THE UNIVERSITY OF CHICAGO

BEHIND THE GLASS DOOR
UNRAVELING HIERARCHICAL STRUCTURE IN OPEN SOURCE SOFTWARE

A BACHELOR THESIS SUBMITTED TO

THE FACULTY OF THE DEPARTMENT OF ECONOMICS

FOR HONORS WITH THE DEGREE OF

BACHELOR OF THE ARTS IN ECONOMICS

BY CHRIS LIAO

CHICAGO, ILLINOIS
JUNE 2024

Contents

Abstract

[1]

[1]

Advisors: Ali Hortaçsu, Jesse Shapiro
Hierarchies/OSS: Luis Garicano, Josh Lerner, Frank Nagle
Econ Friends: Jordan Rosenthal-kay, Noah Sobel-Lewin
Research Supervisors: Scott Nelson, Thomas Sargent, George Hall, James Traina
Undergrad Econ Dept: Victor Lima, Kotaro Yoshida
Friends
Family
UChicago IT: Colin Hudler
OSS contributors: Anna Woodard + various survey respondents

1 Introduction

Paragraphs 1-2: Motivation. After reading these paragraphs a reader in any field of economics should believe that if you answer your research question your paper will make an important contribution.

The development of open source software (OSS), software that can be accessed, modified and redistributed without additional cost, is an $8.8 trillion industry (Hoffmann, Nagle, and Zhou 2024). OSS components are present in 70-90% of all software and have been adopted by businesses across many industries, such as tech, retail and auto (Nagle 2017). For example, this paper was produced using software such as LaTeX, Overleaf, Markdown, Python, all of which are open source. The distribution of OSS sans cost means that many of the organizations that develop OSS software projects are unlike traditional firms because OSS contributors, who help build the software, are volunteers. Consequently, the full time job of many OSS contributors is not developing OSS, so their contribution time to OSS is constrained. However, as volunteers, OSS contributors also have time freedom, as they, not the firm, determines how to allocate their time across different tasks. However, OSS organizations aren't totally unrecognizable when compared with traditional firms. OSS organizations have multiple layers of hierarchy and are just as affected, if not moreso, by technological advances, such as the advent of artificial intelligence or platform improvements by OSS hosting sites like GitHub.

The relationship between OSS organizational structure and OSS development, and the impact of technological advancement on that dynamic is not well understood. Developing an economic model that can help us better understand OSS development is crucial because of the importance of OSS in the economy. Using the lens of OSS organizational structure to study OSS development is important because OSS cannot be produced without the work of its contributors, whose tasks and roles are affected by OSS organizational structure. Moreover, understanding how technological advances affect the relationship between OSS organizational structure and OSS development is also important for two reasons. First, the digital and technology-focused nature of OSS development means that OSS contributors, and hence, the organization, are likely to be among the first affected by technological advances. Moreover, understanding the impact of technological advances can help policymakers and stakeholders make more informed decisions when the decision is OSS-relevant. The hierarchical nature of OSS organizations makes it tempting to apply models of hierarchy from the economics literature (Garicano 2000); however, those models are insufficient because they are

inspired by traditional firms with typical employment structures.

Paragraphs 3-4: Challenges. These paragraphs explain why your research question has not already been answered, i.e., what are the central challenges a researcher must tackle to answer this question.

The organizational structure of OSS development has been widely documented by researchers as a hierarchical process, but how that hierarchy affects OSS development have not been widely studied (Crowston and Howison 2006). Economists have modelled hierarchical structure in traditional firms, but modelling hierarchical structure in OSS development requires solving two challenges that existing models are not equipped for (Garicano 2000). First, the model must incorporate the constrained nature of an OSS contributor's contribution time and their time freedom. Existing models fail because they model traditional firms that determine what tasks their employees do and how much time they spend on the job. In OSS development, contributors choose what tasks to work on, and their total contribution time is their choice, not the OSS organization's choice (Lerner and Tirole 2002). Second, the model must describe the decision making process of OSS organizations, which differ from traditional firms. Traditional firms decide how to allocate employees across the hierarchy after observing employee wages and training costs, which they incur. OSS organizations also determine the allocation of employees across the hierarchy, but wages and training costs are not part of their decision making function as they incur none of those.

A model that solves both of these challenges will allow us to characterize the decisions that OSS contributors and organizations make. These changes are necessary because existing models of hierarchy produce puzzling results when applied to OSS organizations. One notable result from the hierarchy literature, applied to OSS development, is that OSS contributors ranked higher on the hierarchy should never be the ones writing code to solve new tasks.. It might be nice to explain what particular feature of my model enables this. This is not true in OSS software development, as the data shows plenty of highly ranked contributors solving newly encountered problems. Puzzle for comparative statics, if any

Paragraph 5: This Paper. This paragraph states in a nutshell what the paper accomplishes and how.

