0.1 Background

Developing a OSS project involes solving a variety of tasks, such as fixing software bugs, implementing new features or writing documentation for the codebase. OSS contributors affiliated with the project are responsible for solving these tasks. Each task has difficulty $z \in [0, 1]$ and each contributor (of rank $\tau \in \{r, w\}$) has knowledge k^τ; contributors with knowledge k^τ can only solve tasks with difficulty $z \leq k^\tau$.

My model explains how OSS contributors and the OSS project make decisions. OSS contributors are ranked as either read (r) or write (w); write rank is above read rank in the hierarchy. A real-world example of a read rank contributor is an OSS project user encountering a bug. They need the bug resolved to continue their work but have no interest in contributing to the project beyond finding a solution. In contrast, write-rank contributors are typically closely involved with the project and knowledgeable about its purpose and goals. Write-rank contributors want the project, in the aggregate, to grow because the more well-known the project, the higher her visibility to employers offering attractive positions and the better her career prospects improve (Hann et al. 2002).

All OSS contributors (of rank τ) spend time t_p^τ solving new tasks (henceforth referred to as production) and acquiring knowledge k. Read rank contributors attempt to solve tasks with effort t_p^r, but their limited knowledge $k^r < 1$ means they cannot assess the correctness of their proposed solutions. Thus, write rank contributors also spent time t_h^w helping correct incorrect task solutions. I follow the literature and assume that write rank contributors have perfect knowledge $k^w = 1$[1]. (Bloom et al. 2014) so they successfully correct all incorrect problem solutions and that contributors have identical characteristics (and thus make identical decisions) conditional on rank (Garicano 2000).

The OSS project's role is to determine the allocation of 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 contributors remaining[2]. The OSS project understands the decision-making process of contributors at each rank because the incentives for contributing at each level are well-documented in the literature (Lerner and Tirole 2002, Lakhani and Hippel 2003, Krogh, Spaeth, and Lakhani 2003, Robert G. Wolf

[1] Justify why this doesn't make a difference, I think Garicano 2000 has something on this

[2] 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 $\beta < \beta_w^*$, but in these cases. What are the implications?

and Karim R. Lakhani 2003). This means the OSS project incorporates the time allocation decisions of contributors into its choice of β_w.[3]

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. Contextualizing this choice, read rank contributors face time tradeoff between reading documentation to understand the project and spending time solving specific issues. Both tasks compete for the contributor's time, but effective contribution requires a balance between the two.

The read rank contributor's utility is determined by how much effort they apply and how much knowledgde they acquire.
Moreover, contributors benefit in their on-the-job education from learning how to solve problems that have long-run benefits and their ability to use a problem solution is enhanced when they 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

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

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, so $u_r = \mathbb{E}[u_r]$.

   Linear utility requires contributors who are invariant between mean preserving spreads. In my setting, this means that the utility read rank contributors believe they will get for solving t_p^r problems with knowledge k^r is invariant to potential

[3] 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. decide whether to include this as a footnote or in an appendix section

dispersion created by mean-preserving changes in the problem difficulty distribution $F(k^r)$. I actually can't explain this... They obviously behave differently with different $F(k^r)$...

One idea is to use the normal approximation to the binomial distribution (perhaps with truncation)

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

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

for all t_p^r, k^r. 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,

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

for all t_p^r, k^r. 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 production t_p^r and knowledge k^r will exceed zero. I make this assumption because this paper is focused on actual, not hypothetical OSS contributors.

0.2.2 Write Rank

I define a project's success by its overall output; users are more likely to use OSS projects that resolve all of their software bugs and implement all requested features. There are $1 - \beta_w$ read-rank contributors in total who solve $(1 - \beta_w) t_p^r F(k^r)$ tasks correctly. Each write-rank contributor spends t_h^w time correcting incorrect problem

solutions and the remaining t_p^w time in production. 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 [4]. Note that t_h^w is decreasing in β_w, t_p^r and $F(k^r)$, and increasing in h. Thus, β_w write rank contributors solve $\beta_w t_p^w$ tasks in production and $(1-\beta_w)t_p^r(1-F(k^r))$ tasks through helping and in total, the project solves $(1-\beta_w)t_p^r + \beta_w t_p^w$ tasks.

