Code Ownership and Quality: Building a "Component Health Service" at Appian Corporation

(Technical Paper)

Code Ownership in Organizations and its Effects on Code Health

(STS Paper)

A Thesis Prospectus Submitted to the

Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements of the Degree
Bachelor of Science, School of Engineering

Ethan Gahm

Fall, 2022

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Author Name

Approved _____ Date _____

Capstone/Technical Advisor Name, Department

Approved _____ Date _____

STS Advisor: Richard D. Jacques, Ph.D., Department of Engineering & Society

Introduction

Within organizations that produce software, *code ownership* refers to which specific developers are allowed to modify and contribute to particular pieces of code (Bird, et. al., 2011). *Team code ownership* is a specific model whereby portions of a code base are assigned to a particular team or organizational subdivision and all members of that team are allowed to make changes to any part of the team's code (Sedano et. al., 2016). Ownership is handled differently by different companies and software development communities, with some organizations taking a looser approach than others, allowing larger or smaller numbers of developers to make changes to each piece of code. The term *contribution* can also be defined loosely. In one study, Thongtanunam et. al. (2016) expand the scope of code contributions to include code review, arguing that modern day software engineers spend a significant portion of their time reviewing and suggesting changes to their peers' code without making direct changes.

Significant research has explored how different levels and/or approaches to code ownership can affect code quality. Bird et. al. (2011) examined code ownership in two large software projects—Windows Vista and Windows 7—and found that a smaller number of experienced developers contributing to a piece of code correlated with fewer faults found in that code. In contrast, Foucault et. al. (2014) failed to find similar correlations when examining large open-source projects. Greiner et. al. (2015) expanded on the findings of the previous two studies and examined ownership on a lower level, considering ownership of particular files and directories instead of entire, large-scale binaries. They argue that this model for code ownership is more meaningful and more actionable, since code-ownership policies can be more readily implemented on this level. Ultimately, Greiner et. al. confirm the findings of Bird. et. al. as they do find significant correlation between strength of code ownership and code quality.

Technical Topic

My technical report will explore and reflect on my experience as a software engineering intern at Appian Corporation between June and August of 2022. At Appian, I worked as a member of the "Development Insights" squad, which attempts to use metrics and metric analysis to improve developer efficiency, code quality, and quality of life for developers within the company. My main focus during the summer was on building a new application called the Component Health Service. The project began by dividing Appian's codebase into a number of "components," usually defined as a single repository, but sometimes consisting of a smaller sub-module within a repository. The goals of the project were then to provide automated analysis of A) which developers were contributing to each component, and B) how "healthy" each component was. Although it is not *always* strongly defined, a notion of team code ownership exists within Appian. In the case of goal A, we wanted to see whether the contributors to a component matched up with the theoretical "owners" of that component—that is, the members of the team that was supposed to own that component. In the case of goal B, code "health" was determined by examining various metrics as measured by SonarQube, a static code analysis tool which programmatically examines source code and provides reports on code quality. Research has shown that the continual monitoring of code quality within an organization, like we aimed to do with the Component Health Service, can help to expose both system-level and lower-level issues, lending some legitimacy to our project (Kothapalli et. al., 2011).

The Component Health Service was built as a cloud-based Python application and deployed on Appian's internally managed Kubernetes cluster. The backend used the Python FastAPI library to construct a number of endpoints which could be used to retrieve data

regarding code contributions or code quality (as measured by SonarQube) of particular components. While these endpoints proved useful for testing purposes, they were eventually supplanted by an internal scheduler, written using the "Rocketry" library for Python. This scheduler would periodically trigger scanning/data collection. Data was aggregated and stored in a PostgreSQL database deployed in an adjacent Kubernetes pod. The database included tables for recording component contributions as well as SonarQube metrics. A third "component health" table was constructed by combining data from the other two. This third table consolidated core metrics and attempted to summarize the overall health of each component. The final application was deployed to Kubernetes using the "Helmfile" library and a set of "GitLab Runners" (scripts that were automatically triggered as part of a pipeline when code was pushed to a remote repository on GitLab).

My experiences working at Appian on the Component Health Service strongly influenced my choice of topic for my STS research. At Appian, I experienced first-hand how poorly defined code ownership could lead to confusion and to the neglection of maintenance for critical infrastructure. Perhaps more importantly, I discovered how poorly assigned code ownership (that is, assigning ownership of a component to the wrong developer or developers) can lead to infuriating inefficiency. In many ways, the Component Health Service was an attempt to address these concerns within Appian.

STS Topic

My STS research will explore the relationship between code ownership and code quality and the impacts on society of high versus low quality code. Tornhill and Borg (2017) examine correlations between code health, as measured by a static code analysis tool, and observable measures of code robustness including the number of reported defects and time to resolve

defects. They determine that “low quality” code contains fifteen times more defects, requires 124% more time in development to resolve issues, and has nine times longer cycle times than “high quality” code. This serves as compelling evidence that there is real value in studying the relationship between code ownership and code health. By better understanding this relationship we may adopt approaches to code ownership that result in faster, safer code deployment.

Dawson et. al. (2010) examined the differences in cost associated with catching and addressing software defects at different stages in the software development lifecycle. They found that it was 100 times more costly to fix software bugs discovered during maintenance than those discovered during the design phase, and six to seven times more costly than fixing bugs discovered during testing.

