamwork can inform the development of technologies that are more appropriately designed for different components of collaborative work.

Several threads of research on software development in GitHub are most relevant to the present work; namely, studies of individual and collective productivity in addition to studies of coordination. In these domains, researchers typically examine developers' activity profiles and/or their responses to surveys and interviews to better understand development processes, social dynamics, and project outcomes. The literature discusses differences between contributors, including their influence on each other (e.g., Blincoe, Sheoran, Goggins, Petakovic, & Damian, 2016), and their levels and length of participation (Onoue et al., 2013; Vasilescu, Posnett, et al., 2015). The evaluation of collective processes and outcomes, like productivity and efficiency, is important as it provides insights about the viability of the OSS approach for a given software product or service. Although performance measurement in GitHub is challenging, some measures have been proposed in the literature, including integrator productivity (e.g., proportion of merged pull requests), code quality (e.g., density of bug-fixing commits; Vasilescu, Yu, Wang, Devanbu, & Filkov, 2015), issue support quality (e.g., issue resolution times; Jarczyk, Jaroszewicz, Wierzbicki, Pawlak, & Jankowski-Lorek, 2018), and average developer productivity (e.g., work per person; Murić et al., 2019).

Researchers have also begun to explore different approaches for the study of coordination (Joblin et al., 2017) and its relationship with productivity (Choudhary et al., 2018). Coordination can be implicit or explicit, but it generally describes the means through which team members organize their activities towards a shared goal (Rico et al., 2008). 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. Various coordination processes in social coding support the knowledge building function of the MITM. This includes information sharing and discussion and deliberation of externalized cognition maintained in repos. For example, implicit coordination in social coding endeavors takes the form of information-gathering behaviors for the maintenance of awareness, such as reviewing code artifacts or following a developer's activities (Blincoe & Damian, 2015), whereas explicit coordination is observed in comments linked to issues and source code. In complement to each other, these processes support knowledge building through internalized cognition (coding expertise) and externalized cognition (e.g., artifacts). Last, research on coordination shows that the size of a team is likely to influence its members' performance due to associated social dynamics (e.g., social loafing and freeriding; Kidwell & Bennett, 1993) and evolving coordination requirements (Joblin et al., 2017).

In the context of software development, researchers have suggested that an investigation of the relationship between team size and productivity can aid in the identification of a "critical size" for a project's community in terms of its efficiency (Murić et al., 2019; von Krogh et al., 2003). This is an important feature of coordination given that researchers applying sociotechnical systems theory have long studied how process changes relative to team size. Community growth in open source projects has been positively associated with task completion (Chou & He, 2011) and is used by developers as an indicator of project success (McDonald & Goggins, 2013), but it can be negatively associated with issue resolution times (Jarczyk et al., 2018).

Coordination is also affected by team size. Studies of distributed software development suggest that, as in other industries, increasing team size results in increased communication costs and the need for task delegation (e.g., Joblin et al., 2017; Romero et al., 2015). But studies also show that coordination problems arising from large teams can be ameliorated. For example, the presence of a small team of developers, who focus on managing and coordinating work in large open source projects, is linked to improved issue resolution (Jarczyk et al., 2018). One way that open source teams have sought to improve their performance and manage coordination challenges is through the adoption of technologies based in AI: bots. We next discuss some of the research that has explored the implementation of bots in open source projects.

Bots in Software Development

The widespread adoption of bots in software development is attributable to technological advances that ease their integration in systems that support collaborative work (Lebeuf, 2018). Bots are expected to improve software development processes, resulting in increased productivity and better products (Lebeuf et al., 2018). Developers have recognized the utility of bots as facilitators of their collaborative work and have thus sought to harness their potential. These bots can differ in terms of their capabilities, as well as how they interact with the humans on a team. In the relevant literature, bots are conceptualized as an interface between user and services, enabling a distinct interaction style that provides value above and beyond traditional user interfaces.

Role-based classification of software bots reveals that the majority are taskwork-oriented, with capabilities for a particular set of tasks in the development process (see Table 2). Although bots vary in terms of their reasoning and agency, they are increasingly autonomous and equipped with social, perceptual, and cognitive capabilities associated with agent-hood. GitHub bots have been classified as having mid to high levels of agency, being able to take goal-directed action both with and without supervision (i.e., partial and complete autonomy, respectively; Lebeuf, 2018; Lebeuf et al., 2019). These capabilities enable bots to respond to their human counterparts' taskwork needs (e.g., running tests on their code and offering potential followup actions) in addition to assigning taskwork to them (e.g., directing them to sign an agreement or review code modifications and submitted issues). Some research has begun to explore how bots can move beyond taskwork to also improve teamwork by, for example, reducing conflict between developers (Lebeuf et al., 2017).

Table 2. The types and roles of bots in software development, adapted from Storey and Zagalsky (2016). In the fourth column, we identify the MITM processes and products that are supported by each bot type.

Bot Type	**Role**	**Work Orientation**	**Link to MITM**
Coding Support	Coding support bots help developers to improve their workflow efficiency while engaged in coding activities. This includes, for example, synchronizing tools to reduce workload, supporting awareness of changes to the code, and merging code changes from multiple sources.	Taskwork	Individual knowledge building
Code Testing	Test bots help developers offload repetitive tasks by evaluating code. These bots can help reduce the time developers devote to testing and may assess the quality of the code or user interfaces, keep track of potential issues, and offer suggestions for improvement.	Taskwork	Individual knowledge building
DevOps	DevOps (AKA ChatOps) bots are designed to reduce communication needs and improve the efficiency of operations associated with software development and release. These bots specifically help coordinate the activities of developers and other personnel across communication tools, and can help “bridge the technical-knowledge gap for stakeholders on the team” (p. 930).	Taskwork/Teamwork	Team knowledge building
User Support	User support bots help developers manage reports and requests from the software product’s user base. For example, these bots can directly communicate with users, provide answers to frequently asked questions, and record user feedback.	Taskwork	Team problem solving outcomes

Documenting	Documenting bots, as their name implies, help with code and software release documentation by "aggregating information from code commits and issue comments" (p. 930).	Taskwork	Externalized team knowledge

Some bots used in open source projects are provided for developers through GitHub Marketplace, the platform's store for development tools. GitHub Apps, which are official GitHub bots, are provided to automate developer tasks and potentially improve efficiency. Recent research on software bots in GitHub finds that, although they complete a variety of tasks, their adoption in a project does not yield significant increases in developer productivity (Wessel et al., 2018). Furthermore, some developers perceive bots as lacking sufficient intelligence for decision support roles and report dissatisfaction with bots' contributions to collaborative processes (e.g., because of perceived unfriendly tone; Wessel et al., 2018).

Researchers have also analyzed differences in software repositories before and after the adoption of Continuous Integration bots (Bernardo et al., 2018). In this study, it was observed that 51% of teams increased their pull request merge rate after bot adoption. Researchers found that the reason for the slower integration of external contributions in the remaining teams was a substantial increase in pull request submissions after bot adoption. These mixed results suggest time delays associated with the introduction of bots are not necessarily reflective of stagnant productivity levels. Instead, this reflects how bots can change development processes and potentially influence project growth and performance. While bots may help reduce developers' workload for some tasks, they may also inadvertently increase workload for developers managing the repository due to greater participation by both newcomers and established collaborators in open source projects.

