2014

SOCIAL PREFERENCES AND OPEN SOURCE SOFTWARE DEVELOPMENT

Xianjun Geng  
*University of Texas at Dallas*, geng@utdallas.edu

Lihui Lin  
*Tsinghua University*, lin.lihui@gmail.com

Gang Peng  
*Youngstown State University*, gpeng@ysu.edu

Follow this and additional works at: [http://aisel.aisnet.org/pacis2014](http://aisel.aisnet.org/pacis2014)

Recommended Citation

[http://aisel.aisnet.org/pacis2014/6](http://aisel.aisnet.org/pacis2014/6)

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2014 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
SOCIAL PREFERENCES AND OPEN SOURCE SOFTWARE DEVELOPMENT

Xianjun Geng, School of Management, University of Texas at Dallas, Dallas, Texas, USA, geng@utdallas.edu

Lihui Lin, School of Economics and Management, Tsinghua University, Beijing, China, lin.lihui@gmail.com

Gang Peng, Williamson College of Business Administration, Youngstown State University, Youngstown, OH, USA, gpeng@ysu.edu

Abstract

Open source software (OSS), and open innovation in general, has received increasing attention from both researchers and practitioners. Based on recent literature on social preference from behavior economics, we propose a finite-horizon dynamic model to study the interactions between OSS developers who are either purely self-interested or conditional cooperators. We find that self-interested developers who are predicted to free ride under conventional analysis may contribute to a public good, and the existence of purely these developers may, under certain conditions, even benefit the provision of a public good. We further analyze how code architecture affects OSS development outcome and propose that a higher level of code modularity leads to more code contributions overall, due to the strategic behavior of self-interested developers. However, a right mix of the two types of developers plays a critical role for modular design to make an impact. The findings bear important theoretical as well as practical implications and provide guidelines for OSS development and the collective innovation in general.

Keywords: Behaviour economics, Game theory, Open source software, Social preferences


1 INTRODUCTION

Over the past two decades, the phenomenon of open source software (OSS) has attracted much attention from both academics and practitioners (see, among others, Feller et al. 2005; von Krogh et al. 2012). With the remarkable success of such OSS as Linux, Apache, MySQL, Perl, PHP, and Mozilla, open source software has become an integral part of corporate information systems (IS) infrastructure, making OSS a subject of high interest to IS managers at all levels.

One of the fundamental questions in OSS research, as articulated by Lerner and Tirole (2002), is: “Why should thousands of top-notch programmers contribute freely to the provision of a public good?” Economic research shows that purely self-interested individuals inevitably free ride in the provision of a public good when their marginal cost of contribution exceeds their marginal benefit from the public good. This implies that developers contribute to OSS because they either gain high private benefits in excess of their cost (e.g., Johnson 2002, Baldwin & Clark 2006) or have social preferences other than self-interest (e.g., Fischbacher & Gachter 2010). Lerner and Tirole’s question has spawned a plethora of research on the motivations of OSS developers, most of which recognizing the distinction between intrinsic and extrinsic motivations (see, among others, Hars & Ou 2002; Lakhani & Wolf 2005; Lerner & Tirole 2002; Roberts et al. 2006; Wu et al. 2007; and see von Krogh et al. 2012 for a comprehensive survey). The intrinsic and extrinsic motivations identified in the literature can be considered either social preferences or private benefits for the contributing developer in excess of the value of the public good. For example, the enjoyment of coding (an intrinsic motivation), reputation (internalized extrinsic motivation), and pay (extrinsic motivation) provide high private value for developers. Empirical evidence also shows that social preferences such as altruism (an intrinsic motivation) and reciprocity (an internalized extrinsic motivation) are important motives for OSS developers (von Krogh et al. 2012).

While OSS research has considerably advanced our understanding of the motivations of developers, one may further inquire: Are private value and social preferences the only reasons why developers contribute? In other words, is it possible for people without these motivations to contribute to OSS? Prior research also shows that OSS participants have varied motivations, which leads to related questions: how do developers with different motivations influence each other? And how do the dynamics between these developers affect the outcome of OSS projects? We attempt to address these questions in this paper.

