A Group and Reputation Model for the Emergence of Voluntarism in Open Source Development

Lik Mui, Mojdeh Mohtashemi, Sameer Verma

lmui@stanford.edu, mojdeh@mit.edu, sverma@sfsu.edu

Abstract

The open source development efforts of the past decade have brought much benefit for the users of such free software as Apache, GNOME, GNU/Linux, JBoss, among others. Many questions have been raised about the motivation behind contributions of developers around the world to create useful software and then freely give it away Traditional rational agency theory predicts that rational, self-interested individuals are not likely to volunteer contributing to the common good (Olson, 1965; Hardin, 1982). In contrast, the voluntary actions of the open source developers seem altruistic and defy our intuitive notion of how a market-driven, capitalistic society works.

Von Hippel and von Krogh (2003), Lerner and Tirole (2000) among others have documented field observations on possible motives for why top developers around the world would volunteer to participate in open source projects. In this paper, we develop a multi-agents model of open source development with a focus on how reputation and social groups influence the emergence of volunteerism for open source developers. We analytically derive conditions for how voluntary actions among the developers would emerge to dominate over overt self-interested concerns. We also report simulation results which correlate well with field observations about voluntary actions in the open source world. For the management community, results of this paper illustrate how such implicit and indirect incentives as reputation gain and group identification can have powerful influence on motivating voluntary actions in software development.

Introduction

Voluntary acts are necessary for actions such as tasks forces, special interest groups, and many other informal organizational actions (Katz, 1964). However, potential volunteers face the following dilemma: while contributing to the welfare of everyone in a group, volunteers themselves typically incur costs not shared by the group. A large body of literature has attempted to address the motivation of these seemingly altruistic, cooperative individuals who volunteer to contribute to the common good freely (Diekmann, 1985; Munighan, *et al*., 1993; Andreoni, 1995; Ostrom, 1997, 2000). Nevertheless, Tullock (1995), Lichbach (2000), and others have observed a so called "5 Percent Rule" – in a group engaged in collective action, only a small percentage (about 5%) of these "… are willing to run great risks ... for the poor or the public good" (Tullock 1995).

In the context of these literatures, the open source phenomenon poses significant challenges: the Open Source community depends *entirely* on voluntary actions by its contributors in order to exist. Thus, the open source development process can be cast in light of a number of sociological dilemmas: social traps (Diekmann, 1985; Rapoport, 1988), collective action dilemma (Lichbach, 1996; Ostrom, 2000), irrational behavior (Olson, 1965), *etc*.

This paper first describes the open source movement and field studies on what motivates developers in the open source world. We then points out the background literature related to voluntary games and collective actions. The bulk of the paper will be devoted to describing an analytical model for how voluntary acts by developers can emerge to contribute to the common good of open source software. This model is based on earlier work on evolution of cooperation from Mohtashemi and Mui (2003). We then report simulation results based on the model and discuss how these results correlate with empirical observations on voluntarism in general and the working of the open source community in particular. A brief conclusion ends this paper.

Background Literature

It is a common belief that the success of an open source project relies on the support from, and cooperation among, various groups involved in the open source community. Motivations for contribution have been attributed to altruistic behavior (Raymond, 1999), a gift-giving culture (Bergquist,and Ljungberg, 2001) and in some cases, a negative consensus toward large software corporations that sell proprietary software (Dotzler, 2003).

Explaining the Open Source Incentives

The behaviors of open source developers to share code without monetary or organizational incentives have challenged traditional rational agency analyses. Lerner and Tirole (2000) have suggested two main approaches to address these seemingly irrational behaviors. One is to study career prospects as an incentive and the other to consider open source participation as a signaling of programming abilities. These hypotheses are echoed by findings by von Hippel and von Krogh (2003), and open source pioneers such as Raymond (1999).

> "*The 'utility function' Linux hackers is maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers*."
>
> *-- Raymond, 1999*

Participation in open source projects is well acknowledged to have significant reputation benefits. According to Raymond (1999), successful open source developers create "good reputation among one's peers, attention and cooperation from other, … higher status [in the] … exchange economy."

Von Hippel and von Krogh (2003) have proposed a "private-collective model" for explaining how the innovation process within the open source world works. In contrast to a collective action model where free-rider incentives induce participants not to contribute, this model describes incentives for those who contribute to the development and not to free-ride based on private-investment considerations:

- Increase of innovation diffusion and so increase an innovator's innovation-related profits through network effects
- Open source developers inherently obtain private benefits from the contributions. These private benefits include: learning and enjoyment during the development process, gaining a sense of ownership and control over their work product, being able to participate in a community

Traditional model of innovation assumes that the innovator typically invest in innovation based on an incentive that their efforts will be repaid when that innovation is diffused to the consumers. Innovation within open source projects defies this incentive structure because by definition, open source projects will result in freely distributed, downloadable intellectual property. Von Hippel and von Krogh (2003) argues that the open source development does not violate the traditional private-investment model of innovation due to several causes:

- Positive network effects which could lead to increased sales of complementary goods
- Private losses associated with freely revealing source code is low
- For developers, since the costs to freely reveal and diffuse their development effort are low, small rewards "should be sufficient to induce the [voluntary] behavior"

Von Krogh (2002) and Lerner and Tirole (2000) have observed that rewards to the open source participants include: "elevated reputation among fellow developers, expected reciprocity, and incentives to build a community." In addition, other benefits for the open source developers: (1) gain skills and knowledge usable in their paid jobs (2) sense of fun and control performing the open source development project. There are also delayed (signaling) incentives: (1) improved prospect for career options by signaling his abilities, (2) possible shares or other monetary interests in commercial open source companies, (3) enhanced chances for access to venture capital market, (4) "ego gratification incentive stems from a desire for peer recognition."

"Strategic complementarities" is an economic term that can be used to describe why developers are eager to join popular open source efforts (Lerner and Tirole, 2000). To be associated with "fads" in the open source community, developers can exhibit strong signals to a large body of "audience" (e.g., peers, labor market, venture capital community) of their performances. Consequently, they are more motivated to produce high-quality work.

An N-person Volunteer's Dilemma Game

For voluntary actions in general, field experiments have shown that instrumental rewards such as increased money, reduced work time, and recognition encourages voluntary actions (Adam, 1965; Pillutla and Munighan 1993; Munighan, et al., 1993). For studying voluntary actions, an N-person game first described by Diekmann (1985) is often used. (The model to be described in this paper will adapt this N-person game to the open source development process.)