Present Study

As we have described, the increasing role of AI in teams requires interdisciplinary research within the cognitive science community to improve their implementation and evaluation. From this, research can better account for the ways in which machine agents change performance and how such changes differently affect taskwork and teamwork (Fiore & Wiltshire, 2016). In the present study, we build on this body of work by taking a multifaceted approach to the study of productivity to explore differences between human-only and human-bot teams. Towards this end, we conducted a study of large-scale, distributed collaborative work in open source projects by analyzing data extracted from GitHub. As this is a relatively new area of research in the future of work, our hypotheses are exploratory in nature. Our hypotheses were guided by an overarching research question: Does the introduction of bots into the sociotechnical ecosystem alter the nature of work? We set out to compare two fundamental team parameters. First, we separated out team types; that is, whether or not a bot is present. Second, given the aforementioned issues of size and coordination, we considered the size of a team as a variable for analyses. With these distinctions, we developed the following hypotheses to determine if there are differences in productivity between human-only teams and human-bot-teams.

- *Hypothesis 1*. Bots alter the **productivity levels** observed in a team.
- *Hypothesis 2*. Bots alter the **degree to which work is centralized** in a team.
- *Hypothesis 3*. Bots alter the **efficiency of productivity** observed in a team.

Methods

To test our hypotheses, we analyzed data extracted from an online platform, GitHub, used by developers to work on open source projects. Information about both projects and individuals can be gleaned from these data to study collaborative work in the wild as opposed to in the controlled laboratory environment in which much prior research on human-machine interaction and teaming is typically set. In the following paragraphs, we describe in detail the approaches employed to collect, process, and analyze the data.

Data Source and Preprocessing

The data set analyzed for the purposes of this study represents a subset of a larger social media platform data set curated and provided to us by a data provider as part of a larger grant program. In accordance with the requirements of the organization funding the research, the data provider, an information technology corporation, anonymized the data set prior to sharing it with researchers to protect the privacy of GitHub users. The data set was collected from the GitHub API[1] in 2017. In past research, project age has been established as a significant predictor of activity levels and number of contributors (Fronchetti et al., 2019). To control for the effect of project age, we selected GitHub repositories of the same age in our larger sample, specifically only including data from repositories created in January 2016. This resulted in the selection of approximately 900,000 repos. We additionally only included repos with a specified programming language to ensure our analyses were applied to software projects as GitHub repos are sometimes used for purposes other than software development (e.g., content curation and data storage; Kalliamvakou et al., 2014), reducing the original selection to approximately 500,000 repositories.

Work Events. We refer to four specific GitHub event types as work events: pushes (internal file modifications), issue comments and pull request review comments (team communications), and accepted pull requests (external file modifications). Prior research has classified pushes as work events (e.g.,

[1] https://docs.github.com/en/rest

(Murić et al., 2019); we additionally included external file modifications because our study sample included projects that relied on this type of contribution in addition to pushes.

Project Team. We considered a project contributor to be a team member if they submitted: at least one push; at least ten issue comments or ten pull request review comments; or at least five merged (i.e., accepted) pull requests to the repository. Push events were used as criteria for team membership because they are made by individuals with special permissions, or access, to the repo in question. We used the stated thresholds for comments and merge requests because they represent relatively high levels of activity associated with higher levels of engagement with a project. The individuals who met these thresholds thus made substantial contributions and were classified as team members. Similar thresholds have been applied in prior research on collaborative work in GitHub (Murić et al., 2019). Teams that generated fewer than 20 work events in the platform in the first 6 months were excluded from the data set. These criteria were used to ensure that the projects were maintained by an active group of developers (≥ 2) and not abandoned soon after creation as has been observed in GitHub (e.g., Kalliamvakou et al., 2014). We identified 20,119 GitHub repositories that met these criteria. Event data for the first 13 months following repository creation was extracted for further processing and analysis.

Team Type. As part of the data sharing agreement, the data provider labeled a user as a bot if: the account type was identified by GitHub as a bot; the username ends with '-bot'; and/or the account generated repeated identical comments. Similar approaches have been described in the literature (e.g., Golzadeh et al., 2021). This information was used to classify the teams in the data set as either a human-only team or human-bot team. Of the over 20,000 teams in the data set, only 304 (2%) teams had at least one bot.

Smart Sampling. In addition to the issue of significant group imbalance between team types, there is a possibility that the members of human-bot teams had higher levels of expertise than the members of human-only teams in our data (i.e., only more experienced developers choose to adopt bots). To address this, we adapted a smart sampling approach developed by Saadat and Sukthankar (2020) and implemented in Python using metrics in the Scikit-learn library (Pedregosa et al., 2011) to control for variations in developer expertise. This algorithm was used to select the subset of human-only teams that were most similar to the human-bot teams in terms of expertise.

First, we identified GitHub features that serve as proxies for expertise in the platform and extracted a vector for every team member. While expertise research in other domains has identified specific criteria to differentiate experience levels, what counts as expertise in open source development is less well defined (Baltes & Diehl, 2018). As such, we determined a set of user characteristics that could differentiate more from less expert teams. For each team member, we created a vector consisting of: 1) their number of followers, 2) the number of users they follow, 3) the number of public repositories they own, and 4) their gh-impact. A user's gh-impact is based on the popularity of the repositories they own and thus quantifies influence on GitHub (Miller, 2016). A high gh-impact score indicates that the user owns many projects that exhibit high levels of activity.

Then, we generated a team-level expertise vector by summing each variable in the expertise vector of team members to represent their collective expertise. For example, in a team with two human members, let $u = (u_1, u_2, u_3, u_4)$ and $v = (v_1, v_2, v_3, v_4)$ be their individual expertise vectors. The sum of u and v is their collective expertise vector, $u + v = (u_1 + v_1, u_2 + v_2, u_3 + v_3, u_4 + v_4)$. To mitigate the dominance of variables with large values, we normalized the expertise vectors to range between zero and one prior to running the similarity calculation. For example, prior to normalization, the number of followers varied between 0 and 1,644 and the number of repos owned varied between 8 and 88,791.

Finally, team expertise vectors were matched between team type groups. For each human-bot team, the smart sampling algorithm selected the most similar human-only team based on minimum Euclidean distance. This process was repeated for each human-bot team and resulted in the selection of 304 human-only teams with expertise levels corresponding to the 304 human-bot teams in the data set. The final data set consisted of work events generated by these 608 teams (i.e., our study sample; Table 3).

Team Size. We classified teams as small if they had two or three human contributors, medium if they had between four and six human contributors, and large if they had more than six human contributors. Other researchers (e.g., Vasilescu et al., 2015) have used higher thresholds to categorize teams. However, we chose these thresholds in order to align with the traditional definition of team size as found in the human factors and organizational sciences literature (e.g., Weiss & Hoegl, 2016). Furthermore, by separating along these three size categories, we can better characterize how the presence of bots alters work activity in teams of traditional size as compared to those where their larger size may increase coordination demands.

Table 3. The number of teams in our sample by team size and team type. Team size range is provided in brackets. Although the small team size category was the largest (281 teams), the majority of teams in the sample had at least four human members (327 teams).

Team Size [min, max]	**Team Type**	**Number of Teams**
Small [2, 3]	Human-only	153
	Human-bot	128
Medium [4, 6]	Human-only	96
	Human-bot	84
Large [7, 246]	Human-only	55
	Human-bot	92

Team Productivity Variables

Our analysis of productivity focused on three complementary measures derived to help us characterize the amount and efficiency of work completed in a repository as well as the distribution of work across members of an open source project. This multifaceted approach helped us investigate and interpret differences in productivity in relation to the presence or absence of bots on a team, and thus test each of our aforementioned hypotheses.