We develop multiple-period, game-theoretic models to study the behavior and group dynamics of OSS developers with different motivations. In particular, we identify three types of developers: 1) developers with no additional private value from OSS other than using the outcome, whom we call strategic developers; 2) developers with a social preference to cooperate provided others cooperate as well, whom we call conditional cooperators following related literature (e.g., Fischbacher et al. 2001; Gachter 2007); and 3) developers who gain private value other than utilizing the software (e.g., from enjoyment of coding), whom we call hackers. Such groupings are consistent with the empirical literature. In their research of OSS developers from the SourceForge.net database, Lakhani and Wolf (2005) find that while each developer is often motivated by multiple factors, four clusters of developers by motivation type emerge: Cluster 1 (25%) who are predominantly motivated by work-related need and Cluster 2 (27%) by nonwork-related need; Cluster 3 (29%) who contribute mainly for the intellectual stimulation and to improve their skills; and Cluster 4 (19%) who mostly feel obliged to contribute to the OSS community in return for the software tools it provides. Clusters 1 and 2 can be considered as one group driven by their need to use the software, which resemble the “strategic developers” in our model. The hacker type in our model is a stylization of Cluster 3, while Cluster 4 has similar motivations as the conditional cooperators in our analysis.

1 Open source software is generally considered to be public goods since the software can be distributed and further incorporated freely into any other software through open source licenses (von Hippel & von Krogh 2003; Baldwin & Clark 2006).
In this paper, we study the behaviour and dynamic interaction between the first two types of developers. This means the participant pool consists of strategic developers and conditional cooperators only, the case least amenable to the development of OSS. One of the most interesting and surprising results of our analysis is that strategic developers may contribute to OSS as long as there are conditional cooperators in the participant pool. We further find that the existence of strategic developers may, under certain conditions, even benefit an OSS project. In future research, we plan to extend our model to include the third type: hackers.

Within the framework of our models, we further consider an important feature of software development process: modularity, and examine its impacts on OSS development outcome. In our setting, modular design simply means participants working in smaller teams and does not lead to other efficiency gains. We find that a higher level of modularity leads to more contributions to a project overall, due to the behavior of strategic developers, who now interact with a smaller number of conditional cooperators working on the same module. A key distinct between this paper and prior research is that we emphasize the role of OSS developers without private or social preferences for the development of a public good. Researchers have been seeking motivations for participating in OSS other than the utility from OSS as a public good. In contrast, we acknowledge that some developers have the additional motivations identified in the literature, but we focus on the behavior and impact of developers whose sole motivation is the use of OSS. Such developers are predicted not to contribute by conventional analysis, but the result in Lakhani and Wolf (2005) clearly shows that more than 50% of the contributing developers are primarily motivated by own use and are not strongly motivated by other factors (see Table 1). Our model allows us to explain this puzzling phenomenon.

Our research also contributes to the OSS literature by analytically modeling the behavior of conditional cooperators in OSS development. While reciprocity has been widely recognized and empirically proved to be an important motivation for OSS developer, there is no analytical model about such behavior. Based on the experimental studies of the provision of public goods (Fischbacher & Gächter 2010), we quantify the beliefs and behavior of conditional cooperators and further examine their impact on the dynamics and outcome of OSS development. Our findings shed light on the role of developers mostly motivated by community obligation and reciprocity.

Our results also provide a new mechanism through which modularity may benefit software development. We also show that the composition of teams matters in OSS development. These findings have interesting implications for OSS development.

Reviewing the OSS literature of the last twenty years decades, Von Krogh et al. (2012) point out that the extant OSS literature does not provide satisfactory answers to three key questions about the existence of OSS, one of which being “why developers sustain OSS development?” Our research makes progress in this direction.

2 LITERATURE REVIEW

In this section, we discuss how our research is related to the literature. We draw upon several strands of literature: 1) research on the motivation of OSS developers; 2) recent research on the economics of public goods; and 3) research on modularity in the context of software development.