This game assumes that a volunteer produces a common good but at a personal cost. When deciding on her action, a volunteer faces a dilemma: she can volunteer to contribute to a common good but gain less than others in the group, or she can hope for another to volunteer and free-ride her that effort. When there is no volunteer, no one benefits: all lose. The personal dilemma is a direct result of the incomplete information an individual has on how others would choose.

Let U be the utility for the collective good, and K be the production cost. An N-person volunteer's game can be summarized by the following payoff matrix:

	0	1	2	…	N - 1
C	U-K	U-K	U-K	U-K	U-K
D	0	U	U	U	U

where C represents "cooperation" by a given individual to volunteer and D denotes "defection" by that individual to not volunteer (thereby hoping for free-ride).[1]

If N persons realize the need for a software feature, each person might be inclined to wait for someone else to development the feature. This is the "free-rider" incentive. The person who develops the feature incurs a cost K (in terms of opportunity cost, time, and possibly lost wages). Everyone else gains a utility of U. As the size of the population increases, each agent perceives that they share less responsibility for the common good – an effect called the "bystander effect" or the "diffusion of responsibility." (Latane and Darley, 1968, 1970).

Rational Egoist

What is the rational behavior for the volunteer's game?

There exists an asymmetric strong equilibrium with (N-1) free riders and one volunteer who contributes the voluntary effort (Diekmann, 1985). To achieve this equilibrium in an iterated game, *social coordination* can ensure the volunteer to gain an expected utility of (U - K / N). However, as in many social situations, such coordination is not possible (Darley and Latane, 1968).

Explaining Incentives for Voluntary Actions

In the game theoretic literature, both prisoner's dilemma (PD) and volunteer's dilemma arise due to rational agents using strategies which take advantage of short term gains at the expense of the group's gain, and at the expense of future payoffs. The PD game in particular has provided a fertile group for examining strategic actions by Axelrod (1981, 1984) and many following his work (*e.g.*, Busch and Reinhardt, 1993). As a highly successful strategy, the Tit-for-Tat (or the "trigger strategy") can be used as a reference to explain voluntary actions. The explanation hinges on earlier work by the evolutionary

[1] Although "cooperation" has a more general meaning in the evolutionary literature (*e.g.*, Mohtashemi and Mui, 2003; Mui, 2003), this paper treats the act to cooperate in the open source world as a voluntary action toward an open source effort – including bug fixing, development, documentation, *etc*. "Defection" denotes the opposite meaning of not volunteering toward an open source effort.

biologist Trivers (1971), who observed that many human activities are based on *reciprocal altruism*. As de Wall and Luttrell (1988) noted, Trivers' reciprocal altruism assumes a fair and balanced exchange over the longer term. In order for reciprocal altruism to induce "evolution of cooperation" for repeated PD game using the Tit-for-Tat strategy, Axelrod (1981, 1984) and others have demonstrated a critical threshold. This threshold depends on the ratio of payoff benefits to cost and the probability of future encounters. Cooperation can emerge among rational egoists despite the absence of a central authority. Another way to interpret this result is that acts of *reciprocal altruism* are not altruistic at all, but rather, calculated acts of self-interest with a long time payoff horizon. For the open source world, reciprocal altruism can be used to claim that developers are simply hedging their voluntary contributions on future reciprocal payments (either in terms of job offers, leadership opportunities, venture funding, *etc.*).

Yet, there are key differences between volunteering and reciprocal altruism. As Smith (1983) pointed out, volunteering behaviors usually have more market value to the recipient than it does to the volunteers themselves. Volunteers are motivated to perform such acts based on expectation of psychic and moral benefits (Hoffman 1981; Allen and Rushton, 1983). They have suggested that voluntary acts might be instinctive reactions to seeing others in distress. In addition, Berkowitz and Daniels (1964) observed that voluntary acts of kindness toward others are often performed without expectation of direct rewards, only with expectations of "generalized reciprocity" from the social group. Batson (1987) has further documented through a series of experiments that pro-social behaviors are more likely among empathetic individuals – such as those among a **closely knitted social group**. Regan and Totten (1975) have also found the importance of empathic attitude for inducing prosocial altruistic actions without enforcing contracts.

The generalized reciprocity notion nevertheless points to voluntary actions that are selfishly motivated – for either tangible or intangible gains. Research results in the voluntary organizations generally support the claim that voluntary actions are correlated to the ratio of individual benefits to cost (Smith, 1983). Stinson and Stam (1976) have found that volunteers' actions could be motivated by needs to improve their skills and employability. People volunteer in order to seek social interactions and

companionship (Sharp, 1978). Murnighan et al. (1993) have documented field findings which suggest that voluntary actions can be connected to promotion concerns in an organization. In our daily lives, many of us have probably experienced working in companies which explicitly reward and complement employees for helping their community, "being good social citizens." Even for such voluntary actions as donating blood, Kessler (1975) has reported findings that the amount of blood donated is correlated to demands in donors' occupations. With these findings, the altruistic links with many voluntary actions can be easily questioned.

Without explicit rewards, are there other motivators for voluntary actions? Kinship (Hamilton, 1963, 1964) and group-based affiliation (Williams, 1971; Wilson and Sober, 1994) have been suggested to motivate cooperative behaviors. Kramer (1993) has shown that people are more likely to volunteer when their organizational identification becomes strong. Strong social bonds can create positive environments for individuals to participate in voluntary actions toward the common good.

Modeling Open Source Development

The open source community is composed of developers who freely distribute their innovations in the form of source code. Voluntary actions by these developers contribute to the common good – software binaries and codes available for all to download.

The literature of voluntary action tells us that volunteers are not entirely altruistic but are motivated by various tangible and intangible gains. This is also observed among the open source developers:

> "*The 'utility function' Linux hackers is maximizing is not classically economic, but is the intangible of their own ego satisfaction and reputation among other hackers*."
>
> *-- Raymond, 1999*

Economists Lerner and Tirole (2000) have observed the importance of "elevated reputation among fellow developers" as an important incentive for open source developers. Due to all of the above field

observations, our model for open source development must therefore incorporate reputation calculation in determining agent actions.

Studies have also indicate that voluntary actions could have stemmed from a socialization need among empathetic individuals (Sharp, 1978; Batson, 1987), or a calculated decision to participate in prominent development groups (Lerner and Tirole, 2000). Our model of open source should also account for group affiliation and cooperation among group members within open source projects.

