Maintenance vs Growth: Two Sides of the Same Coin

The Exploitation of the Open-Source Philosophy

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

Eric Knocklein

Fall, 2022

Introduction

This paper is divided into two parts: the technical topic and the STS topic. The first is focused on my time working as an intern at a Virginia-based technology company. In particular, it discusses the balance between maintenance and growth that is inherent to development of all products but to software in particular. The work I did, its outcomes, and how I was empowered to do that work by UVA courses will be discussed in this section. Additionally, I will discuss what future work is required for optimal performance of the system.

The latter part of this report will discuss the social and moral implications of open-source software and the exploitation of the open-source philosophy. Open source code and, more broadly, open source software (OSS) are bodies of code where the code is open to modification and examination. The open source philosophy is a general way of thinking that holds that programming is not purely about making money but about forming a collaborative community.

According to a report by the 2022 Open Source Security and Risk Analysis (OSSRA) report written by Synopsys, Inc, about 99 percent of all codebases include some form of open-source code. Any technology that is so prevalently used must be analyzed critically in both usage and production to ensure that all technologies built upon it are morally and ethically upright. Thousands of companies profit off of the work of open-source developers, but are these developers being adequately compensated? Are they being recognized?

It is part of the so-called “open-source philosophy” to share code with others freely and to be able to depend on others in turn for code. In such an interconnected web, it can be difficult to determine ownership or rights. With the prevalence of the use of open-source components accelerating, it is important to answer these questions before thousands of developers are treated unjustly.

Technical Topic

Abstract

A Virginia-based technology company with a long history sought to grow and modernize their service but was held down by legacy code and a lack of maintenance. Using concepts and best-practices gathered in my classes, I aided in both the growth and maintenance of these services by fixing bugs (modernizing the code) and implementing new features. I used concepts that are often seen as inconsequential to the development of programs: in-line documentation, descriptive variable names, modularization, etc. To achieve our goals for this task, the team and I had to communicate effectively both within and outside the codebase. Though my project has only minimal outcomes at the moment, the outcomes are, nevertheless, crucial for the continued success of the product. Because both maintenance and growth are continual processes, future work includes adding structure to the codebase and increasing its modularity.

What is more important: adding new features to a product or maintaining already-implemented features? This is a common question for many companies as they allocate their limited resources. Each has its own benefits, but maintenance is often neglected in favor of growth..

Growth is the flashier of the disciplines. It is usually much more satisfying for both developers and managers to create new features than to maintain old features. New features draw in new customers with the promise of future prospects. On the other hand, maintenance is inherently a retrospective act. It serves to retain already-existing customers. Maintenance serves more than already-existing customers, though, because a poorly maintained software will not gain any new customers, either. Because of this, it is arguably more important than growth. However, the two disciplines cannot so easily be separated, for maintenance provides a foundation from which the program can grow. Growth, in turn, if done sustainably, can ease the burden of maintenance. Because of this, growth is only viable in the long term if it is done sustainably and if the program is well maintained.

STS Topic

Introduction

While there is some research into the prevalence of open-source code, most of this research focuses on the use of the code and how that relates to security. While this is undeniably important and will be discussed here, this report is focused more so in the production of open source code and how its philosophy can be exploited. Open source software (OSS), its use and production will be examined through a Motivation-Practice Theory perspective as well as one concerned with Virtue Ethics. Some of the questions that this report will seek to answer are: How is open source software developed and distributed? How are the developers of that software compensated? Is this compensation adequate? How is their work used? And what broader social questions are raised by this system?

Literature Review

As mentioned above, much of the literature in this area is done on how open-source software influences the security of software. Primary amongst this literature is the aforementioned 2022 Open Source Security and Risk Analysis (OSSRA) report. This report is an exhaustive review of the state of the security of open source code and components. It emphasizes the need for continual maintenance of codebases particularly in how they are affected by open source code and the width of influence of open source code. According to these reports, 99% of codebases contain some form of OSS component and "80% of the codebases were composed of open source" (Synopsis, 2022, Page 12). Additionally, "Twenty-three percent of open source projects have only one developer contributing the bulk of code. Ninety-four percent of the projects have fewer than 10 developers accounting for more than 90% of the lines of code" (Synopsis, 2022, Page 20). These statistics reveal the scope of the technology and thereby the necessity of reports such as the others reviewed in this paper. They also emphasize the fact that many OSS projects are developed individually or in small groups instead of being developed by large companies. This fact could point to the need for more protection for these developers, as they do not have the backing of large companies to protect them.

More in line with the main topic of this paper is the paper titled, "Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development", which reviews much of the available literature in the field and compiles it along with providing its own insights. It suggests that the current view of the motivations for development of open source software is too minimal and shallow to reveal any truths about the nature of open source software. It offers a framework by which to view open source software: the motivation-practice framework, which we will use later in this paper.

This paper discusses the interaction between proprietary software and OSS, concluding that their relationship is a complementary one. Or, rather, OSS complements proprietary software in that it nurtures a culture that benefits that proprietary software and provides experience for future developers for that software.

To better understand the incentives and motivations of OSS developers, we consult a foundational work in OSS: “The GNU Manifesto”. This document summarizes the foundations of OSS philosophy. It claims that OSS will bridge the gap between producers of OSS and consumers of OSS because every consumer can freely engage with the source code of the software, making them a producer. Producing such a coalition between developer and user stands in opposition to traditional software vendors, who “want to divide the users and conquer them” (Stallman, 1985). Within this coalition, developers and users are encouraged to share code because “The fundamental act of friendship among programmers is the sharing of programs” (Stallman, 1985).