In this paper, I develop a model of hierarchical structure in OSS development, where OSS contributors have time freedom and OSS organizations determine the structure of the hierarchy without accounting for contributor associated costs. Solving for the equilibrium OSS contributors' and organization's decision functions is the key to characterizing how hierarchical structure affects OSS development. Using these equilib-

rium results, I follow the hierarchy literature in examining how technological advances - specifically, the cost of knowledge acquisition and the cost of communication - affects organizational structure and subsequently OSS development. Finally, I use novel microdata on OSS development from GitHub, the world's largest OSS hosting platform to compare my model's predictions to the empirical reality of OSS development.

Paragraphs 6-7: Model. Summarize the key formal assumptions you will maintain in your analysis.

The hierarchical model I propose features two layers of hierarchy. OSS contributors in both ranks are risk neutral and choose between spending time acquiring knowledge, or performing various tasks that help further develop the OSS. Following the hierarchy literature, OSS contributors within a rank are homogeneous (Garicano 2000). OSS contributor incentives differ by rank; while lower rank contributors are only concerned with finding a problem solution, higher rank contributors are prioritize the project's overall welfare. OSS organizations want to promote the optimal number of highly rank contributors to maximize output without incurring excessively high costs related to hiring and managing lots of highly ranked OSS contributors.

Paragraphs 8-9: Data. Explain where you obtain your data and how you measure the concepts that are central to your study.

Paragraphs 10-11: Methods. Explain how you take your model to the data and how you overcome the challenges you raised in paragraphs 3-4.

Paragraphs 12-13: Findings. Describe the key findings. Make sure they connect clearly to the motivation in paragraphs 1-2.

Paragraphs 14-15: Literature. Lay out the two main ways your paper contributes to the literature. Each paragraph should center around one contribution and should explain precisely how your paper differs from the most closely related recent work.

2 Model

2.1 Background

An OSS project's development can be abstracted to solving a variety of tasks; example tasks include fixing software bugs, implementing new features or writing documentation for the codebase. These tasks are solved by OSS contributors from the project, whose task-solving ability depends on the task difficulty $z \in [0, 1]$ and their own knowledge $k \in [0, 1]$, as tasks of difficulty z can only be solved by those with knowledge $k \geq z$.

There are two types of agents whose decision-making process we're interested in: OSS contributors, and the OSS project. OSS contributors are primarily characterized by their rank: read or write. Read rank contributors are below write rank contributors on the hierarchy. One real-world example of a read rank contributor is a OSS project user whose encountering some bugs related to that software. He needs the bug solved to accomplish his primary mission and he's not interested in contributing to the project beyond helping find a solution to the bug so he can continue using the software. They don't have any broader interest in the OSS project's welfare, such as helping develop a new feature, beyond their specific use case. On the other hand, a write-rank contributor is typically a longstanding contributor to the OSS project. She's knowledgeable about a large proportion of the project's scope and importantly, her rank also provides a signal to employers about her skill level. The more developed and well-known the project is, the more she benefits in terms of future wages, because it will increase her visibility to employers offering attractive positions (Hann et al. 2002).

All OSS contributors spend time t_p in production, the broad term I use for solving new tasks and acquiring knowledge k. An equivalent real-life example would be the time tradeoff someone makes between reading documentation about the project's different aspect, which helps them understand a larger proportion of the project's scope, and spending time actually trying to solve more problems. While the two choices are substitutes, being an effective contributor would be difficult without having spent time on both tasks as well. Read rank contributors develop solutions for all tasks that they encounter, but their limited knowledge $k^r < 1$ means they cannot assess whether their proposed solution is correct. As a consequence, write rank contributors also spent time t_h^w correcting incorrectly solved problems. Following the literature (Garicano 2000), I assume that write rank contributors try to correct all incorrect problems solutions. I denote distinguish between task type using variable letters or subscripts, and rank-specific variables using superscripts. Following Bloom et al. 2014, all write-rank contributors have perfect knowledge $k^w = 1$ which allows the project to solves all tasks it encounters. This is a reasonable assumption as I'm focused on studying how organizational structure affects software development, not how the level of programmer knowledge affects software development. Notwithstanding this, I'm pretty sure my results are invariant to k^w's level. I also follow the literature on hierarchies (Garicano 2000) by assuming that within a rank, contributors are homogeneous. Once I have my results I believe I'll be able to explain how heterogeneity would not change the qualitative nature of my results