In the sections to follow, we will present a multi-agents model for open source which has the following characteristics:

- Players can communicate and inquire information about the reputation of their co-players.
- Players belong to social groups as non-overlapping cliques of "core development teams." Members in clique are cooperative with other members of the same clique regarding feature or bug-fix requests.
- Information about the reputation [2] of co-players is not obtained randomly nor globally; rather, players selectively acquire information from their acquaintance network by taking advantage of the collective memory of the social group to which they belong.
- Information is not propagated randomly in a population. New information resulting from new interactions modifies the content of the collective memory of a recipient and is therefore selectively propagated through the recipient acquaintance network.[3]
- A social network is an evolving dynamic entity. With new interactions, new links are created which in turn increase the likelihood of any two randomly selected players would *know* each other.

---

[2] Many types of reputation exist: global, observer-based, proprogated via n-degree separations, *etc*. Reputation in this paper is inferred from all interactions in the acquaintance network which an agent belongs to. Discussions and analyses on these various types of reputation can be found in Mui (2003).

[3] The main information propagated includes history of cooperation and defections, number of encounters with a given agent, etc. Mohtashemi and Mui (2003) call these information "social information."

Mapping Open Source to a Model

The open source community relies on the support from, and cooperation among, various groups involved in the open source community. Different roles can be approximately categorized into four groups, including core developers, patch submitters, source code testers, and end-users (Dotzler, 2003).

Core developers are a small group of people who contribute most of the code and control the software releases. In the case of Apache, a popular web-server project, core developers account for over 80% of the coding (Mockus, et al, 2000). Apache has approximately 25 core developers. These members are ones who have the rights to accept or reject requests for features or bug fixes. Following that group is a collection of patch submitters, bug testers and end-users. Although end-users may be reported to be largest group of all, they are usually only interested in using pre-compiled binary software and rarely provide explicit feedback (or any other contributions) to the projects. Most feedback comes from bug testers and patch submitters. We will treat the patch submitters and bug testers in one category as "feature (or bug-fix) requester."

The interactions between core developers and feature requesters usually happen via conversations on e-mail lists or through more well-defined mechanisms such as bug reporting systems such as Bugzilla (Dotzler, 2003). In most cases, several rounds of interaction lead to the fruition of a strategy where certain requests are accepted, while others are rejected, and a new generation or version of the software is released. This is typically called a release in the open source world. It is common for core developers and feature requesters to change their interaction strategies across each generation – some are more active in some than in others.

Based on the game theoretic, open source, and voluntary action literatures, we expect this model to exhibit some of the following characteristics:

- Voluntary actions to participate in open source activities can emerge for rational, self-interested agents if these agents can take advantage of reputation information and group identification for modifying their interaction strategies (Mohtashemi and Mui, 2003).

- Some players are willing to cooperate for the common good – volunteer to develop free software – even when no explicit payoff is given for such actions (Lerner and Tirole, 2000; Ostrom, 2000).
- As observed by Diekmann (1985), Murnighan, et al., (1993) and others, as the size of a population increases, players' willingness to cooperate for the common good decreases due to decreases in group identification and empathy (Batson, 1987).
- Participation in common goods activities increases the socialization for the players, which acts as a motivator for more such participation (Sharp, 1978; Batson, 1987)

Modeling Open Source with Social Information

We construct a multi-agents model to capture the essential features of the open source development process. Variants of this model have been used in the past for modeling evolutionary phenomena (Nowak and Sigmund, 1998; Mohtashemi and Mui, 2003; Mui, 2003). It can be shown that the stage game described below is related to the volunteer's stage game as described earlier (Diekmann, 1985).

Consider a population of n individuals divided into non-overlapping groups of the same size, s. We use the term "agents" to denote such individuals to emphasize the notion that these individuals are in fact rational, albeit without explicit payoffs for guiding their actions – as discussed below. Each agent can assume the role of a *developer* or a feature or bug-fix *requester*.

Each *group* models a set of "core developers" who are working in an open source project. We model the underlying graph structure of a group as a *clique*.[4] Members in a group are more likely to cooperate with each other during feature (or bug-fix) requests than they are to a non-member request. Acquaintances of a group include those who have interacted with members of the group – members of the acquaintance set can evolve over time.

A *generation* models one complete development cycle for a group of affiliated open source developers. Every generation consists of a fixed number of rounds, m. In every round, two agents are

[4] A clique is a fully connected graph.

selected at random: one as a developer and the other as a feature (or bug-fix) requester. The developer has an option of cooperating with or defecting upon the feature requester. If the developer cooperates, he not only would not get any payoff but would incur a cost of $c > 0$. The requester receives a benefit value of b ($b>c$):

b : benefit per round for the requester who receives a feature implementation

c : cost per round to a developer who satisfies a given feature request

At the beginning of each generation every agent assumes a unique clique of acquaintances. Every agent j possesses a strategy, k_j: $k_j \in \{ -5, -4, \ldots 5, 6 \}$ and an reputation score about agent i, s_{ij}: $\in \{ -5, -4, \ldots 4, 5 \}$ [5]

k_j: agent j's strategy (cooperate if the recipient's reputation $\geq k$)

s_{ij}: reputation score that agent j has about agent i

An agent j *cooperates* if he is willing to developer a feature as requested by agent i. When an agent has a strategy $k_j = -5$, he is a *complete cooperator*: he would fulfill any request because any requester's reputation score would be $\geq k_j$. On the other hand, when an agent has a strategy $k_j = 6$, he is a *complete defector*: he would deny any request because all requester's reputation score would be $< k_j$.[6] The private utility gained by the cooperative agent j is an increase in his reputation – despite an opportunity cost c for fulfilling a request. Note that the cooperative agent j receives no direct payoff from agent i during a given round of exchange.

To observe how different strategies affect the fitness of agents, all agents assume a reputation score of zero at the beginning of a generation.[7] A potential developer j cooperates if the reputation score s_{ij} of the recipient i is at least as high as j's own strategy value (k_j). The reputation scores of agents are only

---

[5] The ranges of these values have been successfully used in the past in modeling evolution of cooperation in general in Nowak and Sigmund (1998), Mohtashemi and Mui (2003), and Mui (2003).

[6] Because agents with negative strategy k_j tend to cooperate while those with positive strategy k_j tend to defect, the former will be dubbed "cooperators" while the latter "defectors" in the simulation discussion later.

[7] This assumption can be removed to observe faster convergence of strategies toward the more "robust" strategy which is copied by all other agents.

known to and updated for their acquaintances.[8] If a potential developer cooperates, his reputation score is increased by one unit; otherwise it is decreased by one unit. A developer performs one of the two actions, cooperate or defect.