2.1 OSS research

Central to OSS research is the puzzle why programmers are willing to participate in the development of OSS. In the studies of motivations of OSS developers, the framework provided by the self-determination theory (SDT) (Deci & Ryan 1987; Gagne & Deci 2005) has been used widely, which makes the distinction between intrinsic and extrinsic motivations. Von Krogh et al. (2012) group motivations to contribute to OSS can be grouped into intrinsic, internalized extrinsic, and extrinsic motivations. Ideology and fun are examples of intrinsic motivations while careers and pay are pure extrinsic motivations. Some extrinsic motivations, however, can be internalized by individuals, and thus become self-regulated instead of externally imposed (Roberts et al. 2006). Such internalized
motivations include reciprocity, learning, reputation, and own-use value (von Krogh et al. 2012). It must be noted that each developer tends to be motivated by multiple factors.

This literature also suggests the existence of conditional cooperators in OSS development. OSS developers have reported that they like help others and give something back to others (Osterloh & Rota 2007), and they reciprocate help and favor received previously (Lakhani & von Hippel 2003). Therefore, OSS development environment exhibits certain levels of “gift culture” (Wu et al. 2007). Surveys to OSS developers also suggest that helping each other and returning the favor is one of the major motivations for code contribution (Hars & Ou 2002). The existence of conditional cooperators in OSS developers echoes the recent findings from behavior economics that the conditional cooperation in public good is a widely observed phenomena (Gachter, 2007; Fischbacher & Gachter, 2010; Rustagi et al. 2010).

While researchers have categorized the motives and further examine the interaction among motivation factors (e.g., Osterloh & Rota 2007; Roberts et al. 2006), how developers with different motives interact with each other during the course of a project has remained largely unexplored.

### 2.2 Economics of Public Goods

Research on public goods has shown that people may contribute to public goods more than levels predicted under the assumption of pure self-interest, and has proposed various social preferences including altruism and warm glow as candidates that may explain such behavior (e.g., Fischbacher & Gachter 2010; Klumpp 2012; Rustagi et al. 2010). One type of social preferences that has been well documented is called “conditional cooperation,” a propensity to cooperate provided others cooperate as well (see Gachter 2007 for a review of this literature). Such “conditional cooperation” is able to explain public goods contributions beyond self-interest, as well as the fragility of cooperation.

Recent research from behavior economics on public goods has shown that free riding usually increases in repeated settings across various parameters and participant pools (e.g., Fischbacher & Gachter 2010).

The social preference of conditional cooperation has long been argued by social psychologists, observed in public goods field studies and experiments, and widely accepted in academic research (e.g., Sugden 1984; Guttman 1986; Keser & van Winden 2000; Fischbacher et al. 2001; Frey & Meier 2004; Croson et al. 2005; Croson 2007; Gachter 2007; Croson & Shang 2008; Fischbacher & Gachter 2010). This literature proposes that individuals not only care about their own material payoffs, but also derive non-material payoffs so they engage in sacrificing behaviors. One explanation for this observation is that individual preferences are to maximize something other than their benefits. For example, individuals may care about the well-being of others (altruism) (Andreoni 1995), have a preference for giving (warm glow) (Andreoni 1990), value fairness (Rabin 1993), and prefer reciprocity and cooperation (e.g., Sugden 1984; Guttman 1986; Andreoni 1995; Gachter 2007; Croson & Shang 2008; Fischbacher & Gachter 2010). Such other-regarding preferences than pure self-interest are called social preferences (Klumpp 2012).

Research shows that conditional cooperators form beliefs about the contributions by others, and make contribution decisions based on their belief. Fischbacher and Gachter (2010) find that the belief of a conditional cooperator is a weighted average of his/her previous period belief and the observed average contribution level by others in the previous period. The rationale is that although her belief in the current period is not only based on that from the previous period, but also adjusted by taking into account the actual contributions made by others in the previous period. We build on this empirical finding and model the behavior of conditional cooperators accordingly.

