Jason Yau

Professor Hensley

ENC 1102

26 July 2023

Review of *Understanding Collaborative Software Development: An Interview Study*

For my semester-long paper, I'd like to research how communication and collaboration in software development are needed to facilitate smooth development. I'd like to do a review of one scholarly article relating to this topic by Kattiana Constantino et al., who are affiliated with the Federal University of Minas Gerais in Brazil and/or Carnegie Mellon University in the United States and have a background in Computer Science. The writer also agrees that collaboration is important, particularly in fork-based software development, where users interested in contributing create a copy of the original source code, modify it, and request for the project maintainer to review and integrate the modifications to the main code base, and seeks to understand exactly why and how the collaboration happens in an interview-based study.

The article first describes some basic terminology, such as open-source software (OSS), where programmers develop software and post the source code in public for free, and GitHub, the most popular website where most open-source code is hosted. Then, the article writes about previous similar studies, such as a study by Gousios et al., who investigated how to keep collaborators of a software project motivated by reducing response times, better documentation, higher quality code, etc., and another study by Marlow et al., who observed how developers use GitHub's social features to get an understanding of the level of communication in a project and then reflect on the project's success (2). Overall, the introduction was done well and ensured that the reader understood the programming lingo and prior context of their study. The writers then propose three research questions, which deal with the motivations, methods, and challenges of collaboration in software projects, and write a concise explanation of why they want to further investigate the questions (3). Then, they write how they conducted the interview: they found twelve Portuguese OSS contributors on GitHub with over 500 commits. The interviewees could have been selected better; the writers targeted only the top collaborators on GitHub who spoke Portuguese, and the sample size was also very small; only 12 of the 80 contributors responded to the writers' email, so the sample was not very general to the entire population of software engineers.

By the end of the interview, they concluded that the interviewees preferred to work collaboratively for knowledge sharing, increased productivity, and fewer software

bugs (4-5). They found that the main types of contributions were developing features, code review, issue solving or reporting, and writing or translating documentation, and the main mediums of communication were GitHub Issues and Pull Requests (a forum for GitHub repositories) and Email (5–6). Additionally, they found some challenges of collaborative software development to be challenges of effectively managing collaborators, helping newcomers, inconsistent documentation, lack or loss of contributors, and no compliance with the project guidelines (7). Their conclusion was done very well, as each of the three research questions was specifically and clearly answered, and they provided frequency tables on the number of interviewees who answered a particular way, which concisely summarizes their findings and gives the reader a better overall experience.

Overall, the article conducted their research very well, and their findings mostly reaffirm my argument that collaboration is needed in software development and has vast benefits if communication is done well, as it provides interviewee testimony in a concise manner. The intended audience is likely college students and/or software engineers with less experience in the industry. However, I felt that there were some missing pieces; for example, the writers could have included the specific interview questions they asked to ensure that the questions asked were consistent and weren't biased or leading in nature, which would provide further confidence in the research findings.

Works Cited

Constantino, Kattiana, et al. *Understanding Collaborative Software Development: An Interview Study*. Carnegie Mellon University, 2020, https://www.cs.cmu.edu/~ckaestne/pdf/icgse20.pdf.

Jason Yau

Professor Hensley

ENC 1102

26 July 2023

Plan to Study the Importance of Communication in Software Development

With the rise of remote work, especially in the field of software development, it is a common misconception that programming requires little collaboration, and much is done in isolation. Software has become increasingly complex and larger throughout the years, and it would be infeasible to code with little communication; Google's codebase consists of approximately two billion lines as of 2016 (Potvin and Levenberg). As a Computer Science (CS) major interested in software engineering, I would like to explore the importance and effects of communication in software development further by proposing two research questions. What are the methods of communication in software engineering? For this question, I would like to see what tools, development practices, and open-source code programmers use in their jobs and personal hobby projects. What are the effects of communication in software engineering? With this question, I would like to identify the most common benefits reported by programmers when using best communication practices and see if any negative effects are reported.