Developer's action space: { *cooperate*, *defect* }

cooperate: refers to a developer's action to fulfill a feature or bug-fix request from a requester

defect: refers to a developer's inability or refusal to fulfill a feature or bug-fix request from a requester

Payoff matrix:

	cooperate	*defect*
developer	*-c*	*0*
recipient	*b*	*0*

Reputation update matrix:

	cooperate	*defect*
developer	*+1*	*-1*
recipient	*0*	*0*

After a developer's action, the feature requester and members of the developer's clique update their perception of the developer's reputation; the developer will then become a 'one-way acquaintance' of the feature requester, *i.e.*, if in future rounds the requester is chosen as a developer, in addition to her acquaintances, she will also have access to the previous developer for reputation inquiry about her opponents. In other words, the developer is being added to the feature requester's acquaintance set but not vice versa. This is because the developer gains no information about the feature requester by volunteering his development effort to fulfill a request. However, the requester would have observed the developer in action and can hence attest how cooperative the developer has been.[9] The aggregate reputation of a developer depends on his accumulated fulfillments of feature (or bug fix) requests for a given requester. Intuitively, the more a developer successfully delivers fulfillments for a requester, his reputation will correspondingly be enhanced for that requester. Conversely, if such fulfillments are few in between if at all, his reputation will then suffer.

---

[8] The notion of an acquaintance set is treated dynamically, as discussed earlier.

[9] As a simplifying first step, our model assumes that all fulfillment of a request by a developer is completed.

If a potential developer j does not know the reputation score of the feature requester i, he will make use of the social information available by asking all his acquaintances whether they have ever interacted as feature requesters against the current recipient i -- *i.e.*, if the current feature requester is a one-way acquaintance of any one in the developer's acquaintance set $\boldsymbol{Q}_j$. If no information is learned, the developer will assume a reputation score s_{ij} of zero; otherwise, he assesses the reputation of the requester by adding up her reputation scores, provided by members of the acquaintance set $\boldsymbol{Q}_j$, and dividing by the total number of encounters.

Reputation s_{ij} of a requester i in $\boldsymbol{Q}_j$ = ratio of cooperation over total number of *i's* encounters in $\boldsymbol{Q}_j$

Such an averaging scheme has the benefit of transparency during analysis. More sophisticated reputation estimation techniques have been discussed in Mui (2003).

A potential developer j compares the computed reputation score s_{ij} for a potential requester i to his own strategy k_j. j volunteers to fulfill i's request if $s_{ij} \geq k_j$: this is the decision rule for the open source developer j. Therefore, the outcome of a round depends on the probability of knowing the requester's reputation, which is derived from the collective memory embedded in one's acquaintance set. Furthermore, the underlying social structure is itself evolving as agents meet over time, which causes the probability of knowing a randomly selected requester to increase over the lifetime of a development generation.

Modeling strategy evolution

To model how voluntary action strategies evolve over time, our model uses a multi-generation scheme. In this scheme, n number of agents at the end of a generation disbands and reorganizes themselves into new groups of agents (core developers for new projects). Within a generation, each agent gains payoff benefits (as requester) as well as a cumulative reputation score (as developer). At the end of a generation, that agent's sum of payoff benefits determines how many new agents in the new generation will copy his voluntary action strategy k. The more "fit" a given cooperative strategy, the more new

agents would be included to copy that strategy in the new generation. In turn, how fit an agent is in a generation depends on his accumulated reputation in the community at large.

Open Source Development Model: an Analysis

For the analysis of the framework presented in the previous section, several additional variables need to be defined:

A_i : the average number of acquaintances per agent at round *i*

This variable represents the average network connectivity per agent at round *i*. Then A_0 is the average initial "core developer" clique size at the beginning of each generation.

q_i : the probability that a potential developer knows the reputation score of the feature requester at round *i*.

In terms of A_i, q_i can be expressed as:

$$q_i = A_{i-1}/(n-1) \tag{1}$$

i.e., the likelihood of knowing the feature requester at round *i* is a function of the average number of acquaintances for the developer from the previous round over *n*-1 possible acquaintances one can have in a population of size *n*. The average network connectivity per agent at round *i* can be expressed as the following recurrence:

$$A_i = A_{i-1} + (1-q_i)\frac{A_{i-1}}{2n} \tag{2}$$

This relation shows that a new link is created between two agents in every round of the game only if the developer does not know the reputation score of the feature requester, in which case the developer is added to the acquaintance set of the feature requester resulting in as many as A_{i-1} new links.

Rewrite A_i using equation (1), the following can be derived:

$$A_i = (1+\frac{1}{2n})A_{i-1} - (\frac{1}{2n(n-1)})A_{i-1}{}^2 \tag{3}$$

Substituting $X_i = A_i/(2n+1)(n-1)$, equation (3) can be expressed as the following canonical form:

$$X_i = (1+\frac{1}{2n})X_{i-1}(1-X_{i-1}) \tag{4}$$

which is the familiar logistic equation.

As $n \to \infty$, the growth rate, $\left(1+\frac{1}{2n}\right) \to 1$. Therefore, for n>1 the system is stable with the nontrivial fixed point $A^* = n-1$. This is the maximum network connectivity per agent, *i.e.*, the maximum number of acquaintances an individual can have in a population of size n.

Now consider a population of size n consists of two types of agents: defecting developers who never help the feature requester and discriminators who only help agents with good reputation. (Discriminators are those with strategy $k = 0$). Let the frequencies of these populations be:

x : the frequency of discriminators

y : the frequency of unconditional defectors

For a discriminating developer, a feature requester has a good reputation, *G*, if he is known to have cooperated the last time; otherwise he has a bad reputation, *B*. We modify this model by imposing a *social structure on acquaintance relations*. If discriminating developers do not learn new information about their feature requesters by asking their acquaintances, they always cooperate. This means that in the absence of information, defectors can be mistaken for good scorers by discriminators. Therefore at the beginning of each generation, when no information about agents' behaviors is available, all agents are assumed to be good scorers. Let $A_x(i)$ and $A_y(i)$ denote the respective payoff for discriminators and defectors at round i. These two quantities can be expressed as:

$$A_y(i) = \frac{b}{2}\{x(1-q_i) + xq_i\frac{y_G(i)}{y}\}$$
$$A_x(i) = \frac{1}{2}\{-c(q_i g_i + 1 - q_i) + bx - bxq_i(1-\frac{x_G(i)}{x}\}$$

where $x_G(i)$ and $y_G(i)$ are the respective frequency of good scoring discriminators and defectors at round *i*. These frequencies can be expressed as:

$$x_G(i) = \frac{1}{2}\{x_G(i-1) + x(1-q_i + q_i g_i)\}$$
$$y_G(i) = y2^{1-i}$$

The differential payoff at round *i* for discriminators is:

$$D_x = \frac{1}{2}\{-c(q_i g_i + 1 - q_i) + bq_i(g_i - 2^{1-i})\}.$$

Here $g_i = x_G(i) + y_G(i)$ is the proportion of good scorers at round *i*. Because everyone is assumed to have a good reputation at the beginning of each generation:

$$g_i = \begin{cases} 1 & i = 1 \\ \frac{1}{2}\{g_{i-1}(1+q_i x) + (1-q_i)x\} & 2 \le i \le m \end{cases}$$