Work Per Person. First, to evaluate the productivity levels of teams **(H1)**, we aggregated team members' GitHub activity during the first 13 months following the repositories' creation. Raw event counts and median values are provided in Table 4. This time period was selected to ensure that we collected a sufficient amount of data for our analyses. From these event counts we calculated the average amount of work per team member—number of pushes, issue comments, pull request review comments, and accepted pull requests—within a repository during this period. We used this measure because teams varied in size and our interest was not necessarily overall productivity, but the proportionate amount of activity per team member. The primary question of interest is whether work per person is higher or lower in traditional versus human-bot teams. Our approach can be distinguished from Murić et al.'s (2019) productivity measure in that our definition of team and work extends beyond code contributors (i.e., those that submit pushes and pull requests) to include other teamwork aspects of OSS development, including the coordination of work and knowledge building observed in comments, a team process foundational in the macrocognition in teams model (Fiore et al., 2010).

Table 4. Total counts and median values for work events generated by humans in each team type. Pull request is abbreviated to PR and pull request review comment is abbreviated to PRRC. Median values have been rounded to the nearest whole number. Only 60 (52 human-bot) of the 608 teams in the sample had PRs generated by humans and 346 (237 human-bot) of the 608 teams had PRRCs generated by humans.

Team Type	**Total Push**	**Median Push**	**Total Issue Comms**	**Median Issue Comms**	**Total PRRC**	**Median PRRC**	**Total Merged PR**	**Median Merged PR**
Human-only	45,348	64	28,744	4	9,194	0	217	0

Human-bot	80,985	130	85,983	46	39,820	16	1,184	0

Work Centralization. Second, to understand how work activity varies within a team **(H2)**, we considered the distribution of work across team members. This allowed us to examine an important issue with regard to variations in productivity and team size—that is, the high amount of work completed by a small number of open source project team members. In the management sciences and economics literature, the unequal distribution of a given variable is explained by the Pareto principle, or the 80-20 rule, which states that, in this context, 80% of the work in a team is completed by 20% of team members (Newman, 2005).

The inequality of work distribution can be measured using the Gini coefficient, a calculation initially developed to study income disparity (Dorfman, 1979), and has more recently been adapted to study disparities in other domains (e.g., in group diversity; Solanas, Selvam, Navarro, & Leiva, 2012), including work in OSS development (Jarczyk et al., 2018). The larger the Gini coefficient, the higher the centralization among a small number of people in a population, and thus, the more unequal the work distribution. With this, the Gini coefficient can help us determine if the distribution of work done per team member differs between human-bot teams and human-only teams. We calculated a Gini coefficient for each team using the inequality analysis method[2] provided in the Explore package of the PySAL Python library (Rey & Anselin, 2010). This function was applied to human-generated work events, which were positive, nonzero data.

Work Efficiency. Third, to characterize differences in work efficiency **(H3)**, we examined the rate at which teams resolve open issues in a project. Within GitHub's issue handling infrastructure, users can report a bug or provide a feature request, among other things, by opening an issue. The issue is closed when the problem or request is resolved or otherwise addressed. Issue closure rates thus reflect the speed

[2] https://github.com/pysal/inequality/blob/master/inequality/gini.py

with which teams resolve problems and have been proposed as a means to quantify development process performance of open source project teams (Jarczyk et al., 2018). In linking this measure with the MITM, we propose that the time to resolve, or close, an issue is a team problem solving outcome and specifically reflects efficiency in planning process and plan execution.

We performed survival analysis to calculate how quickly an issue is addressed by the teams in our sample. Survival analysis is primarily used to model the duration of time until an event happens (Klein & Moeschberger, 2003), and has been adapted to study many phenomena with temporal characteristics, including how long issues remain unresolved in software development projects (Jarczyk et al., 2018). We used a non-parametric statistic, the Kaplan-Meier estimator, to estimate the survival curve (Kaplan & Meier, 1958). The work efficiency variable was computed using lifelines, a Python implementation of survival analysis (Davidson-Pilon, 2014).

The repositories in our sample were active at the time of data collection. As a result, it was likely that there were open issues at the time of data collection that may have been closed after data collection, resulting in incomplete information about these issues. This means without an endpoint for such issues, it is unknown if and when they were resolved. Survival analysis is designed to utilize data containing incomplete information to make inferences and is thus well suited to our needs. For this analysis, we included only repositories that had at least 5 issues, resulting in the selection of 303 teams (191 human-bot teams). This threshold was applied to ensure meaningful results (i.e., based on teams who made use of the issue handling infrastructure) and was based on prior research on issue survival in GitHub (Jarczyk et al., 2018).

Statistical Tests and Transformations

Control Check: Smart Sampling. The variables selected as expertise proxies did not follow a normal distribution. As a control check, we used the Mann-Whitney U test, a nonparametric alternative to the independent sample *t*-test that does not assume normality, to statistically evaluate expertise differences between team types.

Transformations of Productivity Variables. The distribution of work per person was right-skewed. Therefore, for the purposes of visualization and statistical analysis, we applied a log transformation to the variable. The distribution of the work efficiency variable, median survival days, was also skewed and, because some of the teams in our sample had a median survival day of zero (i.e., they frequently resolved issues in less than a day), we applied a square root transformation to this variable. Transformations failed to sufficiently approximate a normal distribution leading us to apply nonparametric statistical tests in our analysis of work efficiency.

Productivity Variables Analysis. To statistically analyze differences in work per person and work centralization between team types, we used a series of Welch's unequal variances *t*-tests, a modification of the two sample *t*-test that is appropriate for cases in which there are unequal groups and there is unequal variance between those groups. The Holm method was applied to *p*-values to control the family-wise error rate (Holm, 1979) and we report Cohen's *d* effect sizes and corresponding confidence intervals (CI) for each analysis. We analyzed differences in work efficiency between team types with a series of Mann-Whitney *U* tests; *p*-values were adjusted using the Holm method and effect size for each analysis was calculated by dividing the *z*-score by the square root of the sample size.

Results

Data Characterization

We first provide an overview of the data to ground the presentation of our analyses. All visualizations and statistical tests were conducted using statistical computation software R (R Core Team, 2019) while data processing, the smart sampling algorithm, and survival analysis described earlier were implemented in Python (Python Software Foundation, 2019)[3]. Referring back to Table 4, work event counts and median values by team types reveal that humans in the human-bot teams generally engaged in

[3] Aggregated data and code files are available at https://github.com/small0live/bots-research.

more discussion around code artifacts (issue comments and pull request review comments) and more frequently submitted and accepted changes to source code (merged pull requests).

Team Sizes. We observed that larger teams had higher proportions of merged pull requests (i.e., accepted more external contributions). Summary statistics for team size are provided in Table 5 and counts for teams in each team size level in Table 6. Over half (54%) of the teams in the sample had at least four team members and most (84%) had at least three members. The human-bot teams were more evenly distributed across team size levels. In the large team size class, the majority of teams had 49 or fewer team members. This means the data were skewed with respect to team size, which limits our analysis of interaction effects between team types and team sizes.

Table 5. Summary statistics for team size (i.e., number of team members).

	Median	**Min.**	**25th Percentile**	**50th Percentile**	**75th Percentile**	**Max.**
Team Size	4	2	3	4	7	246

Table 6. The distribution of teams by number of members and the number and percent of human-bot teams in each team size level.