Add note on why I omit approval

The OSS project's role is to allocate contributors across the hierarchy. I normalize the total number of OSS contributors to 1 and since there are only two ranks in the hierarchy, when the project promotes β_w contributors to write rank, there are $1 - \beta_w$ read rank OSS contributors. In my primary results, I assume that given the optimal number of contributors β_w^* the OSS project promotes to write rank, there are $\beta \geq \beta_w^*$ contributors who want to be promoted. This may not always be realistic, as some OSS contributors may not want to be promoted so $\beta < \beta_w^*$. Notwithstanding this, I show that this does not affect my results. The OSS project takes into account the decision function of OSS contributors and how their equilibrium decisions may be affected by the project's choice of β_w^* when choosing β_w^*. I assume that the OSS project is aware of the decision function of its contributors because each rank's specific incentives, as I described earlier, are well known and documented in the literature on incentives for contributing to OSS (Lerner and Tirole 2002, Lakhani and Hippel 2003, Krogh, Spaeth, and Lakhani 2003, Robert G. Wolf and Karim R. Lakhani 2003).

2.2 Set Up

2.2.1 Read Rank

Tasks of difficulty k appear with probability $f(k)$ and cumulative distribution function $F(k)$. In expectation, read-rank contributors with knowledge k^r that spend t_p^r time in production solve $t_p^r F(k^r)$ tasks correctly. Since all tasks solved incorrectly by read rank contributors are corrected by all write-rank contributors, in expectation, $t_p^r(1 - F(k^r))$ of each read-rank contributor's incorrectly solutions are fixed.

The read rank contributor's utility is affected by the proportion of all tasks they attempt that they solve correctly. While their primary goal is to obtain a solution, they also benefit from being the problem solver. In OSS development, contributors acquire skills from learning how to solve problems that have long-run benefits, and their ability to use problem solutions in their primary mission is enhanced when they provided and understand the solution. They face costs of $c_r(t_p^r, k^r)$ from contributing because it takes up their time. Thus, a read rank contributor solves

$$\max_{\{k^r, t_p^r\}} u_r\left(t_p^r F(k^r) + \omega_r(t_p^r(1 - F(k^r)))\right) - c_r(t_p^r, k^r)$$

$\omega_r < 1$ helps mediate the reduced benefit read rank contributors receive when their problems are solved by others. I make the following assumptions about u_r, c_r.

1. u_r is linear. Thus, $u_r(x) = \alpha_r x + \beta_r$ where $\alpha_r > 0, \beta_r \in \mathbb{R}$. Read rank program-

mers make their decisions knowing only the expectation of their outcomes. This complicates the process of solving the model without changing the qualitative results (I think intuitively this is true, but how do I show this? Maybe show examples for risk-averse, risk-favoring individuals), so I assume that the read rank contributor is risk neutral and possesses linear utility, which allows me to remove the expectation.

2. The first derivative of the cost function is increasing. Formally,

$$\frac{\partial c_r}{\partial t_p^r} > 0 \qquad \frac{\partial c_r}{\partial k^r} > 0$$

Intuitively, contributing to OSS costs time that could be spent on an OSS contributor's primary mission.

3. The second derivative and cross partials of the cost function are increasing. Formally,

$$\frac{\partial^2 c_r}{\partial (t_p^r)^2} > 0 \qquad \frac{\partial^2 c_r}{\partial (k^r)^2} > 0 \qquad \frac{\partial^2 c_r}{\partial t_p^r \partial k^r} > 0$$

An extreme but helpful motivating example is to compare the marginal cost of spending 10 minutes contributing to OSS, which is much higher when you have only contributed for 10 minutes, as opposed to when you have already contributed for 23 hours that day, and you really should grab an hour or two of sleep!

4. The marginal cost of the initial time spent contributing to OSS is negligible. Formally,

$$\lim_{k^r \to 0^+} \frac{\partial c_r}{\partial k^r} = 0 \qquad \lim_{t_p^r \to 0^+} \frac{\partial c_r}{\partial t_p^r} = 0$$

The assumption's practical effect is that the optimal choice of k^r, t_p^r will always be economically interesting so $t_p^r > 0, k^r > 0$. I make this assumption because this paper is focused on actual, not hypothetical OSS contributors.