The differential total expected payoff for discriminators is then:

$$P_x = \frac{1}{2}\{-c[m - \sum_{i=1}^{m} q_i(1-g_i)] + b\sum_{i=1}^{m} q_i(g_i - 2^{1-i})\} \quad (5)$$

For $P_x > 0$, we must have:

$$c\{m - \sum_{i=1}^{m} q_i(1-g_i)\} < b\sum_{i=1}^{m} q_i(g_i - 2^{1-i}) \quad (6)$$

Divide both sides of inequality (4) by *m* to get:

$$c\{1-\sum_{i=1}^{m} q_i(1-g_i)/m\} < b\sum_{i=1}^{m} q_i(g_i - 2^{1-i})/m \tag{7}$$

To interpret inequality (7) in an intuitive manner note that the following ratio is the average per round probability of knowing and helping a good scoring discriminator, *i.e.*, the probability of a discriminator making the right decision:

$$\sum_{i=1}^{m} q_i(g_i - 2^{1-i})/m$$

Similarly, the following ratio is the average per round probability of knowing and not helping a bad scorer:

$$\sum_{i=1}^{m} q_i(1-g_i)/m$$

Hence, the term $1-\sum_{i=1}^{m} q_i(1-g_i)/m$ describes the average per round probability of knowing and helping a bad scorer, *i.e.*, the probability of a discriminator making the wrong decision. Inequality (5) then asserts that for discriminators to outperform the defectors the average per round benefit received by a good scoring discriminator must out-weigh the average per round cost to a discriminator for helping a bad scorer. Therefore trust pays off if it is based on information and placed upon the trustworthy.

If we rewrite inequality (7) in terms of cost-to-benefit ratio we derive that for discriminators to be evolutionarily stable we must have:

$$c/b < \frac{\sum_{i=1}^{m} q_i(g_i - 2^{1-i})/m}{1-\sum_{i=1}^{m} q_i(1-g_i)/m} \tag{8}$$

This is the ratio of knowing and helping a good scoring discriminator to knowing and helping a bad scorer must out-weight the cost-to-benefit ratio.

Note that equation (8) can be used to illustrate the calculated motive behind voluntary actions as described by Smith (1983) and Murnighan, *et al*., (1993), and those behind open source developers as postulated by Lerner and Tirole (2000). The calculation here does not involve explicit benefits to an agent; rather, it is a function of reputation scores and probability of knowing other agents in a community. The voluntary action strategy to cooperate with the requester is only robust if the cost-to-benefit ratio of reputation and group affiliation in inequality (8) is satisfied.

A property of the framework outlined here is that the underlying network of trust relations is itself evolving as players interact. Hence, A_i and q_i are changing over time. As the game continues therefore, the threshold on *c/b* increases (see inequality (8)). Under such changing environment the payoff to cooperative strategies may be negative at the beginning, but over time they may be able to recover and even outperform the exploiters. This is achieved through two parameters in the system: the initial clique size, A_0, and the number of rounds, m. The effect of varying the initial clique size on the long term outcome of the game will be demonstrated in a set of simulations in the next section. But even under fixed initial clique size, the probability of knowing the reputation of an opponent through one's acquaintances is increased over time – following the equations (2)-(4).

Social Information: Simulation Results

Simulation results in this section are based on the model proposed and analyzed earlier in the paper. Figure 1 shows that the probability (q_i) of knowing a randomly selected agent's reputation score increases at round i. Initially a population of n=100 individuals is divided into non-overlapping "core developer" cliques of size s=4. As the game is continued, agents meet and make new acquaintanceships. The average number of acquaintances per player increases with every round of the game, and therefore the probability of knowing the opponent's reputation also increases. Here a generation consists of 10,000 rounds. The rise in the probability of familiarity is consistent with the analytic result in Equation (2) in the former section – a trivial result illustrating that socialization increases one's number of acquaintances.

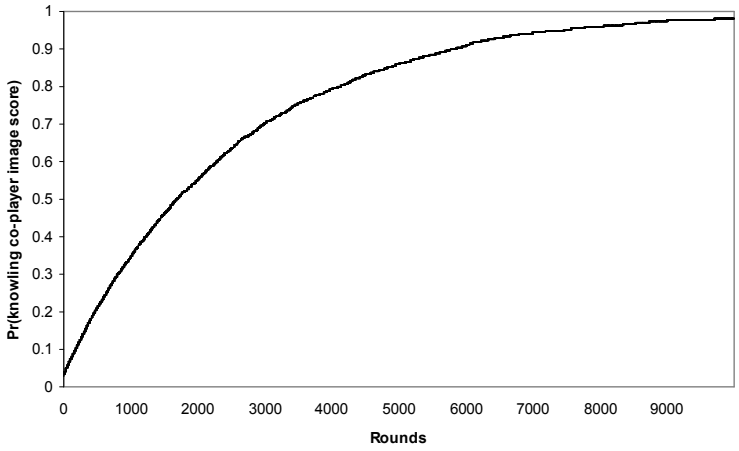

Figure 1. Dynamics of acquaintanceship.