Team Size	**Number of Teams Overall**	**Number of Human-Bot Teams (%)**
Small [2, 3]	281	128 (45.6%)
Medium [4, 6]	180	84 (46.7%)
Large [7, 246]	147	92 (62.6%)

Bots in Teams. Most of the human-bot teams in our sample had only a single identified bot and just over twenty teams had two bots, but a few large teams made use of four or more bots (Table 7). The distribution of work events completed by bots is presented in Figure 2: bots produced a higher proportion of issue comments compared to the other events. This indicates that the bots were coordinating and prompting developer activity. The distribution of bot events suggests that the human-bot teams in our sample likely comprised coding support, code testing, and/or DevOps bots.

Table 7. The number of bots in human-only teams broken down by team size.

Number of Bots	Small Teams	Medium-Sized Teams	Large Teams	Total
1	122	78	80	280
2	6	6	9	21
4	-	-	1	1
8	-	-	1	1
12	-	-	1	1

[Figure 2 Placeholder]

Figure 2. Work events generated by bots grouped by team size. Relative to other event types, bot work is characterized by a high proportion of issue comments and pushes. Pull request is abbreviated to PR in labels on x-axis.

Expertise Control Check. Our smart sampling procedure was intended to serve as a control for expertise differences between teams that use bots and those that do not use bots. A series of Mann-Whitney *U* tests were run to analyze the difference in expertise values between team types and confirmed that they were not statistically significant (Table 8).

Table 8. Mann-Whitney *U* test results for expertise proxies by team type. As is standard for nonparametric approaches, median values are reported for each team type.

	$\textit{Median}_{\text{Human-}}$	$\textit{Median}_{\text{Human-Bot}}$	U	p

	Only			
# Followers	33.5	34.5	45735	.827
# Following	82	84.5	45500	.744
# Repos Owned	173.5	186.5	45552	.762
gh-impact	5	5	46056	.944

Productivity Differences

Work per Person. Because teams varied in size within each team size class, we calculated the amount of work activity generated per person on a team to examine differences in productivity levels. Figure 3 presents the distribution of work per person by team size and type. To reiterate, the analyses are only applied to human-work event data. Across team sizes, human members on the human-bot teams tended to have higher levels of productivity compared to members of human-only teams. Additionally, there is no clear trend—increase or decrease—in productivity as team size changes. Small human-bot teams (M = 3.86, SD = 1.07) compared to small human-only teams (M = 3.30, SD = 0.87) had higher levels of productivity t(243) = 4.81, p < .001; d = 0.57 (lower CI: 0.34; upper CI: 0.82). Medium-sized human-bot teams (M = 4.06, SD = 0.96) compared to medium-sized human-only teams (M = 3.14, SD = 0.97) had higher levels of productivity t(175) = 6.34, p < .001; d = 0.95 (lower CI: 0.64; upper CI: 1.26). Large human-bot teams (M = 4.02, SD = 1.11) compared to large human-only teams (M = 3.51, SD = 1.18) had higher levels of productivity t(108) = 2.59, p < .001; d = 0.45 (lower CI: 0.11; upper CI: 0.79). These results suggest that bots alter productivity levels and provide support for Hypothesis 1.

[Figure 3 Placeholder]

Figure 3. Notched boxplots of productivity in GitHub teams. Data are grouped by team type and size. Notch displays approximately 95% confidence interval around the median which is based on the median +/- 1.58 x IQR/sqrt(n) (Chambers et al., 1983; R Core Team, 2019). There were significant differences between team types across team size classes.

Work Centralization. We were also interested in examining the degree to which work is centralized or distributed across human team members. For this analysis, we used a common measure of inequality, the Gini coefficient. In this context, the Gini coefficient, which ranges from 0 to 1, is 0 if all team members perform an equal amount of work and increases as work distribution among team members becomes more skewed. Thus, a Gini coefficient approaching 1 suggests that work is unevenly distributed and centralized to a small subset of team members. The distribution of Gini coefficients is plotted in Figure 4 and additional information about these distributions is provided in Table 9. These boxplots show that human-bot teams tended to have higher levels of work centralization compared to human-only teams. In other words, the presence of bots is associated with greater disparity in how work is distributed across human team members.

We found higher work centralization, *t*(267) = 3.21, *p* = .004, *d* = 0.38 (lower CI: 0.14; upper CI: 0.61), in small human-bot teams (*M* = 0.38, *SD* = 0.16) compared to small human-only teams (*M* = 0.32, *SD* = 0.16). Similarly, medium-sized human-bot teams (*M* = 0.50, *SD* = 0.13) compared to medium-sized human-only teams (*M* = 0.43, *SD* = 0.16) had higher work centralization *t*(178) = 2.93, *p* < .008, *d* = 0.44 (lower CI: 0.14; upper CI: 0.73). For large teams, there was no significant effect for team type, *t*(97) = 0.77, *p* = .45, *d* = 0.14 (lower CI: -0.19; upper CI: 0.48), with human-bot teams (*M* = 0.58, *SD* = 0.13) having, on average, the same amount of work centralization compared to human-only teams (*M* = 0.56, *SD* = 0.16). These results suggest bots alter the degree to which work is centralized in small and medium-sized teams, providing partial support for Hypothesis 2.

[Figure 4 Placeholder]

Figure 4. Notched boxplots of work centralization in GitHub teams as indicated by the Gini coefficient, where higher values represent higher levels of centralization. Data are grouped by team type and size. There were significant differences between team types in small and medium teams. There was no significant difference between team types in large teams.

Table 9. Sampling distribution of the Gini coefficient. Information is provided for each team size category in addition to all 608 teams in the sample and all 20,119 teams that met our selection criteria prior to the application of the smart sampling algorithm.

Team Size	Median Gini Coefficient	Standard Error	95% CI
Small	0.37	0.16	0.02, 0.62
Medium	0.48	0.15	0.16, 0.72
Large	0.58	0.14	0.26, 0.80
Study Sample	0.45	0.18	0.06, 0.74
~20k Teams	0.32	0.17	0.02, 0.64

Work Efficiency. To evaluate the efficiency of work carried out by the teams in our sample, we used the median number of days that issues survived in a project. A series of Mann Whitney *U* tests revealed no significant differences between team types across team size classes. Boxplots of the distribution of median survival days are provided in Figure 5 and summary statistics and Mann-Whitney *U* tests results are presented in Tables 10 and 11, respectively. These results suggest that bots do not significantly alter this particular dimension of efficiency in teams, failing to provide support for Hypothesis 3.

A closer look at the number of issues reported in projects managed by human-only teams versus human-bot teams reveals that human-bot teams document, on average, far more issues than human-only teams (see Mean Number of Issues column in Table 10). This difference in issue documenting behavior is greatest between team types when the team is either medium-sized or large. This suggests that there is a positive association between the presence of bots and issue documentation, and this relationship is magnified in larger teams (i.e., more issues are identified and documented likely as a result of both bots and an increase in team size). Additionally, for human-only teams, the median number of days an issue remained unresolved and number of documented issues were both highest when the team was large and lowest when the team was small. This suggests that work efficiency in human-only teams appeared to be sensitive to differences in team size. This does not appear to be the case for human-bot teams—although the number of documented issues increased with team size, the difference in median number of survival days across team size classes is smaller.

[Figure 5 Placeholder]

Figure 5. Notched boxplots of median number of days that issues remain unresolved in a project. In this plot, lower values indicate greater efficiency. Data are grouped by team size and team type. There were no significant differences between team types across team size classes.

Table 10. Summary statistics for issue resolution. The mean and median values presented in this table have not been transformed.

Team Type	**Team Size**	**Number of Teams**	**Mean Number of Issues**	**Median of Median Issue Survival Days**
Human-only	Small	39	41.97	3
	Medium	43	52.74	5
	Large	30	232.97	11
Human-bot	Small	65	47.68	8
	Medium	55	102.55	11
	Large	71	422.49	9

Table 11. Mann-Whitney *U* test results and effect sizes for issue resolution by team type. As is standard for nonparametric approaches, median values are reported for each team type.

