## 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 $\beta_w$ 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 $\beta_w^*$ 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 $\beta_w^*$ when choosing $\beta_w^*$. 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 $\beta_w, t_p^r$ and $F(k^r)$, and increasing in $h$. In aggregate, $\beta_w$ 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 $\beta_w, h, t_p^r$ 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˙mythical˙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 $\beta_w$ 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˙mythical˙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

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

### 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 $\frac{\partial t_h^w}{\partial \beta_w} = -\frac{ht_p^r(1-F(k^r))}{(\beta_w)^2}$.

### 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

### 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

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

First, we know the LHS $\alpha_w \beta_w > 0$. We also know that $\frac{\partial t_h^w}{\partial \beta_w} < 0$ and $\frac{\partial c_w}{\partial t_h^w} > 0$ so $\frac{\partial t_h^w}{\partial \beta_w}\frac{\partial c_w}{\partial t_h^w} < 0$. Finally,

- $\alpha_w \beta_w > 0$
- $\frac{\partial t_h^w}{\partial \beta_w} < 0$ and $\frac{\partial c_w}{\partial t_h^w} > 0$ so $\frac{\partial t_h^w}{\partial \beta_w}\frac{\partial c_w}{\partial t_h^w} < 0$
- By 3, $\frac{\partial c_o}{\partial \beta_w}$ is increasing in $t_p^w$. By 2, the concavity of the organization's cost function means that $\frac{\partial c_o}{\partial \beta_w}$ is also increasing in $\beta_w$. 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 $t_p^w > 0$.

### 0.4.2 Comparative Statics

What is the impact of an increase in $\frac{\partial c_r}{\partial k^r}, \frac{\partial c_w}{\partial k^w}$ (cost of knowledge for read or write rank contributors) or $h$ (communication costs) on

1. $t_p^r, k^r$ - How do read rank contributors change their time allocation?

2. The next three statistics of interest are all interrelated, so the best approach is to analyze them simultaneously. We're interested in

   (a) $\frac{\beta_w}{1-\beta_w}$ - organization response - span of workers on each level

   (b) $t_p^w$ - write rank contributor time allocation response

   (c) $t_h^w = h\frac{(1-\beta_w)}{\beta_w}t_p^r(1-F(k^r))$ - write rank contributor time allocation response

   The statistics in the previous two points combine to inform us about how contributors of all ranks and organizations change their time allocation in response to technological changes. These are valuable because they tell us how OSS contributors might adapt their workflows when new technologies are adopted. They also tell us how organizations restructure themselves

3. $(1-\beta_w)t_p^r F(k^r)$ - how much work is done by read rank contributors in aggregate.

4. $\beta_w t_p^w$ - how much work is done by write rank contributors, in aggregate. If I find both of the statistics above, I can also analyze how overall work share is changing and how that's evolving with the quantity of people on each level of the hierarchy, $\beta_w$

5. $\beta_w t_h^w \iff h(1-\beta_w)t_p^r F(k^r)$ - how much helping is done
   The statistics in the aforementioned three points help us understand how technological advances affect the quantity and scope of responsibilities contributors at different levels of the hierarchy have to take on in response to technological change.

## 0.5 Robustness

1. Show that my results are invariant to setting $k^w = 1$ as long as $k^w > k^r$ fairly important
2. Once I have my results, I can explain how heterogeneity within rank would impact my results
3. Show how the results change when $\beta_w$ is capped.
4. Impact of adding approval?
5. How can I affect $\beta_w$ by discincentivizing write rank contributors when $\beta_w$ is too high
6. How can I justify that write rank programmers and organizations are invariant to inputs with mean preserving spreads in $u_w, u_o$? Not sure it's very true...