Our point of departure from this literature is that the research on public goods takes self-interested players’ free-riding behavior as given and focuses on other social preferences, while we take conditional cooperators’ behavior as given, and study strategic developers’ strategies.
2.3 Research on Modularity and Software Development

Recent studies proposed that modular code architecture can potentially reduce free-riding in OSS development (e.g., Johnson 2002; Baldwin & Clark 2006). In a modular design, the whole software code base is divided into several loosely coupled components or modules, and consequently the code structure is transformed from a monolithic architecture into a modular architecture (Parnas 1972; Baldwin & Clark 2006). The responsibility for developing each module is delegated to specific individuals (Mockus et al. 2002), so participation tends to be equalized under the modular architecture (Johnson 2002; Baldwin & Clark 2006). However, prior studies on modular OSS architecture have several limitations. First, OSS is typically considered as a pure public good (e.g., Johnson 2002; Baldwin & Clark 2006), and virtually none have incorporated the private aspect of OSS development. Second, code contribution by participants is studied as a binary variable, i.e., programmers either contribute or not contribute to the software, and none have considered the differences in the quantity of contribution. However, in reality, contribution can be better quantified as a continuous variable not a binary variable. Third, prior studies do not incorporate the diverse motivations and types of participants and their resource endowments, which play a critical role in their reaction to the design of code structure. In this study we intend to fill these research gaps.

3 MODEL

$\Omega$ developers participate in an OSS project. Among them, $N$ are conditional cooperators and the rest $M$ ($=\Omega - N$) are self-interested van-Neumann Morgenstern utility maximizers (self-interested contributors hereafter). The OSS development takes $T$ periods.

In any period $t$ ($t \in \{1,2,\ldots,T\}$), let $a_{nt}$ denote conditional cooperators’ contribution to the OSS project and let $\mu_{nt}$ denote the average contribution by the other $\Omega-1$ developers in the same period. In a same period, all developers contribute simultaneously, thus $\mu_{nt}$ is unobservable to conditional cooperator $n$ when she decides on $a_{nt}$. Instead, she forms $\beta_{nt}$ over $\mu_{nt}$. Following Fischbacher and Gachter (2010), we model a conditional cooperator’s belief and contribution as follows. In any period $t \geq 2$, a conditional cooperator’s belief $\beta_{nt}$ is determined by

$$\beta_{nt} = r\beta_{nt-1} + (1-r)\mu_{nt-1},$$

where constant $r \in (0,1)$. In other words, her belief (over the average contribution of other developers) in one period is a weighted sum of her belief in the previous period and the true average contribution of other developers in that previous period. In period 1, her initial belief $\beta_{n1}$ is exogenously given. Without loss of generality, let this initial belief be $\beta_{n1} \equiv \beta_1$ for any $n$. For all $t \geq 2$, conditional cooperators’ contribution $a_{nt}$ is in turn influenced by belief $\beta_{nt}$:

$$a_{nt} = k\beta_{nt},$$

where constant $k \in (0,1]$. Consistent with Fischbacher and Gachter (2010), we refer to this conditional cooperator as a perfect (imperfect) conditional cooperator if $k = 1$ ($k < 1$).

A self-interested contributor $m$ is strategic in that, in any give period $t$, she contributes only if doing so leads to a positive cumulative return for herself counting all current and future periods $t, t+1, \ldots, T$. Define her period-$t$ utility as following:

---

2 We will use “code contribution” and “effort” interchangeably in this paper.

3 Fischbacher and Gachter (2010) provide experimental evidence of this belief formation process, and highlighted that it is invariant across periods. This indicates that conditional cooperators do not follow Bayesian inference, which would have implied that the weight $r$ should vary across periods due to the improving precision of the belief.