2.2.2 Write Rank

The project's perceived success by outsiders is determined by its output. In aggregate, $1 - \beta_w$ read-rank contributors are expected to solve $(1 - \beta_w) t_p^r F(k^r)$ tasks correctly. Each write-rank contributor spends t_p^w time solving new tasks correctly with their perfect knowledge and t_h^w time helping correct read rank contributors solutions. Since all incorrect problem solutions are corrected, $\beta_w t_h^w = (1 - \beta_w) t_p^r F(k^r)$. Note that

t_h^w is decreasing in β_w, t_p^r and $F(k^r)$. In aggregate, β_w write rank contributors solve $\beta_w t_p^w + (1-\beta_w) t_p^r F(k^r)$ tasks and in total, the project solves

$$(1-\beta_w) t_p^r + \beta_w t_p^w$$

problems correctly.

Note that since their helping time t_h^w is fixed, I can omit it from the write rank contributor's problem, because t_h^w can be defined perfectly using β_w, t_p^r, k^r. Accordingly, the write ranked contributor faces costs $c_w(t_p^w, \beta_w, t_p^r, k^r, k^w = 1)$ from contributing, so they solve

$$\max_{\{t_p^w, t_h^w\}} u_w\left((1-\beta_w) t_p^r + \beta_w t_p^w\right) - c_w(t_p^w, \beta_w, t_p^r, k^r, k^w = 1)$$

I make the following assumptions about u_w, c_w

1. u_w is linear. Thus, $u_w(x) = \alpha_w x + \beta_w, \alpha_w > 0$

2. The first derivative of the cost function is increasing in. Formally,

$$\frac{\partial c_w}{\partial t_p^w} > 0$$

   Since t_h^w is decreasing in β_w, t_p^r and $F(k^r)$,

$$\frac{\partial c_w}{\partial \beta_w} < 0 \qquad \frac{\partial c_w}{\partial t_p^r} < 0 \qquad \frac{\partial c_w}{\partial k^r} < 0$$

   Change this...

3. The second derivative of production in the cost function are increasing. Formally,

$$\frac{\partial^2 c_w}{\partial (t_p^w)^2} > 0 \qquad \frac{\partial^2 c_w}{\partial (t_h^w)^2} > 0 \qquad \frac{\partial^2 c_w}{\partial t_h^w \partial t_p^w} > 0$$

4. The marginal cost of the initial time spent contributing to OSS is negligible. Formally,

$$\lim_{t_h^w \to 0^+} \frac{\partial c_w}{\partial t_h^w} = 0$$

2.2.3 Organization

The OSS organization's objectives are not as simple as maximizing its perceived output. While the OSS organization benefits from increased output, it encounters coordination problems from having too many write ranked programmers. For example, the costs of hiring, screening and coordinating with different write ranked contributors increases as the number of write ranked contributors β_w increase. I describe this cost as $c_o(\beta_w)$. Practically, this prevents the OSS organization from promoting everyone to write rank, which is what it would do absent c_o. This reflects the empirical reality of OSS organizations, which are largely composed of read rank contributors.
Aside: Another way to enforce this is to have write rank programmers be more disincentivized to contribute the higher β_w is. This is also grounded in empirical reality. I wonder how this affects the results though. It certainly makes characterizing β_w in the cost function more difficult for write rank programmers. Perhaps you could do cost invariant to β_w

Recall that the project's total output is

$$(1-\beta_w)t_p^r + \beta_w t_p^w$$

Thus, the OSS organization solves

$$\max_{\{\beta_w\}} u_o\left((1-\beta_w)t_p^r + \beta_w t_p^w\right) - c_o(\beta_w)$$

I make the following assumptions about u_o, c_o

1. u_o is linear. Thus, $u_o(x) = \alpha_o x + \beta_o, \alpha_o > 0$
2. The first derivative of the cost function is increasing. Formally,
$$\frac{\partial c_o}{\partial \beta_w} > 0$$
3. The second derivative of the cost function is increasing. Formally,
$$\frac{\partial^2 c_o}{\partial (\beta_w)^2} > 0$$
4. The marginal cost of coordinating with the first write rank programmer is negligible. Formally,
$$\lim_{\beta_w \to 0^+} \frac{\partial c_o}{\partial \beta_w} = 0$$

2.3 Solving the Model

2.3.1 Read Rank

Solving for the first order conditions tells us that read rank programmers find k^r, t_p^r solving

$$\frac{\omega_r + (1-\omega_r)F(k^r)}{t_p^r(1-\omega_r)f(k^r)} = \frac{\frac{\partial c_r}{\partial k^r}}{\frac{\partial c_r}{\partial t_p^r}}$$