Team Size	***Median*$_{\textbf{Human-Only}}$**	***Median*$_{\textbf{Human-Bot}}$**	***U***	***p***	***abs(r)***

Small	1.73	2.83	1111	.58	0.10
Medium	2.24	3.32	992	.52	0.14
Large	3.32	3.00	1079	.92	0.01

Discussion

We set out to study an important technological change altering the future of work, namely, how the introduction of AI to the workplace alters team processes and outcomes. As intelligent technology is more common, researchers need to study how work is altered when a machine agent becomes a member of a team. The objective of this research was to explore the association between bots and productivity outcomes in open source projects. To this end, we analyzed GitHub event data generated by hundreds of teams. We used a multifaceted approach to characterize differences in productivity between human-bot teams and human-only teams. Our results suggest that variations in processes and productivity are associated with the presence of bots in open source project teams. We find that: 1) human-bot teams have higher levels of human activity in general and this is among a subset of the team, 2) human-bot teams have higher levels of work centralization, and 3) human-bot teams did not differ significantly in their efficiency when compared to human-only teams, but do show increases in coordination processes. We next discuss the implications of these findings for future research and the development of AI in support of teamwork.

In our analysis of productivity levels, we observed consistent differences in work per person across team sizes. Human-bot teams were more productive than human-only teams, with medium-sized human-bot teams showing the highest levels of productivity. This finding is important because our measure of productivity is work per person. This means that individual team members are showing greater productivity when bots are members of a team. This finding suggests that software bots play a role in altering individual output and momentum of work in GitHub repositories—for example, by prompting users to complete an action after code is pushed to the repository or sharing requested information through comments. In our data, this is evidenced by the higher levels of discussion observed in human-bot teams (see Table 4) and the large quantity of comment and push events produced by the bots in hybrid teams (Figure 2). This suggests that bots are fostering the collaborative component of the team and increasing the team's task output. An investigation of how bots affect collective productivity in addition to individual productivity may help clarify the cases (e.g., different phases of the development process) in which bots can best support individual and collaborative work as others have found evidence for the relationship between individual and collective output in GitHub (Murić et al., 2019).

Fitting these findings within our theoretical framework, the discussion around code artifacts illustrates a key bridging function between internal and external cognition as described in the Macrocognition in Teams Model (MITM). In particular, the macrocognitive function of knowledge building emerges from various forms of information sharing and deliberative communications on the part of teams. Thus, we found that human-bot teams engaged in more knowledge building activities around their code artifacts. This was also associated with increases in submissions and acceptances of changes to source code. Similarly, with regard to findings on issue comments, the human-bot teams were engaged in more knowledge building activities around deliberations of problems to be resolved.

Despite our findings on communications, it is common for open source project teams to use communication channels outside of social coding platforms (Storey et al., 2017). We therefore do not know to what degree, and how much, communication may have also been occurring outside of the repos in our study sample. Further, this could be to the detriment of team functioning. Specifically, when information and knowledge about the team's tasks and members are distributed across a potentially unmanageable number of tools and online spaces, this can attenuate performance. Not only might this increase workload by requiring attention to 'where' information and knowledge resides, but it also serves as a barrier to participation in the project by introducing learning challenges for project newcomers, both human and machine. Based on this we suggest that additional research is needed to understand how differences in the number and use of communication channels interact with the presence of bots to alter team growth and productivity in open source projects.

Our results also suggest that bots were associated with higher levels of work centralization. One interpretation of this finding is that the smaller subset of team members who did the majority of the work were more productive in human-bot teams than in human-only teams. On the one hand, given that bots are associated with increased human activity and centralization is related to team size, it is possible that bots are boosting the productivity of team members. On the other hand, it is possible that bots help increase user awareness and participation and this, in turn, increases the task workload of the team members who manage the project. Indeed, research on online innovation communities shows that new members are more likely to participate when other, more active members are responsive to them (Zhang et al., 2013) and, as previously mentioned, bots have been associated with increased workload for project integrators (Bernardo et al., 2018). Future research can build on these findings by examining how different types of bots can be leveraged to diminish or increase work centralization, and shift centralization among members of a team throughout the lifetime of a project.

To assess work efficiency, we examined the speed with which teams address documented issues. In our analysis of issue resolution times, there were no significant differences. But, given that this is exploratory research, and we used conservative levels of significance, the numerical differences are worth reporting to set the stage for future research. First, we found that the median value for issue survival days was lower for human-only compared to human-bot teams when the team was small or medium-sized. The difference between team types was reversed but less in large teams; that is, issues remained unresolved for fewer days in large human-bot teams compared to their human-only counterparts. Again, the differences between these teams were not statistically significant, limiting interpretation of our results. In future research, analysis of issue resolution times as a measure of work efficiency may be enhanced by evaluating the likelihood that an issue will be reopened after initial closure and examining differences in time to resolution based on issue topic and issue comment contents to capture nuances in problem-solving outcomes.

In the context of prior research, our results suggest that work centralization may have predictive utility with respect to variations in work efficiency. Relevant to efficiency in issue resolution, across team sizes, human-bot teams, on average, documented more issues than human-only teams. We can interpret this difference in the context of how the types of events generated by bots in our data affects coordination. Bots produced a large proportion of issue comments and thus likely played a significant role in information sharing and the coordination of work around issues in addition to automating the documentation of issues. Jarczyk et al. (2018) found that the centralization of work, specifically code modifications, was positively associated with issue closure in mature open source projects. Although we did not observe a similar trend across team size levels in our sample of relatively new open source projects, the presence of bots was positively associated with work centralization in small and medium-sized teams. These mixed findings reflect an opportunity to identify factors that moderate the relationship between efficiency and quality in growing projects.

These findings also contribute to the future of work for open source projects. As noted, we utilized a definition of work that was broader than traditional studies because it extended beyond code contributions to include teamwork aspects of open source development. This included for example, the coordination of work and knowledge building observed in comments, a process foundational to the MITM (Fiore et al., 2010). We consider the differentiation and study of teamwork and taskwork critical to advancing research in human-machine teaming. This is relevant to the broader open source landscape because social coding platform researchers whose findings inform the design of the online workspace are recognizing the importance of other types of contributions to collaboration that are not captured when focusing strictly on code. For example, in a recent report, GitHub not only emphasized the importance of discussions, but also included statistics to emphasize different aspects of teamwork and roles involved in open source projects. They note that this illustrates a shift showing that "the world of GitHub is growing—not just in numbers, but also in diversity, with existing users reimagining the projects they can build and how they collaborate" (*Octoverse 2020*, 2020, p. 9).

In sum, the data suggest that human-only teams and human-bot teams differ along a number of productivity dimensions, signaling a shift in work practices that warrants further investigation by the cognitive science community. From this, we can interpret how existing theory on team cognition provides a way forward for future research. First, in the context of the MITM, the use of shared artifacts seems to affect coordination by making concrete the extended cognitive system that can assist interactions (Fiore & Wiltshire, 2016). Specifically, OSS development bots contribute to coordination through the externalization of information, aiding human memory and prompting action. We suggest that research focusing on this area can help us understand additional ways bots can be used to create such artifacts and strategically organize them in the online work environment. In this way, research can study how to design AI to more specifically focus on what are traditionally abstract facets of team cognition, and help us understand how to ease, for example, memory loads on humans to improve overall coordination. Second, in the context of work centralization, our findings suggest that information gleaned from different distributions for different types and aggregations of work needs to be studied to understand changes in coordination behavior in online environments. This can be leveraged to examine varying team needs and performance differences in open source projects and identify opportunities for bots to support taskwork and/or teamwork.