4 Our results continue to hold if conditional cooperators are heterogeneous in their initial beliefs.

5 We consider homogeneous $r$ and $k$ among conditional cooperators for parsimony of the model. Modeling either or both as heterogeneous among conditional cooperators will not qualitatively change our results.
\[ U_{mt} = \sum_{t=0}^{T-t} u_{m,t+\tau} , \]  

where \( u_{m,t+\tau} = w - x_{m,t+\tau} + \rho \left( \sum_{n=1}^{N} a_{n,t+\tau} + \sum_{i=1}^{M} \gamma_{i,t+\tau} \right) \). In the above, \( u_{m,t+\tau} \) is self-interested contributor \( m \)'s utility in period \( t + \tau \) only, which consists of the following components: her endowment \( w \) for this period (e.g., amount of free time that can be used for OSS development in a given month), her code contribution \( x_{m,t+\tau} \) in this period, and the benefit she receives from the collective code contributions by all developers in this period \( \rho \left( \sum_{n=1}^{N} a_{n,t+\tau} + \sum_{i=1}^{M} \gamma_{i,t+\tau} \right) \). Consistent with the OSS literature and the social preference literature (e.g., Fischbacher & Gachter 2010), we assume that \( 1 / (N + M) < \rho < 1 \), so that (1) code contribution is socially beneficial yet (2) self-interested contributors will always free ride (i.e., contribute nothing) in a static game (i.e., if \( T = 1 \)).

In any period \( t \), self-interested contributor \( m \) chooses her optimal code contribution, \( x_{mt}^{*} \), strategically to maximize her period-\( t \) utility:

\[ x_{mt}^{*} = \arg \max_{x_{mt}} U_{mt} . \]  

In this multi-period model, we also need to define the penalty mechanism for any off-equilibrium strategy by a self-interested contributor. We follow the stylized practice in the repeated-game literature and adopt the following trigger strategy: for any equilibrium, if a self-interested contributor deviates from her equilibrium strategy in period \( t \), all other self-interested contributors will contribute zero code (i.e., free ride) from period \( t + 1 \) on.

Figure 1 illustrates model timing in any period \( t \). The same sequence repeats in all periods except for period 1 (where the initial belief \( \beta_{1} \) is exogenously given rather than inferred). The equilibrium concept we adopt for this paper is Subgame-Perfect Nash Equilibrium.

### 4 RESULTS

#### 4.1 Strategic Cooperation and Free Riding by Self-Interested Contributors

We first analyze the impact of a self-interested developer’s code contribution in any arbitrary period \( t \), \( x_{mt} \), on all developer contributions in periods \( t \) to \( T \), where \( t < T \). To facilitate the analysis, we use \( h_{mt} \) as her \( periods-t\)-to-\( T \) strategy profile (or profile in short). For convenience, we also define the following two particular profiles: the \( periods-t\)-to-\( T \) free-riding profile \( \hat{h}_{mt} \equiv (0, \ldots, 0) \) and the period-\( t \) only free-riding profile \( \hat{h}_{mt} = (0, x_{mt+1}, \ldots, x_{mt}) \). As we will show shortly, the comparison between the generic profile \( h_{mt} \) and the particular profile \( \hat{h}_{mt} \) (or \( \tilde{h}_{mt} \)) facilitates our
analysis of how a self-interested developer’s effort influence contributions by all developers. For notational convenience, define \( \gamma \triangleq r + \frac{N-1}{\Omega-1} k(1-r) \).

**Lemma 1.** Holding contributions by all other self-interested developers constant and compare consequences of profiles \( h_{mt} \) and \( \tilde{h}_{mt} \), we have

\[
\beta_{n,t+\tau} = \tilde{\beta}_{n,t+\tau} + \frac{1}{\Omega-1} \sum_{e=0}^{t-1} \gamma^{t-e-1} x_{m,t+\theta}
\]

(5)

\[
a_{n,t+\tau} = \tilde{a}_{n,t+\tau} + \frac{k(1-r)}{\Omega-1} \sum_{e=0}^{t-1} \gamma^{t-e-1} x_{m,t+\theta}
\]

(6)

for any \( n \) and any \( \tau \in \{1,2,\ldots,T-t\} \). (All proofs omitted due to length limit.)