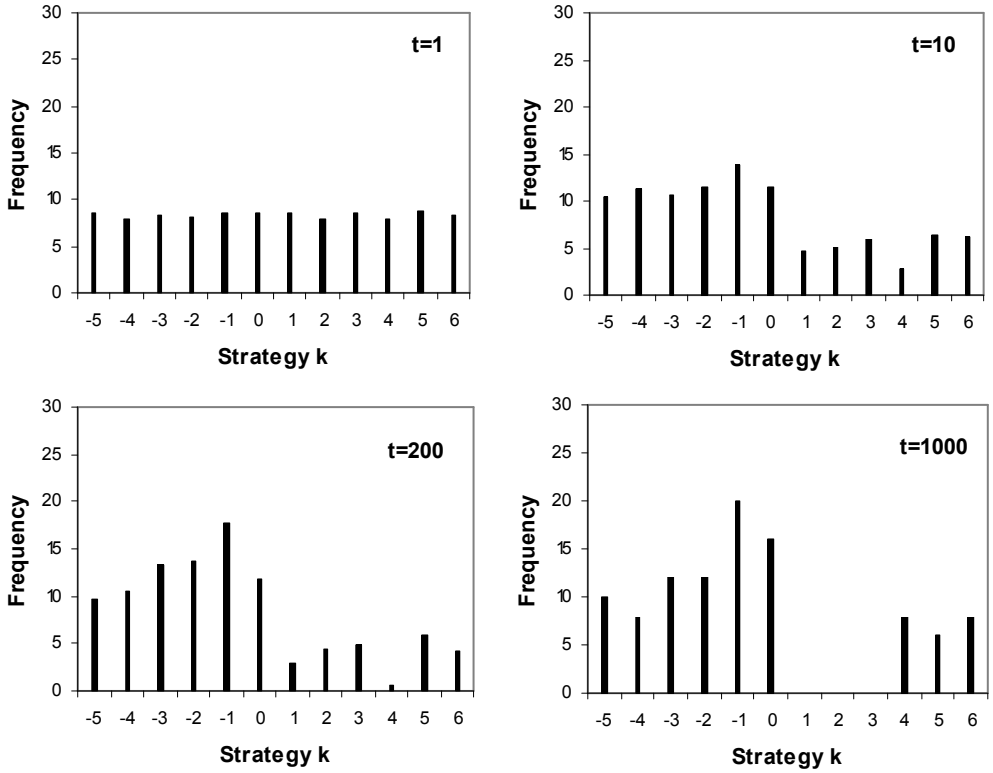

Figure 2. Frequency counts (averaged over 50 runs each) for number of agents with strategy *k* at generation *t*. Developer agents choose to cooperate or defect (on feature requests) based on the reputation score of the feature requesters.

Shown in Figure 2 are aggregate results based on 50 simulation runs each running with number of generations *t*. Every generation consists of a fixed number agents (n = 100) and a fixed number of rounds, $m=10n$. Under these assumptions, the likelihood for a pair of individuals to meet more than once is very small – making the need to gather reputation information from one's peer the more important for making decisions. As mentioned previously, strategy *k* ranges from -5 to 6 where k=-5 represents unconditional cooperators, k=6 represents complete defectors, and k=0 represents the most discriminating agents. Agents with strategy $k < 0$ indicates more cooperative agents while agents with strategy $k > 0$ indicates those who are more willing to defect. The reputation scores range from -5 to 5. A potential developer *j* volunteers to satisfy a request only if the reputation score s_{ij} of the feature requester *i* is at

least as large as j's strategy $\underline{s_{jj}} \geq k_j$. Agents in new generations copy the successful strategies [10] of their former generation's agents -- subject to a copy error rate of 0.001. For developer agents, Figure 2 shows that as the number of generations increases, those who are willing to volunteer to fulfill development requests are more likely to have their strategies copied by other agents.

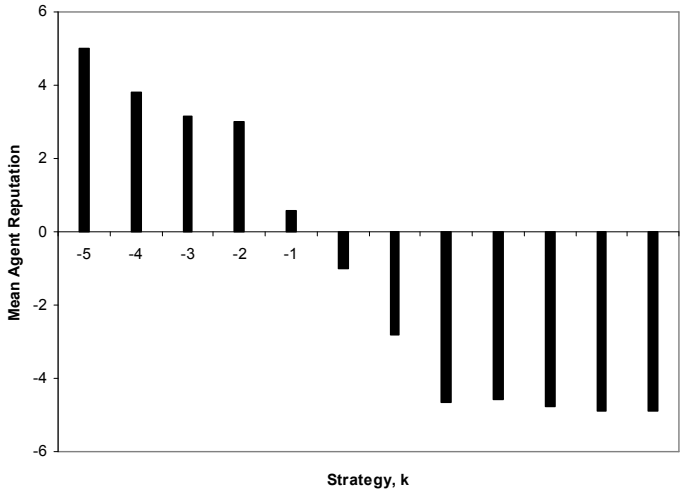

Figure 3. A sample run's average agent reputation for each type of interaction strategy (k) at the end of the first generation (t = 1) in Figure 2.

How do the reputation scores for agents with various strategies vary in the experiment depicted in Figure 2? Figure 3 plots the average reputation score for agents with the same strategy value k at the end of the first generation -- where there are about equal proportions of agents with each strategy k (see top-left graph in Figure 2).[11] For the more cooperative agents (with $k < 0$), their average reputation on average is higher than those whose strategies are less cooperative (with $k > 0$).

Figure 4 shows the results of computer simulations for a population of n individuals with an initial "core developer" clique size s=4. We sampled the frequency distribution of strategies over 106 generations for population sizes n=52, n=100, and n=200. All other parameters are as in the experiment

[10] "Success" refers to the fitness of an agent as measured by his individual accumulated benefits (*c.f.*, the previous section's discussion).

[11] Comparing agents at later generations become less statistically significant as the distribution of strategies becomes increasingly skewed toward the more robust strategies (*c.f.*, Figure 2). This is particularly the case for positive k.

in Figure 2. As observed by Diekmann (1985), Murnighan, et al., (1993) and others, voluntary actions among social agents are expected to drop as the population size increases. Figure 4 shows that as the population size increases, the relative proportion of cooperative agents versus defective ones decreases. In other words, in larger population setting, less proportion of agents are willing to volunteer to fulfill development requests.

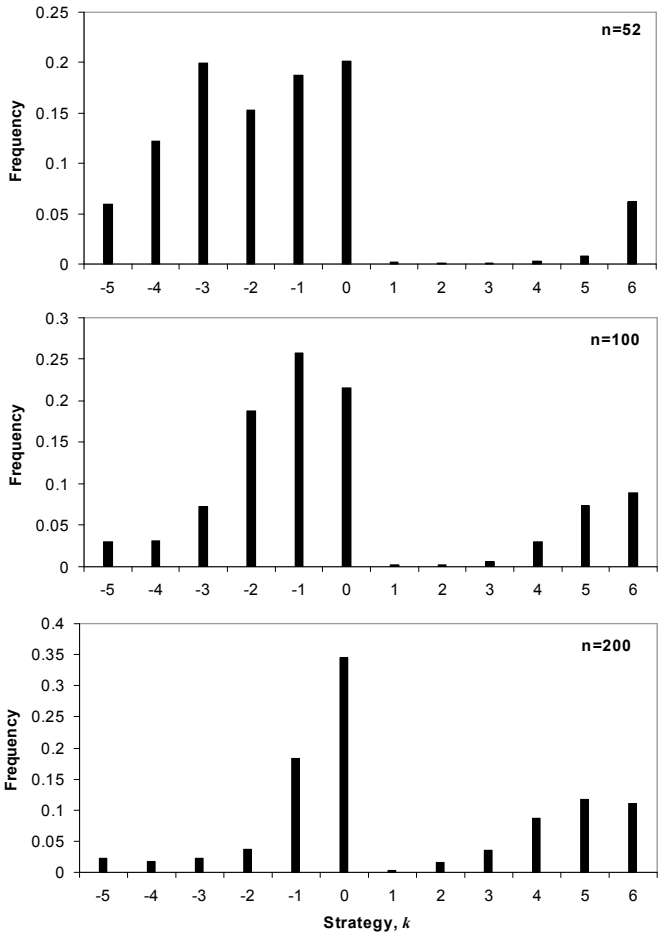

Figure 4. Shifting of proportion of voluntary agents with varying population sizes (n). Each simulation has clique size s=4.