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 solves

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

I make the following assumptions about u_w, c_w

1. u_w is linear. Thus, $u_w(x) = \alpha_w x + \beta_w, \alpha_w > 0$. also problematic, same reason as before

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

Like traditional firms in the theory on knowledge hierarchies, the OSS organization wants to maximize output, subject to costs. Although the organization does not pay

[4] what if I make h increasing in β_w

wages, it does incur increasing coordination costs as the number of write ranked contributors β_w increase. Absent this cost $c_o(\beta_w)$, the organization would likely promote everyone to write rank. 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, $u_o(x) = \alpha_o x + \beta_o, \alpha_o > 0$. also problematic, same reason as before

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

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

When a read rank contributor's knowledge increases, they solve $t_p^r f(k^r)$ more problems, which has $\omega_r t_p^r f(k^r)$ benefit However, note that the utility they derive from write rank contributors solving their problems decreases, because when read rank contributors are more knowledgeable, write rank contributors solve $t_p^r f(k^r)$ less problems. The combination of both factors mean that ultimately, the change in utility caused by an

increase in knowledge on overall problems solved is weighted by a factor of $2\omega_r - 1$. By imposing that $\omega_r > \frac{1}{2}$, which says that contributors prefer to solve problems on their own than have them be solved by others, read rank contributors will acquire some knowledge.

When a write rank contributor's effort increases, they benefit in two ways. First, they solve $F(k^r)$ more problems, which has $\omega_r F(k^r)$ benefit. Second, because more problems are unsolved, write rank contributors have to help fix $1 - F(k^r)$ problems, which has $1 - \omega_r$ benefit.

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

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

Total production increases by β_w when production effort t_p^w is increased, with an aggregate effect of $\alpha_w \beta_w$. The increase in t_p^w also has a secondary effect, which depends on whether increases in production t_p^w increase or decrease the total amount of write rank contributors β_w (aka what's the sign of $\frac{\partial \beta_w}{\partial t_p^w}$).

Typically, we expect that at the size of most OSS organizations (which is typically small), when write rank contributors are engaging in more production t_p^w, they become more valuable (relatively) to the project, so the project wants to promote more people to write rank, especially because the cost of coordinating with them is likely to be small. However, if the organization already has many write rank contributors, when write rank contributors engage in more production, the project might actually find that they do not need that many write rank contributors, because each individual one is more capable, and the project can benefit a lot by reducing excess staff. Here, I've posited that economically, in small OSS projects, $\frac{\partial \beta_w}{\partial t_p^w} > 0$ while in larger OSS projects, $\frac{\partial \beta_w}{\partial t_p^w} < 0$. When $\frac{\partial \beta_w}{\partial t_p^w} > 0$ the second term on the left is positive, as write rank contributors can spend less time helping correct problems (and more time in production) because there are more contributors to correct problems.

Note that throughout this analysis of write rank contributors, I have been considering

how the organization benefits. I can do so because the write rank contributor and the organization have aligned incentives. The key difference is the cost they face - as humans, write rank contributors face concave time costs from engaging in production ($\frac{\partial c_w}{\partial t_p^w}$), so the level of production they choose to engage in is the level at which the marginal prospective benefit to the project (them) equals the marginal cost.

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

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

There are two benefits to increasing write-rank contributors, β_w. First, exchanging a read rank contributor for a write rank contributor has the net effect of $t_p^w - t_p^r$ on production. Second, note that when more write rank contributors are promoted, write rank contributors' individual production increases because they have less helping responsibilities. This is captured by $\frac{\partial t_p^w}{\partial \beta_w}$, and its multiplied by a factor of β_w for each write rank contributor.