Historically, code defects have been responsible for major economic failures, threats to cyber security, and have cost human lives. In 2008, a bug in Heathrow Airport’s baggage reconciliation software resulted in travelers’ bags being misplaced and hundreds of flight cancellations, ultimately costing the airport tens of millions of dollars (Reuters, 2008). A bug in the popular Java-based Log4j library, revealed in 2021, allowed hackers to execute arbitrary code on remote servers, potentially compromising hundreds of millions of devices world wide (Lyngaas, 2021). Although the bug was swiftly patched in a new release of the library, it was the responsibility of individual organizations to update their software to this newer version of the library, meaning that the bug continues to linger and threaten application security to this day. In 2018, a software bug in Uber’s self-driving car software resulted in the death of one person when a car detected a pedestrian, but erroneously judged the detection to be a “false positive” (Lee, 2018). These are just a few of the many ways in which faulty software systems have had meaningful, negative impacts on society as a whole.

Software is written by humans and humans are fallible. If we demand robust software then we must do our best to build organizational systems around software developers that allow them to do their best work. Intelligent assignment and management of code ownership is a critical step in this process and more research is needed to assess the benefits and drawbacks of different code ownership models.

Research Methods

My research will focus on two central research questions: A) what systems are currently used by major organizations for assigning and managing code ownership, and B) what are the benefits and drawbacks to each approach? Specifically, with regards to question B, I would like to examine the impacts of different code ownership models on both code quality and code velocity. I define *code velocity* as the rate at which a company or organization can produce software. It seems plausible that even if a tighter code ownership model (one where fewer developers are allowed to contribute to each piece of code) results in healthier, more robust code, it may come at the cost of efficiency (that is, decreasing code velocity).

To answer these questions I will begin with a thorough review of available research. This paper has already mentioned several studies that have examined the relationships between code ownership and code quality. I would like to take a deeper dive into these studies and perform a loose meta-analysis in order to summarize collective findings. The relationship between code ownership and code velocity seems to be more poorly studied, however I will expand my search and attempt to draw evidence and conclusions from whatever I can find.

To answer the first of my research questions I hope to speak directly with current and former employees of software companies and organizations. My time spent working at Appian

left me curious about whether other companies had radically different approaches to assigning code ownership and whether Appian might be able to improve its own approach. By conducting interviews with software engineers across different organizations and sub-fields I hope to get a better understanding of what approaches exist and to gather anecdotal evidence about the potential benefits and challenges associated with each.

Conclusion

Through my summer internship experience, my technical report, and my STS research, I hope to gain a strong understanding of code ownership and its implications for software companies and society as a whole. Through building the Component Health Service for Appian, I made a meaningful contribution towards improving the company's own code ownership practices and towards monitoring code quality across the company's codebase. My experiences at Appian piqued my interest about code ownership as a larger issue and inspired me to explore what other approaches to managing code ownership might exist. In my STS paper I will explore the relationship between different code ownership models, code quality, and code velocity and provide loose recommendations for what code ownership models might be best for organizations to employ. Through this work I hope to make a meaningful contribution to the software development field and contribute to the development of more robust software systems.

References

Bird, C., Nagappan, N., Murphy, B., Gall, H., & Devanbu, P. (2011). Don't touch my code! *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering - SIGSOFT/FSE '11*, 4–14. https://doi.org/10.1145/2025113.2025119

Dawson, M., Burrell, D., Rahim, E., & Brewster, S. (2010). Integrating Software Assurance into the Software Development Life Cycle (SDLC). *Journal of Information Systems Technology and Planning 3* , 49–53.

Foucault, M., Falleri, J.-R., & Blanc, X. (2014). Code ownership in open-source software. *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, 1–9. https://doi.org/10.1145/2601248.2601283

Greiler, M., Herzig, K., & Czerwonka, J. (2015). Code ownership and software quality: A replication study. *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. https://doi.org/10.1109/msr.2015.8

Kothapalli, C., Ganesh, S. G., Singh, H. K., Radhika, D. V., Rajaram, T., Ravikanth, K., Gupta, S., & Rao, K. (2011). Continual monitoring of Code Quality. *Proceedings of the 4th India Software Engineering Conference on - ISEC '11*, 175–184. https://doi.org/10.1145/1953355.1953379

Lyngaas, S. (2021, December 14). *US warns hundreds of millions of devices at risk from newly revealed software vulnerability | CNN politics*. CNN. Retrieved October 30, 2022, from https://www.cnn.com/2021/12/13/politics/us-warning-software-vulnerability

Sedano, T., Ralph, P., & Péraire, C. (2016). Practice and perception of team code ownership. *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*. https://doi.org/10.1145/2915970.2916002

Thomson Reuters. (2008, April 5). *Computer glitch causes Heathrow Terminal 5 woes*. Reuters. Retrieved October 30, 2022, from https://www.reuters.com/article/us-britain-heathrow/computer-glitch-causes-heathrow-terminal-5-woes-idUSL0551899620080405

Thongtanunam, P., McIntosh, S., Hassan, A. E., & Iida, H. (2016). Revisiting code ownership and its relationship with software quality in the scope of modern code review. *Proceedings of the 38th International Conference on Software Engineering*, 1039–1050. https://doi.org/10.1145/2884781.2884852

Tornhill, A., & Borg, M. (2022). Code red. *Proceedings of the International Conference on Technical Debt*. https://doi.org/10.1145/3524843.3528091