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 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. 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 her career prospects improve, 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 term I use for solving new tasks, and acquiring knowledge k. A real-life example would be the time tradeoff a contributor makes between reading documentation about the project's different aspects, which helps them understand a larger proportion of the project's scope, and spending time solving additional pieces of the encountered problem. 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 attempt to solve all tasks that they encounter using effort t_p^r, 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 helping correct incorrectly solved problems. Following the literature (Garicano 2000), I assume that write rank contributors 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. I also follow the literature and assume that within a rank, contributors are homogeneous 0(Garicano 2000).

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 $\beta < \beta_w^*$. 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).

In OSS development, write rank contributors also spend time approving problems. Approving problem solutions is important because it provides a signal to the general public that problem solutions are correct. In my model, I omit approval from the write rank contributor's choice set because adding it does not provide additional interesting insights about how organizational structure affects OSS development. Approving problem solutions is critical for project success, so it's a responsibility that will always have to be fulfilled by write rank contributors and has the straightforward effect of taking time away from production for write rank contributors.

0.2 Set Up

0.2.1 Read Rank

Tasks of difficulty k appear with probability $f(k)$ and associated CDF $F(k)$. 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 $c_r(t_p^r, k^r)$ because of the opportunity cost of time. Thus, a read rank contributor solves

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

$\omega_r < 1$ 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 $\alpha_r > 0, \beta_r \in \mathbb{R}$. Since read rank programmers make their decisions only knowing the expectation of their outcomes, practically, this assumption allows me to apply linearity of expectations and remove the expectation. This assumption is reasonable because the utility read rank contributors believe they will get for solving t_p^r problems with knowledge k^r is invariant to the dispersion in their quantity of problems solved created by dispersion in $F(k^r)$. I believe read rank contributors are invariant to dispersion because observing programming problem difficulty dispersion ex ante is also a tall order. This problem is further amplified by the fact that read-rank programmers, who are primarily users, not developers of the relevant software, may have limited domain-specific knowledge about implementing the relevant porblem solutions.

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 as their choice of $t_p^r > 0, k^r > 0$. 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 $\beta_w t_p^w + (1 - \beta_w)t_p^r(1 - F(k^r))$ 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, $u_w(x) = \alpha_w x + \beta_w, \alpha_w > 0$. As described in Jr 1995, even experienced software developers are unable to reliably estimate the time required to complete a project, even if they understand the conceptual difficulty of a problem. Thus, because write rank contributors do not observe problem difficulty dispersion, linear utility is a reasonable assumption.

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.
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, $u_o(x) = \alpha_o x + \beta_o, \alpha_o > 0$. Abstractly, we can think about the OSS organization's conception of problem difficulty dispersion as the ability of a few write ranked contributors (who also have additional managerial responsibilities) to perceive problem difficulty dispersion. These special write ranked contributors are responsible for promoting people. Jr 1995 states that the difficulty of estimating how long a project takes to complete typically comes from the manager's misestimation. From this, we can conclude that organizations

also cannot observe problem difficulty dispersion so linear utility is a reasonable assumption.

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,

$$\frac{\partial^2 c_o}{\partial (\beta_w)^2} > 0 \tag{2}$$

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

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

0.3.2 Write Rank

The write rank contributor's problem so

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

Note that $\frac{\partial u_w}{\partial \beta_w} = \alpha_w$ so

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

We also know that $\frac{\partial t_h^w}{\partial \beta_w} = -\frac{h t_p^r(1-F(k^r))}{(\beta_w)^2}$.

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

$$\beta_w \text{ solves } \alpha_o(t_p^w - t_p^r) = \frac{\partial c_o}{\partial \beta_w} \tag{3}$$

0.4 Analysis - Part 1

Suppose we want to analyze how an increase in $\frac{\partial c_r}{\partial k^r}$, the cost of knowledge acquisition for read-rank contributors, affects

1. How read rank contributors allocate their time
2. How write rank contributors allocate their time
3. The organization's response

0.4.1 How read rank contributors allocate their time

Let r^* denote choices prior to the cost increase and r denote choices after the cost increase. The concave nature of costs c_r and the linear form of u_r means that read-rank contributors will not increase output in solution set r^* and that $k^{r^*} > k^r$. Thus,

$$t_p^{r^*}(F(k^{r^*}) + \omega_r(1 - F(k^{r^*}))) > t_p^r(F(k^r) + \omega_r(1 - F(k^r)))$$

Suppose for contradiction that $t_p^{r^*}F(k^{r^*}) < t_p^r F(k^r)$. Note that since $k^{r^*} > k^r$, then $t_p^{r^*} < t_p^r$. Then

$$\begin{aligned} t_p^{r^*}F(k^{r^*})(1-\omega_r) &< t_p^r F(k^r)(1-\omega_r) \\ t_p^{r^*}F(k^{r^*})(1-\omega_r) + \omega_r t_p^{r^*} &< t_p^r F(k^r)(1-\omega_r) + \omega_r t_p^r \\ \iff t_p^{r^*}(F(k^{r^*}) + \omega_r(1 - F(k^{r^*}))) &< t_p^r(F(k^r) + \omega_r(1 - F(k^r)) \end{aligned}$$

This is clearly a contradiction. Thus, $t_p^{r^*}F(k^{r^*}) > t_p^r F(k^r)$. However, without specifying a functional form for c_r and the exact distribution of F, I won't know whether $t_p^{r^*} \gtreqless t_p^r$. However, it seems difficult to imagine that $t_p^{r^*} > t_p^r$. Consider the expression below, which is another way of expressing the read-rank contributor's FOCs

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

1. If t_p^r decreases, $t_p^r\frac{\partial c_r}{\partial t_p^r}$ decreases, which is the exact opposite of what we need for the conditions to equal when $\frac{\partial c_r}{\partial k^r}$ increases. This would require k^r to decrease even further.