In Lemma 1, superscript “~” denotes results induced by profile \( \tilde{h}_{mt} \). This lemma shows that, compared to exerting zero effort, \( m \)'s effort \( x_{mt} \) in period \( t \) induces a linear belief (effort) increment of \( \frac{1-r}{\Omega-1} \gamma^{t-1} x_{mt} \) for any conditional cooperator in any future period \( t + \tau \). The larger \( \tau \) is (i.e., the further into future periods), the smaller this induced belief or effort increment is.

We are now ready to analyze the impact of \( x_{mt} \) on all developer contributions in all current and future periods \( t \) to \( T \). This impact can be denoted as \( U_{mt}(h_{mt}) - U_{mt}(\tilde{h}_{mt}) \).

**Lemma 2.** Holding contributions by all other self-interested developers constant, self-interested developer \( m \)'s code contribution \( x_{mt} \) in period \( t \) (\( t < T \)) induces the following total incremental contribution by all conditional cooperators in periods \( t \) to \( T \):

\[
U_{mt}(h_{mt}) - U_{mt}(\tilde{h}_{mt}) = -(1-\rho)x_{mt} + \rho \frac{N}{\Omega-1} k(1-r) \frac{1-\gamma^{T-t}}{1-\gamma} x_{mt}.
\]

(7)

Equation (7) shows that the total incremental contribution by all conditional cooperators induced by \( m \)'s effort is a linear function in \( x_{mt} \). Let \( L \) be the smallest integer that satisfies

\[
\frac{\rho N}{(1-\rho)(\Omega-1)} k(1-r) \frac{1-\gamma^{L}}{1-\gamma} \geq 1,
\]

or equivalently

\[
\gamma^{L} \leq \frac{1}{\rho} \frac{(1-\rho)(\Omega-1+k)}{N} kN
\]

(8)

Notice that \( L \) exists only if \( k > \frac{\Omega-1}{N/(1-\rho)-1} \). We immediately have:

**Proposition 1.** If \( k > \frac{\Omega-1}{N/(1-\rho)-1} \) and \( T > L \), each self-interested developer adopts a two-stage behavior: strategic cooperation in early periods followed by free riding in late periods. Specifically, each self-interested developer contributes \( w \) per period in periods \( 1, \ldots, T-L \), and free rides thereafter. Otherwise, self-interested developers free ride in all periods.

To show the dynamics of the code contribution by the two types of developers, we simulate the code contribution process and show the results of Proposition in Figure 2. The red line represents the contributions by a self-interested developer over time and the blue line represents those by a conditional cooperator. The Parameters used for this illustration are \( \Omega=20; N=15; T=20; r=0.6, k=0.8; \rho=0.5; w=1; \) and \( \beta_{t}=0.8 \).
As the red line in Figure 2 shows, when $k > \frac{\Omega - 1}{N / (1 - \rho) - 1}$ and $T > L$ the behavior of a self-interested developer is characterized by two sharply different segments along time. In later periods $T - L + 1$ to $T$ — which we refer to as free-riding periods — a self-interested developer does not contribute. This result is consistent with prior study of free riding in OSS development. Nevertheless, in each of the earlier periods $1$ to $T - L$ — which we refer to as strategic-cooperation period — a self-interested developer chooses maximum possible contribution level $w$. This result is a sharp departure from the stylized free-riding prediction in the literature. Intuitively, when conditional cooperators exist, a self-interested developer’s effort $x_{mt}$ in period $t$ brings two types of return to herself: a direct return of $\rho x_{mt}$ in this current period, and an indirect return — which is the totality of incremental contribution by all conditional cooperators induced by $x_{mt}$ — in all future periods. When this future return is high enough — which is more likely when the number of remaining periods is larger, this self-interested contributor will exert effort in period $t$.

The green line in Figure 2 shows the contributions over time by any conditional cooperator. During strategic-cooperation periods, the high contributions of self-interested developers induce consistently high contributions from conditional cooperators as well. During free-riding periods, the contributions by conditional cooperators gradually decline over time. We next study total contribution by all developers.