0.4 Analysis - Part 1

Suppose we want to analyze how a decrease 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

$$\frac{(2\omega_r - 1)t_p^r f(k^r)}{1 - \omega_r + (2\omega_r - 1)F(k^r)} = \frac{\frac{\partial c_r}{\partial k^r}}{\frac{\partial c_r}{\partial t_p^r}}$$

1. Obviously, to some degree, k^r decreases. The marginal benefit of knowledge acquisition, at the previous level of k^r, is now outweighed by the marginal cost

of knowledge acquisition. The direct effect of the decrease is of less interest to us. What's more interesting is how the project responds to this change in their capabilities, and how that affects the work read-rank contributors do.

2. The changes in k^r and t_p^r are interrelated. Since $\frac{\partial c_r}{\partial k^r}$ increased, the marginal cost of acquiring knowledge now exceeds the marginal benefit of knowledge, so the amount of knowledge acquired will be reduced.

   How t_p^r changes depends. It might be the case that because the marginal benefit of production effort is affected by a contributor's level of knowledge it may be optimal to reduce production effort. It might also be the case that increasing production may be optimal. The optimal ratio of knowledge to effort decreases when the cost of knowledge increases; this can be accomplished either by decreasing knowledge, increasing production effort, or both. The combination of both that is chosen depends on the cost curves of knowledge and production. When reducing time spent on knowledge acquisition reduces costs by large (small) magnitudes, has (small) negative effects on utility, and increasing production has large (small) increases on cost, then contributors may choose to decrease the knowledge to effort ratio large through reductions (increases) in knowledge acquisition. Should I describe examples of when one or the other occurs? IE: high cost of knowledge acquisition, low ω_r and low knowledge (so not worth it to acquire more knowledge) vs. low cost of effort, high knowledge and high ω_r. It seems pretty unlikely that cost of knowledge acquisition is high when little knowledge has been acquired. Intuitively, it seems likely that t_p^r increases The reality makes sense - t_p^r increases. People spend less time acquiring knowledge by reading documentary if the documentation is messy and full of errors - instead, they invest that effort towards trying to solve the actual problem at hand.

   We also know that regardless of the intricacies of the read rank contributor's cost and utility functions, $t_p^r F(k^r)$ cannot increase after an increase in $\frac{\partial c_r}{\partial k_r}$. Note also that if k^r decreases, it doesn't make sense for t_p^r to decrease - the read rank contributor wants to maintain the highest $t_p^r F(k^r)$ value it can while reaching the new optimal ratio of k^r to t_p^r.

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

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

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 production effort by read rank contributors t_p^r increases, which is the most likely scenario. Let's first consider how the project responds, because the project's choice of β_w determines the write rank contributor's choice of t_p^w. The project faces the following balancing act - write rank contributors are valuable because they help correct problem solutions and engage in production. On the other hand, they're costly for the project. Read rank contributors engage in production without incurring extra costs for the project.
   When production by read rank contributors increases, they become more valuable, relative to write rank contributors, so the marginal cost of a write rank contributor β_w now exceeds the marginal benefit of the additional write rank contributor. To resolve this, the project has two options, both of which are situation dependent: increase or reduce the number of write rank contributors β_w.

If write rank contributors are very costly for the organization (high $\frac{\partial c_o}{\partial \beta_w}$), especially relative to the benefit of additional write rank contributors, β_w may be reduced. Reducing β_w reduces the organization's costs, on the RHS. It also reduces write rank contributor production (because write rank contributors now have helping responsibilities), which reduces the RHS.
In the context of open source, it seems implausible that $\frac{\partial c_o}{\partial \beta_w}$ is high, given than open source projects are always suffering from a lack of maintainers. If β_w increases, while $\frac{\partial c_o}{\partial \beta_w}$ increases, it will not be large given that OSS projects always benefit from more contributors. Moreover, increasing β_w increases write rank contributor production t_p^w because it enables them to spend less time helping and more time on production.

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 $\frac{\partial t_h^w}{\partial \beta_w} = -\frac{ht_p^r(1-F(k^r))}{\beta_w^2}$. When communication costs increase, write rank contributors spend more effort correcting problems and engage in less production. This is a similar situation to earlier (increases in $\frac{\partial c_r}{\partial k6r}$), with the difference that instead of a t_p^r increase you have a t_p^w increase. Similar to earlier, I believe that β_w will increase in response.