New generations of agents copies neither the reputation score of their parents nor their parental group structure. They copy the strategy of their parents subject to an error rate as mentioned above. At the beginning of each generation, players are grouped to unique cliques of the same size. The game is played for many generations to subject the population to selective pressure. We say that cooperation is

established if the average winning strategy (k) for all individuals at the end of the game is less than or equal zero. We find that under our framework, cooperation evolves and is sustained even in larger populations after the game is played for many generations (*c.f.*, Figure 4). The effect of the initial clique size (of "core developers"), however, is less discernible in larger populations (see the simulation result for n=200 in Figure 4).

Figure 5 shows the result of the simulation for a population of 100 individuals for varying initial clique sizes. With more initial acquaintances, voluntary actions seem to emerge and be sustained faster. When information is scarce, in the absence of a dynamically evolving network of acquaintances, cooperators (players with $k \leq 0$) stand less of a chance against defectors. As soon as we allow for channels of information to evolve by letting individuals make new acquaintances as they interact, the likelihood of dissemination of information about players' reputation is increased, thereby increasing the likelihood for discriminators to be able to discriminate rightfully against exploiters, as well as in favor of cooperators. This is why voluntary actions and cooperators can evolve under this scenario despite the obvious fact that discriminators are the only ones who can make use of information. The emergence of voluntary actions in our open source development model is then a consequence of informed discrimination.

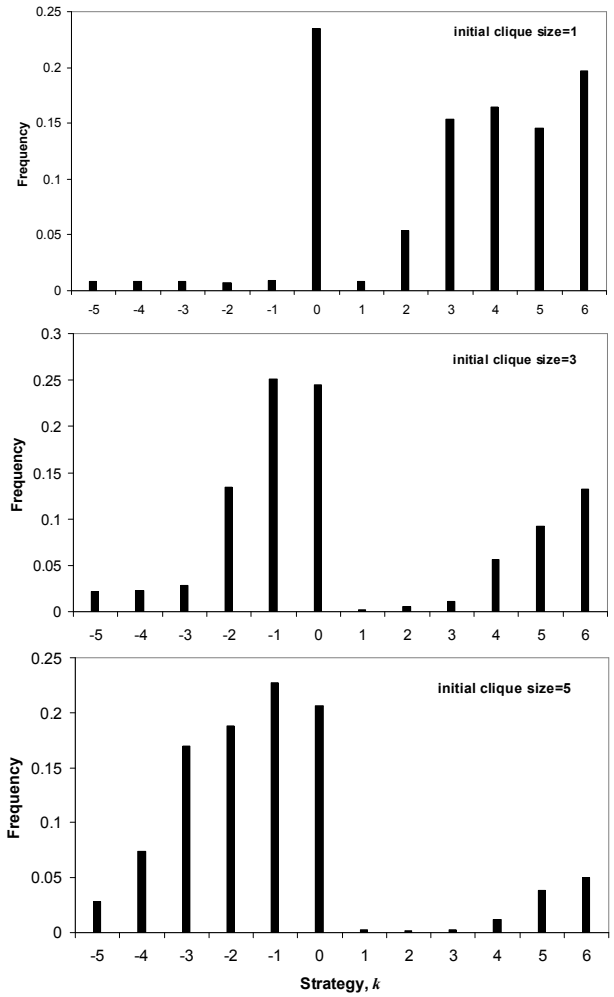

Figure 5. Emergence of cooperation with varying initial clique size. Each simulation has population size n=100.

Discussions

By giving software away for free, open source developers are surely not irrational, nor are they purely altruistic for the good of humankind. We have explained the seeming altruism of open source developers in terms of their concerns for their reputation and group identifications. In addition, we have developed a model for illustrating how voluntarism in open source projects can emerge among rational, self-interested agents. In our model, developers' ability to process complex social information by way of reputation propagation through acquaintance networks has played a critical role in the emergence of

voluntary actions within open source projects. The topology of the underlying social network is allowed to be dynamically modified as members change their interaction patterns.

In addition to simulation results, we have also analytically derived the conditions for the emergence of voluntary actions based on reputation inference through acquaintance sets. Our results suggest this emergence depends on the implicit benefit received by developers (Equation 8), which must out-weight the cost of trusting and helping a random feature (or bug-fix) requester.

Observations

Comparing the outcome of our simulations to field observations in the open source world, we discern a few interesting patterns. We observe a noticeable impact of generations (t) on the emergence of voluntary actions. A generation is modeling the equivalent of an open source project's version release. As opposed to closed source software, open source software releases new versions often. In fact, *"release early and release often"* has become an unsaid motto for the open source philosophy (Raymond, 1999). This strategy is seen at odds with traditional methods of development where release cycles are often long and few in between. For open source projects such as Gaim, we observe releases of over eighty versions over a period of five years (Sourceforge, 2004). This trend is common for many projects. From our results, we can hypothesize that a higher number of generations gives members of the community opportunities at re-aligning their strategies to either imitate successful developers, or to form new groups. As the number of generations increases, we see simulation results which indicate that voluntary actions are more likely to emerge.

With respect to the influence of population size on the strategy of cooperation, we find that with most successful projects such as the Mozilla browser or the Apache web server, the relative size of the core developers is relatively small compared to the overall community. For example, the Mozilla project has approximately 25 core developers (clique size) and over 400 patch submitters (Dotzler, 2002). Similarly, Apache has about 25 core developers, and over 430 patch submitters. Combined with our results, these observations on successful open source projects seem to indicate that by maintaining a

smaller population of core developers interacting with large user population, these projects are able to sustain the developers' voluntary tendency.