2. Instead, consider the following. We can have t_p^r increase/stay the same and k^r decrease by less than it would have had to in the previous scenario in order to achieve equality. This would still be a possible optimal solution for the read-rank contributor because the condition holds with equality. Moreover, in this scenario, the equilibrium k^r, t_p^r are higher so this is clearly the preferred option.

0.4.2 How write rank contributors allocate their time & the organization's response

Finding t_p^w, β_w requires simultaneously solving

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

and

$$\beta_w \text{ solves } \alpha_o(t_p^w - t_p^r) = \frac{\partial c_o}{\partial \beta_w} \tag{4}$$

Here's why trying to figure out how t_p^w, β_w respond to an increase in $\frac{\partial c_r}{\partial k^r}$, the cost of knowledge acquisition for read-rank contributors, is complicated.

1. Suppose that t_p^r increases (what I consider the mostly likely scenario). Note also that t_p^w decreases, as k^r decreases. Consequently, $\alpha_o(t_p^w - t_p^r) < \frac{\partial c_o}{\partial \beta_w}$. To remedy this, one solution is to decrease β_w. The problem is that when β_w decreases, t_p^w is affected. Write-rank contributors now have more problems they need to help correct, so t_h^w increases and t_p^w decreases even more. But if the decrease in β_w causes a decrease in $\frac{\partial c_o}{\partial \beta_w}$ that outweighs the decrease in $\alpha_o(t_p^w - t_p^r)$, this could be a possible solution.
   Economically, what's happening is that write rank programmers become less valuable to the organization because they're engaged in less production (and

read rank programmers are attempting more problems, so the model conceptualizes them as "more" valuable). Consequently, it may be optimal for the project to have less write rank programmers because they contribute less to production, and it can improve it's overall welfare by reducing coordination costs.

Note also that any change in β_w must also still maintain that $t_p^w > t_p^r$ as $\frac{\partial c_o}{\partial \beta_w} > 0$ always.

2. But does the decrease in $t_p^w - t_p^r$ exceed the decrease in $\frac{\partial c_o}{\partial \beta_w}$? What if the solution is to increase β_w? Even though $\frac{\partial c_o}{\partial \beta_w}$ increases, the corresponding increase in t_p^w might be able to account for both that increase, the increase in t_p^r. Note that it's not totally clear the increase in β_w will always increase t_p^w, but in this situation it must happen.
   Economically, if the number of write rank contributors β_w increases, the amount of helping each contributor needs to perform decreases so they might be able to spend more time on production. In this scenario, the increased production gained means that this new increased level of β_w is actually worth it, if the improved level of production outweighs the cost of an increase in β_w.

0.5 Analysis - Part 2

Suppose we want to analyze how an increase in h, the cost of communication, affects

1. How read rank contributors allocate their time
2. How write rank contributors allocate their time
3. The organization's response

0.5.1 How read rank contributors allocate their time

They are not affected.

0.5.2 How write rank contributors allocate their time & the organization's response

Recall that $t_h^w = \frac{h(1-\beta_w)t_p^r(1-F(k^r))}{\beta_w}$, so $\frac{\partial t_h^w}{\partial \beta_w} = -\frac{ht_p^r(1-F(k^r))}{\beta_w^2}$. Thus,

$$t_p^w \text{ solves } \alpha_w\beta_w + \frac{ht_p^r(1-F(k^r))}{\beta_w^2}\frac{\partial c_w}{\partial t_h^w}\frac{\partial \beta_w}{\partial t_p^w} = \frac{\partial c_w}{\partial t_p^w}$$

When h increases, then there are also two possible options. Recall that β_w and t_p^w move together.

1. If h increases, one potential counterbalancing solution is for t_p^w to also increase, to equalize the LHS and RHS. However, whether this works also hinges on whether β_w responds by increasing or decreasing. Note that regardless of whether β_w increases or decreases, the change in $\alpha_w\beta_w + \frac{ht_p^r(1-F(k^r))}{\beta_w^2}\frac{\partial c_w}{\partial t_h^w}\frac{\partial \beta_w}{\partial t_p^w}$ is also of unclear sign.

2. If h increases, another solution is for β_w to decrease because the decrease in $\alpha_w\beta_w$ may outweigh the unclear change in magnitude of $\frac{ht_p^r(1-F(k^r))}{\beta_w^2}\frac{\partial c_w}{\partial t_h^w}\frac{\partial \beta_w}{\partial t_p^w}$. The decrease in β_w will decrease t_p^w for sure, more helping is required from two fronts: increased communication cost time and less write rank contributors to split responsibilities among.

0.6 Note for Jesse

At the risk of the previous sections being unclear, here's a summary of the dilemma I'm encountering. For most of the comparative statics I'm interested in, the answer I've arrived at so far is that the sign of the change is **unclear**. I haven't substituted functional forms in this note because when I've tried to do so in the past, the calculations haven't provided any useful insights and it tends to lead to a Mathematica rabbit hole. I'm debating between two different next steps

1. Trying to find one reasonable example (using functional forms) where it's possible that for example, depending on the specifics of the utility & cost function, the sign change for each of my variables of interest could be either positive or negative to formally prove that

2. Since the change in sign seems at this point like an empirical question, I would pivot towards analyzing the data I have. Personally, I would prefer this because I'm a bit tired of thinking about this model, the marginal gains of working on

the empirical analysis are larger and I enjoy data analysis more. I also know that it's important to make sure I do a job done well and the model will offer little use if it's not done well and doesn't help us understand how different forces are causing change.