2.3.2 Write Rank

If we assume that all problems that need help are solved,

$$\beta_w t_h^w = (1-\beta_w)t_p^r(1-F(k^r))$$

so I can rewrite the write rank contributor's problem as

$$\max_{\{t_p^w\}} u_w\left((1-\beta_w)t_p^r + \beta_w t_p^w\right) - c_w(t_p^w, \beta_w, t_p^r, k^r, k^w = 1)$$

so

$$t_p^w \text{ solves } \alpha_w\beta_w = \frac{\partial c_w}{\partial t_p^w} + \frac{\partial c_w}{\partial \beta_w}\frac{\partial \beta_w}{\partial t_p^w}$$

2.3.3 Organization Solution

When

$$t_h^w = (1-\beta_w)t_p^r(1-F(k^r))$$

$$\max_{\{\beta_w\}} u_o\left((1-\beta_w)t_p^r + \beta_w t_p^w\right) - c_o(\beta_w)$$

and

$$\beta_w \text{ solves } \alpha_o(t_p^w - t_p^r) = \frac{\partial c_o}{\partial \beta_w}$$

2.4 Brainstorming

2.5 Analysis

2.5.1 Contributor Time Allocation

Do write rank programmers code ($t_p^w > 0$)? Yes!

1. When $\frac{\beta_w t_h^w}{1-\beta_w} = 1 - F(k^r)$ then

$$\alpha_w \beta_w = \frac{\partial c_w}{\partial t_p^w}$$

   Since $\alpha_w > 0$ by definition and $\beta_w > 0$ is implied by $\lim_{\beta_w \to 0^+} \frac{\partial c_o}{\partial \beta_w} = 0$, then $\frac{\partial c_w}{\partial t_p^w} > 0$. Since $\frac{\partial^2 c_w}{\partial (t_p^w)^2} > 0$ and $\lim_{t_p^w \to 0^+} \frac{\partial c_w}{\partial t_p^w} = 0$, then $t_p^w > 0$

2. When $\frac{\beta_w t_h^w}{1-\beta_w} < 1 - F(k^r)$, $t_p^w = 0$ since $\frac{\partial c_w}{\partial t_h^w} < \frac{\partial c_w}{\partial t_p^w}, \forall t_h^w, t_p^w$

2.5.2 Comparative Statics

Need to think more about how to do this.

1. Figure out whether it is every optimal for the project to select a β_w that means write-ranked programmers program ($t_p^w > 0$

2. Figure out under what conditions β_w, t_h^w, k^r combine such that $t_p^w > 0$ or $t_p^w = 0$

3. Analyze what happens when $\frac{\partial c_r}{\partial k^r}$, $\frac{\partial c_w}{\partial t_h^w}$ (and perhaps $\frac{\partial c_w}{\partial k^w}$) changes

Initial approach is to separate into 4 bins. Also, want to understand when you fall into each bin.

1. Helping isn't maxxed out and post change, helping is still not maxxed out
   For example, suppose $\frac{\partial c_r}{\partial k^r}$ increases. That leads to a decrease in k^r and a decrease in $t_p^r F(k^r)$. Now we want to see whether an increase or decrease in β_w causes $t_h^w(\beta_w, k^r) + \beta_w \frac{\partial t_h^w(\beta_w, k^r)}{\partial \beta_w} - \frac{\frac{\partial c_o}{\partial \beta_w}}{\alpha_o}$ to decrease.
   Intuitively, we would think that β_w increases.

   (a) Increasing β_w actually causes t_h^w to increase because $\frac{\partial c_w}{\partial t_h^w}$ is concave.

   (b) It's not totally clear to me how $\beta_w \frac{\partial t_h^w(\beta_w, k^r)}{\partial \beta_w}$ is affected - β_w obviously increases and my instinct says $\frac{\partial t_h^w(\beta_w, k^r)}{\partial \beta_w}$ increases because as β_w increases, for the write rank contributor, there are increasing returns to helping more (since everyone helps more, and the multiplier effect of one person helping more increases).

   (c) $\frac{\frac{\partial c_o}{\partial \beta_o}}{\alpha_o}$ increases. But does it increase more than $t_h^w(\beta_w, k^r) + \beta_w \frac{\partial t_h^w(\beta_w, k^r)}{\partial \beta_w}$? I think I need functional forms to make this work.

2. Helping is already maxxed out and post change, helping is already maxxed out

3. Helping isn't maxxed out and post change, helping is maxxed out

4. Helping is maxxed out and post change, helping isn't maxxed out

Likely, for a given comparative static, only one of 3, 4 will be true. Analyzing 3, 4 will also be the most difficult part