Limitations

Although this is one of the first studies to examine real-world behavioral changes between human-only and human-machine teams, there are limitations that identify important future directions. First, the nature of the data collected did not allow us to conduct a natural experiment comparing team productivity before and after the introduction of machine agents in a project. Thus, our results, while informative, do not provide insights about the causal relationship between bots and team productivity. Future research should identify appropriate data collection methods for the study of work in real-world contexts that allow for pre- and post-type evaluation of technological interventions.

The approach used in this research can be extended to include the effects of bot adoption in addition to analyses based on temporal information that can elucidate the relationship between our measures of productivity and the concepts of stability and metastability introduced earlier in this work. For example, high degrees of work centralization may be associated with stability and the particular network structure characteristics (e.g., hierarchy, modularity, etc.) that emerge as open source projects grow, might reflect the conditions and interactions that are needed to achieve metastability. This could be manifested in terms of how expertise, knowledge, or influence is distributed among team members in a project over time, potentially resulting in the evolution of a network towards or away from stability and metastability. An analysis of burstiness applied to GitHub event data (Saadat et al., 2020) and phase transitions in collaborative problem solving (Wiltshire et al., 2018) observed around documented issues can complement such an approach to provide a more comprehensive understanding of how bots contribute to and alter team dynamics.

The interpretation of the analyses carried out in this study was primarily limited to comparisons within team size categories (e.g., small human-only teams were compared to small human-bot teams). This was due not only to the statistical tests selected based on the characteristics of the data and our research questions but also the measures we selected to operationalize variables of interest. Work centralization was quantified using the Gini coefficient, an approach susceptible to bias in small samples. In our research, this bias would result in lower Gini coefficient values for small teams compared to large teams with a proportional distribution of work. However, because we compared teams of similar sizes, we contend that this measure provides insights regarding the effects of bots on the distribution of work in open source projects.

Furthermore, because bot identification remains challenging for researchers, it is possible that the bot classification method described here failed to capture all automated activity. Approximately 8% of the accounts in our sample were identified as bots. Manual identification of bots in prior research has detected and labeled 11% of GitHub accounts as bots (Golzadeh et al., 2021), suggesting that our classification method may be relatively conservative. We also did not have access to usernames due to data sharing constraints. This limited our ability to conduct any aliasing to identify users who may have used multiple accounts to contribute to projects in GitHub. It is unclear whether this can be overcome, but if so, researchers can also examine how team cognitive factors, like team familiarity, might interact with variables of interest when studying human-bot teams. Lastly, the anonymized nature of the data did not allow for an analysis of differences in motivation between salaried and volunteer developers as it relates to productive output and bot adoption. Although prior research on motivation in open source projects provides evidence for the effects of both altruistic and economic incentives on participation (Baytiyeh & Pfaffman, 2010; Wu et al., 2007), less is known about the relationship between motivation and bot adoption.

Conclusions

We conducted an observational study of large-scale collaborations in distributed software development in which we characterized the role of bots in open source teams to better understand the future of work in complex sociotechnical systems. In this study, we found that only a small number of teams maintaining repositories in our sample are human-machine hybrids and most of these teams have less than two bots. Our results indicate that there are significant differences in productivity between human-bot teams and human-only teams and that human-bot teams generally have higher levels of productivity compared to human-only teams. Relevant to communication and efficiency, more issues were documented in human-bot teams than in human-only teams. Lastly, in considering social dynamics, bots remain limited in their ability to support teamwork, necessitating further research on developing socially intelligent machine agents (Fiore et al., 2021).

Our research characterizes the association between machine agents and team outcomes. The results of our analyses indicate that bots contribute to the productive capacity of teams engaged in distributed complex collaborative work. The findings presented here can help researchers in cognitive science and beyond conceptualize next steps for human-machine teaming research. In their current state, machine agents in the context of OSS development are able to take on a specific set of tasks with relative ease. Although the bots in our sample produced a high proportion of comments, their ability to engage in social interactions is greatly limited. Indeed, bot communications tend to be limited to predefined, or canned, messages. To more adequately engage in communication with humans, bots need to be more intelligent about not only the taskwork in a project but also about their teammates (Fiore & Wiltshire, 2016). We contend that human-machine teams built on a strong foundation of team theory are more likely to exhibit improved team performance.

In terms of practical implications, our results suggest there are tradeoffs associated with the adoption of bots in GitHub teams. Bots seem to be drivers of increased human activity in software development projects. High levels of activity and an influx of new contributors are both used as indicators of project success by developers (McDonald & Goggins, 2013). However, these patterns of user behavior can also result in higher workload for key developers in the projects. This was evidenced by Wessel et al.'s (2018) work showing that bot adoption results in an increase of pull request submissions and our finding that human-bot teams handle more issues than human-only teams. Additionally, the inclusion of many bots in a team may introduce challenges associated with the management of the bots themselves. Indeed, developers report that they experience notification fatigue as a result of automated activity (Mirhosseini & Parnin, 2017). The development of more socially intelligent bots may help reduce these issues. For example, machine agents able to derive social cues and signals from developer behavior to infer mental states may be able to detect the most appropriate time windows for notifications (Fiore et al., 2013).

Overall, this research provides important insights for the future of work. Given the changing nature of the sociotechnical systems, cognitive science must build both a theoretical and empirical foundation of research to help develop technologies for improving work, as well as inform design of intelligent technologies that can augment human performance. As intelligent machines become more prevalent parts of the labor landscape, we need to better understand how they influence cognition and collaboration.

Acknowledgements

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

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods: Review and analysis*. Otavamedia Oy. http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf.

Baltes, S., & Diehl, S. (2018). Towards a theory of software development expertise. *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering - ESEC/FSE 2018*, 187–200. https://doi.org/10.1145/3236024.3236061

Baytiyeh, H., & Pfaffman, J. (2010). Open source software: A community of altruists. *Computers in Human Behavior*, *26*(6), 1345–1354. https://doi.org/10.1016/j.chb.2010.04.008

Bernardo, J. H., da Costa, D. A., & Kulesza, U. (2018). Studying the impact of adopting continuous integration on the delivery time of pull requests. *Proceedings of the 15th International Conference on Mining Software Repositories*, 131–141. https://doi.org/10.1145/3196398.3196421

Blincoe, K., & Damian, D. (2015). Implicit coordination: A case study of the Rails OSS project. *IFIP International Conference on Open Source Systems*, 35–44.

Blincoe, K., Sheoran, J., Goggins, S., Petakovic, E., & Damian, D. (2016). Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology*, *70*, 30–39. https://doi.org/10.1016/j.infsof.2015.10.002

Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). Comparing data distributions. In P. J. Bickel, W. S. Cleveland, & R. M. Dudley (Eds.), *Graphical Methods for Data Analysis*. Wadsworth International Group.

Choudhary, S. S., Bogart, C., Rose, C. P., & Herbsleb, J. D. (2018). *Modeling coordination and productivity in open-source GitHub projects* (CMU-ISR-18-101; p. 23). Carnegie Mellon University.

Claggett, J. L., & Karahanna, E. (2018). Unpacking the Structure of Coordination Mechanisms and the Role of Relational Coordination in an Era of Digitally Mediated Work Processes. *Academy of Management Review*, *43*(4), 704–722. https://doi.org/10.5465/amr.2016.0325

Coleman, E. G. (2013). *Coding freedom: The ethics and aesthetics of hacking*. Princeton University Press.