### 4.2 Developer Composition and Total Code Contribution

This subsection answers the following question: fixing the total number of developers, does having self-interested developers in the developer pool always hurt total code contribution, as predicted by the free-riding literature? Note that the answer is apparently yes in a static model (e.g., setting $T = 1$ in our model) where self-interested developers always free ride. In a dynamic model, however, the intuition can go both ways: while the free-riding behavior of the self-interested developers in later periods hurts total code contribution, their possible strategic cooperation behavior in earlier periods may boost total code contribution. We study whether and when the latter effect can dominate the former.

For ease of analysis, we first analyze code contribution by conditional cooperators if all self-interested developers free ride in all periods.

**Lemma 3.** If self-interested developers always free ride, period $t$ belief and code contribution by any conditional cooperator is $\beta_t = \gamma^{-1} \beta$ and $a_t = k \gamma^{-1} \beta_t$. 
We next consider per-period code contribution by all developers. Suppose \( k > \frac{\Omega - 1}{N / (1 - \rho) - 1} \) and \( T > L \) so that strategic cooperation periods exist.\(^{6} \) Let periods 1 to \( T - L \) be the strategic cooperation periods. In any period \( t \leq T - L \), total developer contribution (in this period only) can be calculated by adding up three components: (1) contribution by all conditional cooperators as specified in Lemma 3, (2) contribution by all self-interested developers, and (3) contribution increment by all conditional cooperators as induced by the periods-1-to-\( t \) contribution by all self-interested contributors, i.e.,

\[
Nk\gamma^{-1}\beta_t + Mw + NM\frac{k(1-r)}{\Omega - 1}\sum_{t'=1}^{t-1}\gamma^{t' - t}w.
\]

The third component above follows Lemma 1. Also for notational convenience, when \( t = 1 \), let \( \sum_{t'=1}^{t-1}(\bullet) = 0 \) so the above expression is well defined.

From Lemmas 1 and 3 we also know that, in period \( t = T - L + 1 \) (the first free-riding period), belief of all conditional cooperators is \( \beta_{T - L + 1} = \gamma^{T - t}\beta_t + M\frac{1-r}{\Omega - 1}\sum_{t'=1}^{T - L - t}\gamma^{t' - t}w \). In any period \( t > T - L \), all self-interested developers free ride, therefore period-\( t \) total developer contribution is \( Nk\gamma^{-1-(T - L + 1)}\beta_{T - L + 1} \), i.e.,

\[
Nk\gamma^{-(T - L + 1)}(\beta_{T - L + 1} + M\frac{1-r}{\Omega - 1}\sum_{t'=1}^{T - L - (T - L + 1)}\gamma^{t' - (T - L + 1)}w).
\]

**Lemma 4.** Total code contribution by all developers in all periods is

\[
Nk\beta_t \frac{1 - \gamma^T}{1 - \gamma} + (T - L)Mw + NM\frac{k(1-r)}{\Omega - 1}w\frac{1}{1 - \gamma} (T - L - \gamma^L - \gamma^T) - \gamma^T - \gamma^T).
\]

Furthermore, if we take equality for (8), this total code contribution can also be written as

\[
Nk\frac{1 - \gamma^T}{1 - \gamma}[\beta_t - Mw + (T - L)Mw] + \frac{kN}{\Omega - 1 - k(N - 1)} + \frac{(1 - \rho)Mw}{\rho(1 - \gamma)}.
\]

### 4.3 The Impact of Code Architecture on Code Contributions

In this subsection we consider the case where the code base is divided into \( h \) modules and developers are evenly assigned to these modules. Therefore, each module is developed by \( N / h \) conditional cooperators and \( M / h \) self-interested developers. To keep our analysis parsimonious, we assume both \( N / h \) and \( M / h \) need to be integers. Similarly, we ignore the integer constrain on \( L \) and take equality in (8) when deriving \( L \).\(^{7} \) We use subscript "mo" to denote results under modular design.