To answer the proposed research questions, my plan is to conduct interviews and surveys. I believe this will be the easiest method to gather the most data regarding my research questions. I would most likely choose college students with job experience or who regularly create personal projects. This will be the easiest and most available demographic to interview or survey; I can post a survey link on the University of Central Florida's (UCF) CS Discord server and other UCF CS clubs' Discord servers, such as Knight Hacks. I could also interview students in collegiate hackathons, or events where the goal is to create functional real-world software with a team or individually, at college campuses such as UCF, Georgia Tech, FIU, MIT, etc., throughout the year. According to Warner and Guo, in 2016, collegiate hackathons sponsored by Major League Hacking had over 65,000 students across 200 events, so interviewing at a hackathon would be ideal as there is a large conglomerate of collaborative computer science students. I would try to interview as many students as possible, as they would give the most genuine answers; however, it may be challenging and time-consuming to interview a large body of students, so I will likely mostly have survey responses with some interview responses. This research plan is modeled similarly to the study from Constantino, Kattiana, et al., who also decided to interview a select group of collaborative software engineers to obtain data to understand exactly why and how collaboration happens in fork-based development, where users interested in

contributing will create a copy of the original source code, modify it, and request for the project maintainer to review and integrate the modifications to the main code base.

Additionally, I have compiled a list of questions to ask both interviewees and survey participants. My first question would be: "Do you have job experience in the field of software engineering or have developed a software project? Please describe." This question is just to gauge if they qualify for the interview or survey and to get some context on exactly what they do. The second question would be: "What tools and/or practices do you use in the process of development and testing? How do they relate to communication between programmers?" This question would attempt to answer the first research question on how programmers use communication. With this question, I would create a table sorted by the frequency of tools and practices that relate to communication reported by the respondents to better organize the data. This question is similar to the study by Theunissen, Theo, et al., who found that the tool stack, or the "set of tools to produce a software product," leads to better communication and comprehension (11). For the third question, I would first provide the definition of open-source software (OSS), where programmers develop software and post the source code in public for free, then ask: "What open-source projects have you used in the past six months?" I would ask this question since OSS is a great example of communication as it allows programmers to view, modify, update, and distribute collaboratively to create a product, and OSS is often dependent on one another to function. I would then create a frequency table like the one in the second question. My fourth question would be: "How do you think these tools, practices, and open-source software have affected your teams' productivity?" This question would attempt to answer the second research question on the effects of communication on software engineers. This question is similar to the study by Zhou et al., who found that there are inefficiencies in collaborative fork-based development, such as "lost contributions, rejected pull requests, redundant development, and fragmented communities." (10) Since this question is more open-ended, I would try my best to summarize both the most common positive and negative effects of communication and determine if one side outweighs the other. For the fifth and final question, I would ask: "Do you prefer creating large software projects individually or collectively? Why?" This would be a simpler type of question where participants could give a quick answer. Since there are only two possible answers, I would create a pie chart for preference for individual versus group, which would create a very clear and concise visualization of the data.

Afterwards, I would write a report of all my findings. Under the headings for each research question, I would summarize the common responses to the interview questions that pertain to the respective research question and include any visual charts I created. Then, I would conclude whether my proposed argument that communication and collaboration are needed in software engineering was correct or not based on my findings.

Works Cited

Potvin, Rachel, and Josh Levenberg. “Why Google Stores Billions of Lines of Code in a Single Repository.” *ACM*, 1 July 2016, https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext.

Warner, Jeremy, and Philip J. Guo. “Student Perceptions of College Hackathons”, *University of San Diego*, 2017, https://pg.ucsd.edu/publications/student-perceptions-of-college-hackathons_ICER-2017.pdf.

Working Bibliography

Zhou, Shuru, et al. “What the Fork: A Study of Inefficient and Efficient Forking Practices in Social Coding”, *ACM*, 2019, https://dl.acm.org/doi/pdf/10.1145/3338906.3338918.

This study found that there are inefficiencies in collaborative fork-based development, such as: “lost contributions, rejected pull requests, redundant development, and fragmented communities.

Theunissen, Theo, et al. “In Continuous Software Development, Tools Are the Message for Documentation”, *Utrecht University*, 2021, https://dspace.library.uu.nl/bitstream/handle/1874/417159/103679.pdf.

This study found that the tool stack, or the “set of tools to produce a software product,” leads to a better communication and comprehension.

Constantino, Kattiana, et al. *Understanding Collaborative Software Development: An Interview Study*. Carnegie Mellon University, 2020, https://www.cs.cmu.edu/~ckaestne/pdf/icgse20.pdf.

My research plan in conducting interviews on a group of collaborative software engineers is modeled after this study.