Cooke, N. J., Gorman, J. C., Myers, C. W., & Duran, J. L. (2013). Interactive Team Cognition. *Cognitive Science*, *37*(2), 255–285. https://doi.org/10.1111/cogs.12009

Davidson-Pilon, C. (2014). *lifelines: Survival analysis in Python*. https://github.com/CamDavidsonPilon/lifelines/

Demir, M., Cooke, N. J., & Amazeen, P. G. (2018). A conceptual model of team dynamical behaviors and performance in human-autonomy teaming. *Cognitive Systems Research*, *52*, 497–507. https://doi.org/10.1016/j.cogsys.2018.07.029

Demir, M., Likens, A. D., Cooke, N. J., Amazeen, P. G., & McNeese, N. J. (2019). Team Coordination and Effectiveness in Human-Autonomy Teaming. *IEEE Transactions on Human-Machine Systems*, *49*(2), 150–159. https://doi.org/10.1109/THMS.2018.2877482

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, *85*(6), 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033

Erickson, I., Robert, L. P., Crowston, K., & Nickerson, J. V. (2018). Workshop: Work in the Age of Intelligent Machines. *Proceedings of the 2018 ACM Conference on Supporting Groupwork*, 359–361. https://doi.org/10.1145/3148330.3152159

Fiore, S. M. (2008). Interdisciplinarity as Teamwork: How the Science of Teams Can Inform Team Science. *Small Group Research*, *39*(3), 251–277. https://doi.org/10.1177/1046496408317797

Fiore, S. M., Bracken, B., Demir, M., Freeman, J., & Lewis, M. (2021). Transdisciplinary Team Research to Develop Theory of Mind in Human-AI Teams Panelists. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *65*(1), 1605–1609. https://doi.org/10.1177/1071181321651351

Fiore, S. M., Carter, D. R., & Asencio, R. (2015). Conflict, trust, and cohesion: Examining affective and attitudinal factors in science teams. In E. Salas, W. B. Vessey, & A. X. Estrada (Eds.), *Research* *on Managing Groups and Teams* (Vol. 17, pp. 271–301). Emerald Group Publishing Limited. https://doi.org/10.1108/S1534-085620150000017011

Fiore, S. M., Smith-Jentsch, K. A., Salas, E., Warner, N., & Letsky, M. (2010). Towards an understanding of macrocognition in teams: Developing and defining complex collaborative processes and products. *Theoretical Issues in Ergonomics Science*, *11*(4), 250–271. https://doi.org/10.1080/14639221003729128

Fiore, S. M., & Wiltshire, T. J. (2016). Technology as teammate: Examining the role of external cognition in support of team cognitive processes. *Frontiers in Psychology*, *7*, 1531. https://doi.org/10.3389/fpsyg.2016.01531

Fiore, S. M., Wiltshire, T. J., Lobato, E. J. C., Jentsch, F. G., Huang, W. H., & Axelrod, B. (2013). Toward understanding social cues and signals in human–robot interaction: Effects of robot gaze and proxemic behavior. *Frontiers in Psychology*, *4*. https://doi.org/10.3389/fpsyg.2013.00859

Fiore, S. M., Wiltshire, T. J., Oglesby, J. M., O'Keefe, W. S., & Salas, E. (2014). Complex Collaborative Problem-Solving Processes in Mission Control. *Aviation, Space, and Environmental Medicine*, *85*(4), 456–461. https://doi.org/10.3357/ASEM.3819.2014

Fronchetti, F., Wiese, I., Pinto, G., & Steinmacher, I. (2019). What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity. In F. Bordeleau, A. Sillitti, P. Meirelles, & V. Lenarduzzi (Eds.), *Open Source Systems* (Vol. 556, pp. 91–103). Springer International Publishing. https://doi.org/10.1007/978-3-030-20883-7_9

Goggins, S. P., Mascaro, C., & Valetto, G. (2013). Group informatics: A methodological approach and ontology for sociotechnical group research. *Journal of the American Society for Information Science and Technology*, *64*(3), 516–539. https://doi.org/10.1002/asi.22802

Golzadeh, M., Decan, A., Legay, D., & Mens, T. (2021). A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software*, *175*, 110911. https://doi.org/10.1016/j.jss.2021.110911

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of*

Statistics, *6*(2), 65–70. JSTOR.

Hukal, P., Berente, N., Germonprez, M., & Schecter, A. (2019). Bots coordinating work in open source software projects. *Computer*, *52*(9), 52–60. https://doi.org/10.1109/MC.2018.2885970

Jarczyk, O., Jaroszewicz, S., Wierzbicki, A., Pawlak, K., & Jankowski-Lorek, M. (2018). Surgical teams on GitHub: Modeling performance of GitHub project development processes. *Information and Software Technology*, *100*, 32–46. https://doi.org/10.1016/j.infsof.2018.03.010

Joblin, M., Apel, S., & Mauerer, W. (2017). Evolutionary trends of developer coordination: A network approach. *Empirical Software Engineering*, *22*(4), 2050–2094. https://doi.org/10.1007/s10664-016-9478-9

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., & Damian, D. (2014). The promises and perils of mining GitHub. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 92–101. https://doi.org/10.1145/2597073.2597074

Kaplan, E. L., & Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, *53*(282), 457–481. https://doi.org/10.1080/01621459.1958.10501452

Kidwell, R. E., & Bennett, N. (1993). Employee Propensity to Withhold Effort: A Conceptual Model to Intersect Three Avenues of Research. *Academy of Management Review*, *18*(3), 429–456. https://doi.org/10.5465/amr.1993.9309035146

Klein, J. P., & Moeschberger, M. L. (2003). *Survival analysis: Techniques for censored and truncated data* (2nd ed). Springer.

Larson, L., & DeChurch, L. A. (2020). Leading teams in the digital age: Four perspectives on technology and what they mean for leading teams. *The Leadership Quarterly*, *31*(1), 101377. https://doi.org/10.1016/j.leaqua.2019.101377

Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., Gutmann, M., Jebara, T., King, G., Macy, M., Roy, D., & Van Alstyne, M. (2009). Computational social science. *Science*, *323*(5915), 721–723.

https://doi.org/10.1126/science.1167742

Lebeuf, C. R. (2018). *A Taxonomy of Software Bots: Towards a Deeper Understanding of Software Bot Characteristics* [Masters, University of Victoria]. http://dspace.library.uvic.ca/bitstream/handle/1828/10004/Lebeuf_Carlene_MASc_2018.pdf

Lebeuf, C. R., Storey, M.-A., & Zagalsky, A. (2017). How software developers mitigate collaboration friction with chatbots. *ArXiv:1702.07011 [Cs]*. http://arxiv.org/abs/1702.07011

Lebeuf, C. R., Storey, M.-A., & Zagalsky, A. (2018). Software bots. *IEEE Software*, *35*(1), 18–23. https://doi.org/10.1109/MS.2017.4541027

Lebeuf, C. R., Zagalsky, A., Foucault, M., & Storey, M.-A. (2019). Defining and classifying software bots: A faceted taxonomy. *Proceedings of the 1st International Workshop on Bots in Software Engineering (BotSE '19)*, 1–6. https://doi.org/10.1109/BotSE.2019.00008

Mathieu, J. E., Heffner, T. S., Goodwin, G. F., Salas, E., & Cannon-Bowers, J. A. (2000). The influence of shared mental models on team process and performance. *Journal of Applied Psychology*, *85*(2), 273–283. https://doi.org/10.1037/0021-9010.85.2.273

McDonald, N., & Goggins, S. (2013). Performance and participation in open source software on GitHub. *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, 139–144. https://doi.org/10.1145/2468356.2468382

McNeese, M., Salas, E., & Endsley, M. R. (Eds.). (2020). *Fields of practice and applied solutions within distributed team cognition* (First edition). CRC Press.

