0.1 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).

0.2 Set Up

0.2.1 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 programmers 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.

0.2.2 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, $\beta_w t_h^w = h(1-\beta_w)t_p^r(1-F(k^r))$. The $h>1$ represents communication costs encountered in helping correct problems. Note that t_h^w is decreasing in β_w, t_p^r and $F(k^r)$, and increasing in h. 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 define it perfectly as a function of β_w, h, t_p^r and k^r in the write rank contributor's problem. Accordingly, the write ranked contributor faces costs $c_w\left(t_p^w, t_h^w = \frac{h(1-\beta_w)t_p^r(1-F(k^r))}{\beta_w}, k^w = 1\right)$ from contributing, so they solve

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

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,

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

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 production in OSS is negligible. Formally,

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

0.2.3 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,
$$\frac{\partial^2 c_o}{\partial (\beta_w)^2} > 0 \tag{1}$$
4. The marginal cost of coordinating with the first write rank programmer is negligible. Formally,
$$\lim_{\beta_w \to 0^+} \frac{\partial c_o}{\partial \beta_w} = 0$$

0.3 Solving the Model

0.3.1 Read Rank

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

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

0.3.2 Write Rank

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

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

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

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

so

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

0.3.3 Organization Solution

The organization's problem is

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

so

$$\beta_w \text{ solves } \alpha_o(t_p^w - t_p^r) = \frac{\partial c_o}{\partial \beta_w} \tag{2}$$

0.4 Analysis

0.4.1 Contributor Time Allocation

The key question we're interested in is whether write rank programmers code ($t_p^w > 0$)? In Garicano 2000, we observe that the factory manager never picks up a hammer. While this aligns with our knowledge of factory production, in OSS development, highly ranked project members still write code and spearhead the production of new software features. Thus, we want a model that's able to reflect this empirical reality. Recall that

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

- $\alpha_w \beta_w > 0$
- $\frac{\partial c_w}{\partial \beta_w} < 0$
- By 2, $\frac{\partial c_o}{\partial \beta_w}$ is increasing in t_p^w. By 1, the concavity of the organization's cost function means that $\frac{\partial c_o}{\partial \beta_w}$ is also increasing in β_w. Thus, $\frac{\partial \beta_w}{\partial t_p^w} > 0$.

Since $\lim_{t_p^w \to 0^+} \frac{\partial c_w}{\partial t_p^w} = 0$, if $t_p^w = 0$, then $\alpha_w \beta_w < 0$. As $\frac{\partial^2 c_w}{\partial (t_p^w)^2} > 0$ then $\alpha_w \beta_w > 0$ requires $t_p^w > 0$.
Moreover, note that $\frac{\partial c_w}{\partial t_p^w} > 0$

0.4.2 Comparative Statics

Let's assume

- $F(k^r) = 1 - e^{-\lambda k^r}$, so $f(k^r) = \lambda e^{-\lambda k^r}$
- $c_r(k^r, t_p 6r) = (k^r)^R (t_p^r)^{P_r}$
- $c_w(t_p^w, t_h^w, k^w = 1) = (k^w)^W (t_p^w)^{P_w} (t_h^w)^H \iff (t_p^w)^{P_w} (\frac{h(1-\beta_w) t_p^r (1-F(k^r))}{\beta_w})^H$
- $c_o(\beta_w) = (\beta_w)^O$

What is the impact of $\frac{\partial c_r}{\partial k^r}, \frac{\partial c_w}{\partial k^w}$ or h on

1. t_p^r, k^r - read rank contributor time allocation response

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

   Thus,

   - $t_p^r (1-\omega_r) f(k^r)$ increases
   - $\omega_r + (1-\omega_r) F(k^r)$ decreases

2. $\frac{\beta_w}{1-\beta_w}$ - organization response - span of workers on each level
   This is equivalent to studying whether β_w increases.

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

- t_p^r increases, which should cause a decrease in β_w
- Recall that by
$$\alpha_w \beta_w = \frac{\partial c_w}{\partial t_p^w} + \frac{\partial c_w}{\partial t_h^w} \frac{\partial t_h^w}{\partial \beta_w} \frac{\partial \beta_w}{\partial t_p^w}$$
a decrease in β_w decreases t_p^w. So whether β_w should increase or not depends on how much t_p^w responds.

3. t_p^w - write rank contributor time allocation response
4. $t_h^w = h\frac{(1-\beta_w)}{\beta_w} t_p^r (1 - F(k^r))$ - write rank contributor time allocation response
5. $(1 - \beta_w) t_p^r F(k^r)$ - how much work is done by read rank contributors
6. $\beta_w t_p^w$ - how much work is done by write rank contributors
7. $\beta_w t_h^w \iff h(1 - \beta_w) t_p^r F(k^r)$ - how much helping is done