More broad in scope is C. Doctorow and R. Giblin’s book *Chokepoint Capitalism*, which describes a number of exploitations of creative works. Of particular note for this discussion is the idea of a minimum wage for creative work (found in chapter 17). While not directly about OSS, the points made by the authors about restoring rights to those who do work that is currently unpaid or created without the promise of payment. This describes both musical artists, who are the main focus of much of the chapter, and OSS developers. If exploitation indeed exists within the OSS community, the ideas in this book may be valuable in suggesting a more just system.

Stakeholders

There are two primary groups of stakeholders in this discussion: those producing OSS and those consuming OSS. These groups, however, are not mutually exclusive because OSS developers can also integrate other OSS projects into their open-source code. After all, a large part of the open-source philosophy is the collaborative nature of the industry. Therefore, a more useful distinction would be those within the OSS community and those building other software with OSS components who don't reside within the OSS community.

An important distinction should be made between developers and executives within companies that use OSS components. This divide does not exist as sharply in purely OSS endeavors because OSS is not developed as consistently in large companies, as was shown before. A division like this is important because such executives hold power over their workers, so the actions of the workers can be seen as being caused by the executives. As such, use of OSS components by workers is also a use of OSS components by executives.

Up until this point, we have not considered those who use OSS and other software with OSS components simply as users and not developers. This is a crucial group to consider because it is, in the end, the group towards which all this software we are discussing is geared. While the other groups do have the power to change the system within which they operate, this is only likely to occur when this group applies pressure to the others. However, in OSS it is sometimes intentionally difficult to separate users from developers, as is the entire goal of the open-source philosophy.

Frameworks

We will discuss OSS in relation to the Motivation-Practice Framework proposed by von Krogh, Haefliger, Spaeth, and Wallin. Within this framework, actions and motivations are considered in how they interact with social practice, institutions, and internal and external good. This framework allowed these authors to formulate and answer three questions about OSS that will be summarized here.

First, the paper concludes that OSS developers interact with the social practice of OSS by producing internal good to that social practice (i.e. they enhance the social practice through their work). In turn, the enhancement of the social practice through the development of this good alters the standards of the work done for that social practice. This contrasts with work in regular institutions, in which more of a one-way relationship from worker to institution exists. In some ways, it seems more apt to think of OSS development not as individuals but as the social practice as a whole, being composed of individuals.

The second conjecture of the paper under the framework deals with the method by which OSS developers change institutions. As before, this is not a unary relationship. The paper conjectures that "2: OSS developers change institutions when and where these institutions no longer protect sufficiently the standards of excellence of the social practice" (Georg von Krogh, et al, 2012, Page 667).

Lastly, the paper examines the functioning of the social practice and how it is sustained by developers. It conjectures that social practice and its standards create the motivation for developers to uphold it. This is done through the fostering of a community where actions have true and visible consequences upon the standards and course of the community as a whole. In such a community, actions are meaningful, even mundane ones like bug-fixing. Work with meaning is more appealing than work without, which encourages people to work in this field.

There are also the future prospects of the developers to consider. Many companies look for developers who have worked within OSS communities. In order to seem appealing to these companies, some developers might accept a low paying or volunteer role within OSS in hopes of landing a job at a large company. In this way, the company is benefiting both from the free software developed by the OSS community but also the training, the standards, and the culture of the OSS community. Now that the motivations and nature of the OSS community are more understood – or are more able to be understood through the use of the framework – it can be critically examined.

This examination will focus on the Virtue Ethics of the OSS community and those of the companies who use OSS or hire OSS developers. Where motivation-practice theory considers the motivation of individual actors, virtue ethics examines their actions. In this analysis, we must ask whether the use of OSS software corresponds to virtuous human ideals such as fairness and justice.

Methods

To conduct further research into this field, I intend to focus on two main methods of research and analysis: literature review and ethical analysis. The first is meant to provide me with more information about the state of OSS and its developers, how it is used and how its developers are compensated. The second is meant to analyze the findings of the first to determine whether or not the work done in OSS is exploitative. In short, it is meant to determine whether the title of this report is accurate.

Next Steps

The first next step to my project is to research the stakeholders in this question and to specify exactly what is at stake. I hope to have this research done by the end of November. Next, I plan on consulting the framework listed above to evaluate the ethics of the system. I hope to have this done by the end of December. I need to postulate on potential solutions and the ramifications of those solutions. Lastly, I plan on writing my final report. I hope to have these two tasks completed by the end of January and February, respectively.

Sources

Giblin, Rebecca, and Cory Doctorow. "CHAPTER 17 Minimum Wages for Creative Work." *Chokepoint Capitalism: How to Beat Big Tech, Tame Big Content, and Get Artists Paid*, Beacon Press, Boston, MA, 2022.

Stallman, R. "The GNU Manifesto." [A GNU Head], 1985, https://www.gnu.org/gnu/manifesto.en.html.

Synopsis, Inc. "2022 Open Source Security and Risk Analysis Report." 2022, https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rep-ossra-2022.pdf

Van Elderen, M. "Synopsys Study Shows That Ninety-One Percent of Commercial Applications Contain Outdated or Abandoned Open Source Components." Synopsys Study Shows That Ninety-One Percent of Commercial Applications Contain Outdated or Abandoned

Open Source Components, 12 May 2020, https://www.prnewswire.com/news-releases/synopsys-study-shows-that-ninety-one-percent-of-commercial-applications-contain-outdated-or-abandoned-open-source-components-301057386.html.

von Krogh, G., et al. “Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development.” *MIS Quarterly*, vol. 36, no. 2, 2012, pp. 649–76. *JSTOR*, https://doi.org/10.2307/41703471. Accessed 9 Oct. 2022.