McNeese, N. J., Demir, M., Cooke, N. J., & Myers, C. (2018). Teaming with a synthetic teammate: Insights into human-autonomy teaming. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *60*(2), 262–273. https://doi.org/10.1177/0018720817743223

Miller, I. D. (2016). *Gh-impact*. https://github.com/iandennismiller/gh-impact

Mirhosseini, S., & Parnin, C. (2017). Can automated pull requests encourage software developers to upgrade out-of-date dependencies? *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 84–94. https://doi.org/10.1109/ASE.2017.8115621

Murić, G., Abeliuk, A., Lerman, K., & Ferrara, E. (2019). Collaboration Drives Individual Productivity. *Proceedings of the ACM on Human-Computer Interaction*, *3*(CSCW), 1–24. https://doi.org/10.1145/3359176

Newman, M. (2005). Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, *46*(5), 323–351. https://doi.org/10.1080/00107510500052444

Newton, O. B., Fiore, S. M., & Song, J. (2018). Developing theory and methods to understand and improve collaboration in open source software development on GitHub. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *62*, 1118–1122. https://doi.org/10.1177/1541931218621256

Newton, O. B., Wiltshire, T. J., & Fiore, S. M. (2018). Macrocognition in Teams and Metacognition: Developing Instructional Strategies for Complex Collaborative Problem Solving. In *Building Intelligent Tutoring Systems for Teams (Research on Managing Groups and Teams* (Vol. 19, pp. 33–54). Emerald Publishing Limited.

NSF's 10 Big Ideas: Future of Work at the Human-Technology Frontier. (2020). [Government]. National Science Foundation. https://www.nsf.gov/news/special_reports/big_ideas/human_tech.jsp

O'Neill, T., McNeese, N., Barron, A., & Schelble, B. (2020). Human–Autonomy Teaming: A Review and Analysis of the Empirical Literature. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 001872082096086. https://doi.org/10.1177/0018720820960865

Onoue, S., Hata, H., & Matsumoto, K. (2013). A study of the characteristics of developers' activities in GitHub. *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*, 7–12. https://doi.org/10.1109/APSEC.2013.104

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Python Software Foundation. (2019). *Python programming language*. https://www.python.org/

R Core Team. (2019). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/.

Rey, S. J., & Anselin, L. (2010). PySAL: A Python Library of Spatial Analytical Methods. In M. M. Fischer & A. Getis (Eds.), *Handbook of Applied Spatial Analysis: Software Tools, Methods and Applications* (pp. 175–193). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03647-7_11

Reyes López, A. (2017). *Analyzing GitHub as a Collaborative Software Development Platform: A Systematic Review* [Masters]. University of Victoria.

Rico, R., Sánchez-Manzanares, M., Gil, F., & Gibson, C. (2008). Team implicit coordination processes: A team knowledge–based approach. *Academy of Management Review*, *33*(1), 163–184.

Romero, D. M., Huttenlocher, D., & Kleinberg, J. M. (2015). Coordination and efficiency in decentralized collaboration. *Proceedings of the 9th International AAAI Conference on Web and Social Media*, 367–376.

Saadat, S., Newton, O. B., Sukthankar, G., & Fiore, S. M. (2020). Analyzing the Productivity of GitHub Teams based on Formation Phase Activity. *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 169–176. https://doi.org/10.1109/WIIAT50758.2020.00027

Saadat, S., & Sukthankar, G. (2020). Explaining Differences in Classes of Discrete Sequences. *ArXiv:2011.03371 [Cs]*. http://arxiv.org/abs/2011.03371

Salas, E., Cooke, N. J., & Rosen, M. A. (2008). On Teams, Teamwork, and Team Performance: Discoveries and Developments. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *50*(3), 540–547. https://doi.org/10.1518/001872008X288457

Solanas, A., Selvam, R. M., Navarro, J., & Leiva, D. (2012). Some common indices of group diversity: Upper boundaries. *Psychological Reports*, *111*(3), 777–796. https://doi.org/10.2466/01.09.21.PR0.111.6.777-796

Storey, M.-A., & Zagalsky, A. (2016). Disrupting developer productivity one bot at a time. *Proceedings* *of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, 928–931. https://doi.org/10.1145/2950290.2983989

Storey, M.-A., Zagalsky, A., Filho, F. F., Singer, L., & German, D. M. (2017). How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering*, *43*(2), 185–204. https://doi.org/10.1109/TSE.2016.2584053

The 2020 State of the Octoverse. (2020). GitHub: Octoverse. https://octoverse.github.com/static/github-octoverse-2020-community-report.pdf

Vasilescu, B., Posnett, D., Ray, B., van den Brand, M. G. J., Serebrenik, A., Devanbu, P., & Filkov, V. (2015). Gender and tenure diversity in GitHub teams. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, 3789–3798. https://doi.org/10.1145/2702123.2702549

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., & Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in GitHub. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*, 805–816. https://doi.org/10.1145/2786805.2786850

von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, *32*(7), 1217–1241. https://doi.org/10.1016/S0048-7333(03)00050-7

Walliser, J. C., de Visser, E. J., Wiese, E., & Shaw, T. H. (2019). Team Structure and Team Building Improve Human–Machine Teaming With Autonomous Agents. *Journal of Cognitive Engineering and Decision Making*, *13*(4), 258–278. https://doi.org/10.1177/1555343419867563

Weiss, M., & Hoegl, M. (2016). Effects of relative team size on teams with innovative tasks: An understaffing theory perspective. *Organizational Psychology Review*, *6*(4), 324–351. https://doi.org/10.1177/2041386615620837

Wessel, M., de Souza, B. M., Steinmacher, I., Wiese, I. S., Polato, I., Chaves, A. P., & Gerosa, M. A.

(2018). The power of bots: Characterizing and understanding bots in OSS projects. *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 1–19. https://doi.org/10.1145/3274451

Wiltshire, T. J., Butner, J. E., & Fiore, S. M. (2018). Problem-Solving Phase Transitions During Team Collaboration. *Cognitive Science*, *42*(1), 129–167. https://doi.org/10.1111/cogs.12482

Wiltshire, T. J., Warta, S. F., Barber, D., & Fiore, S. M. (2017). Enabling robotic social intelligence by engineering human social-cognitive mechanisms. *Cognitive Systems Research*, *43*, 190–207. https://doi.org/10.1016/j.cogsys.2016.09.005

Wu, C.-G., Gerlach, J. H., & Young, C. E. (2007). An empirical analysis of open source software developers' motivations and continuance intentions. *Information & Management*, *44*(3), 253–262. https://doi.org/10.1016/j.im.2006.12.006

Yu, Y., Yin, G., Wang, H., & Wang, T. (2014). Exploring the patterns of social behavior in GitHub. *Proceedings of the 1st International Workshop on Crowd-Based Software Development Methods and Technologies (CrowdSoft 2014)*, 31–36. https://doi.org/10.1145/2666539.2666571

Zhang, C., Hahn, J., & De, P. (2013). Research note—Continued participation in online innovation communities: Does community response matter equally for everyone? *Information Systems Research*, *24*(4), 1112–1130. https://doi.org/10.1287/isre.2013.0485