To facilitate the exposition, hereafter we refer to \( M / N \) as the “self-interested developer to conditional cooperator ratio”, or “S-C ratio” in short. We next consider the scenario where, when assigning developers into modules, the S-C ratio remain unchanged (before and after modular design).

One example is if all developers of each type are evenly assigned to modules.

**Proposition 2.** Suppose the S-C ratio of the participants pool is preserved in each module in a modular design. When the number of module increases,

(i). the number of free-riding periods weakly decreases,

---

\(^{6}\) If strategic cooperation periods do not exist, it is straightforward that self-interested contributors always hurt total code contribution for any fixed \( \Omega \).

\(^{7}\) The accuracy of such non-integer approximation is higher when \( M, N \) and \( L \) are larger.
(ii). the marginal impact of a self-interested developer’s effort in any period on the contribution by conditional cooperators in all future periods increases, (iii). and the total code contribution weakly increases.

Proposition 2 is significant in that it shows, even without the technological efficiency gains from modular design, modular design can still increase OSS code contribution by better aligning the economic incentives among developers. To see the intuition behind this result, first recall that self-interested contributors choose to strategically cooperate in early periods because doing so increases the per-period average contribution among all developers (minus one conditional cooperator): in the case of $h=1$, $\mu_{it} = (\sum_{p=1}^{N} a_{ip} + \sum_{p=1}^{M} x_{ip}) / (\Omega - 1)$. This further affects the belief and contribution of conditional cooperators in future periods. When the OSS project is divided into more modules, the marginal impact of a self-interested developer’s effort $x_{it}$ on $\mu_{it}$ increases.

5 DISCUSSIONS AND CONCLUDING REMARKS

We develop a model to examine the interactions between two types of OSS developers and the impact of modularity on code contribution in OSS development. Our model explicitly considers the resource heterogeneity, various motivations of the programmers as well as the strategic interactions between them, thus extending the results from prior literature (e.g., Johnson 2002; Baldwin & Clark 2006; Shah 2006). We make several important contributions to the emerging literature of OSS development, open innovation, and contribution to public good in general.

First, we find that self-interested developers who are predicted to free ride under conventional analysis may contribute to a public good out of pure self-interest when the participant pool also includes conditional cooperators. We further find that the existence of purely self-interested participants may, under certain conditions, even benefit the provision of a public good. These novel findings shed new light on the dynamics of OSS development. Note that the above results do not specifically concern OSS development, and thus apply to the provision of other any public goods.

Second, while prior literature have suggested that code architecture will influence the extent of free-riding which endangers the healthy development of OSS communities, they do not distinguish between code contribution and code participation (e.g., Johnson 2002; Baldwin & Clark 2006). We extend prior literature by arguing that these are two different concepts depicting involvement into OSS development: while two programmers may both participated in the coding activities, the amount of the code they contributed can differ substantially, thus describing free-riding as coding the OSS project or not is not accurate, and may even be misleading.

Third, the results show that the existence of both self-interested developers and conditional cooperators may help improve the total code contribution. In a one-shot that the composition of teams matters in OSS development have interesting implications for OSS development. Our results contribute to our understanding about the broader literature of open innovation as well. Open innovation is attracting increasing attention from both researchers and practitioners, and it reflects the innovation trend that firms and organizations are increasing rely on the external as well as the internal resource for innovation due to stronger global competition and accelerated flow and exchange of knowledge and information across firm boundaries (Chesbrough 2003). The idea of open innovation originated from OSS development (e.g., Gruber & Henkel 2006, West & Gallagher 2006), and has quickly proliferated into other research areas as well. Researchers suggest that one of the most important challenges to open innovation is how to design the model so to facilitate innovation processes (Chesbrough 2007). So far, quite some research has devoted to the study of organizational design (e.g., Jacobides & Billinger 2006; Dittrich & Duysters 2007), but few have consider the fact that organization members may possess diverse sets of motivations and preferences. When these are taken into consideration, one has to keep in mind that, when plying modular project design, it is
critical to have a right mix of members of social preferences in every module in order to maximize the quasi-strategy complementarity among developers.

References