Clique size relates the size of core developers in a given open source project. Large clique sizes result in faster emergence of volunteering among the developers. Continuing with the example of Gaim, it was a project initiated by one developer and has since grown to a core developer group size of twelve over a period of five years. Gaim is also rated as being one of the top SourceForge open source projects in terms of the activity (93 percentile or above since April 2000) on their mailing list and code repository. This is evidence of a high degree of cooperation and socialization in the Gaim project community. Similarly, in case of PHPmyAdmin, the project began with one developer in 1998, and grew to a team of seven core developers. Activity for the PHPmyAdmin project has been rated in the 97th percentile or higher since 2001 (Sourceforge, 2004). These observations lend support to our results on how social groups (cliques) could lead to the emergence of voluntarism in the successful open source projects. By studying these top ten projects on SourceForge today, we find that most of the top ten projects have between three and twenty developers. These core developer sizes are extremely small compared to the size of the community (population) itself. We postulate that only by keeping clique sizes to a smaller size, open source projects maintain the voluntary nature of developer participation.

Conclusion

This model and analyses presented here challenge the common belief that developers are altruistic for their participation in open source project (Raymond, 1999). We have developed a multi-agents model of open source development focusing on how reputation and social groups influence the emergence of voluntarism for open source developers. We have also analytically derived conditions for how voluntary actions among rational, self-interested developers could emerge. Our results seem to correlate well with field observations on the less lofty motivators for open source developers – in particular reputation, and group involvement (von Hippel and von Krogh, 2003; Lerner and Tirole, 2000; among others). This

paper has illustrated how such implicit and indirect incentives as reputation gain and group identification can have powerful influence on motivating voluntary actions in software development.

Clearly, our results only paved a beginning step toward a more realistic model of the open source development process. Nevertheless, by explicitly modeling reputation and group involvement, our model permits a quantitative discussion on how important different model parameters are in sustaining voluntary actions in open source projects. Future work is needed to extend our model to cover overlapping generations for projects of different lengths and group sizes, agents with different types of implicit and delayed payoffs, varying levels of coordination between and across projects, more sophisticated mechanisms for reputation updating, *etc*. We also would like to study how core developers' size could influence the emergence and decay of voluntarism in open source projects, thus determining the success rates for these projects.

References

Allen, N. J., J. P. Rushton (1983) "Personality Characteristics of Community Mental Health Volunteers: a Review," *Journal of Voluntary Action Research, 12*, pp. 36-49.

Andreoni, J. (1995) "Cooperation in Public-Goods Experiments: Kindness or Confusion," *The American Economic Review, 85* (4), pp. 891-904.

Axelrod, R. (1984) *The Evolution of Cooperation*. New York: Basic Books.

Axelrod, R, W. D. Hamilton (1981) "The Evolution of Cooperation." *Science, 211*, 1390.

Batson, C. D. (1987) "Prosocial Motivation: Is it ever Altruistic?" in L. Berkowitz (ed.) *Advances in Experimental Social Psychology, 20*, New York: Academic Press, pp. 65-122.

Berkowitz, L., L. R. Daniels (1964) "Affecting the Salience of the Social Responsibility Norm: Effects of Past Help on the Response to Dependency Relationships," *Journal of Abnormal and Social Psychology, 68*, pp. 275-281.

Bergquist, M., J. Ljungberg (2001) "The power of gifts: Organizing social relationships in open source community", *Information Systems Journal, 11,* pp 305-320.

Busch, M. L., E. R. Reinhardt (1993) "Nice Strategies in a World of Relative Gains," *Journal of Conflict Resolution, 37* (September), pp. 427-445.

Darley, J., B. Latane (1968) "Bystander Intervention in Emergencies: Diffusion of Responsibility," *Journal of Personality and Social Psychology, 10*, pp. 215-221.

Diekmann, A. (1985) "The Volunteer Dilemma," *The Journal of Conflict Resolution, 29* (4), pp. 605-610.

Dotzler, A. (2003) "Getting involved with Mozilla: The people, the tools, the process", LUGoD lecture series, Davis CA. http://lugod.org/meetings/past/2003.03.04.php

Hamilton, W. D., (1963). "The Evolution of Altruistic Behavior." *American Naturalist. 97*: 354-356.

Hamilton, W. D., (1964). "The Ggenetical Evolution of Social Behavior." *J. Theor. Biol. 7*: 1-16.

Hardin, R. (1982) *Collective Action.* Baltimore, MD: Johns Hopkins University Press.

von Hippel, E., G. von Krogh (2003) "Open Source Software and the 'Private-Collective' Innovation Model: Issues for Organization Science," *Organization Science, 14* (2), pp. 209-223.

Hoffman, M. L. (1981) "Is Altruism Part of Human Nature?" *Journal of Personality and Social Psychology, 40,* pp. 121-137.

Katz, D. (1964) "The Motivational Basis of Organizational Behavior," *Behavioral Science, 9*, pp. 131-146.

Kessler, R. C. (1975) "A Descriptive Model of Emergence On-call Blood Donation," *Journal of Voluntary Action Research, 4*, pp. 159-171.

Kramer, R. M. (1993) "Cooperation and Organizational Identification," in J. K. Murnighan (ed.) *Social Psychology in Organizations: Advances in Theory and Research*, Englewood Cliffs, NJ: Prentice Hall, pp. 244-268.

Von Krogh, G. (2002) "The Communal Resource and Information Systems," *Journal of Strategic Information Systems, 11* (2), pp. 85-107.

Latane, B., J. M. Darley (1970) *The Unresponsive Bystander: Why Doesn't He Help?* New York: Appleton-Century-Crofts.

Lerner, J., J. Tirole (2000) "The Simple Economics of Open Source," NBER Working Paper No. w7600. http://www.nber.org/papers/w7600.

Lichbach, M. I. (1996) *The Coopreator's Dilemma*, Ann Arbor, MI: The University of Michigan Press.

Mockus, A., E. Fielding, and J Hersleb, (2000) "A Case Study of Open Source Software Development: The Apache server", *Proceedings of International Conference on Software Engineering (ICSE)*, Limerick, Ireland

Mohtashemi, M., L. Mui (2003) "Evolution of Indirect Reciprocity by Social Information: the Role of Trust and Reputation in Evolution of Altruism," *Journal of Theoretic Biology, 223* (4), pp. 523-531.

Mui, L. (2003) *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*, Ph.D. Dissertation, EECS, Massachusetts Institute of Technology (MIT).

Murnighan, J. K., J. W. Kim, A. R. Metzger (1993) "The Volunteer Dilemma," *Administrative Science Quarterly, 38* (4), pp. 515-538.

Nowak, M. A., K. Sigmund (1998) "Evolution of Indirect Reciprocity by Image Scoring," *Nature, 393:*6685, pp. 573-577.

Olson, M. (1965) *The Logic of Collective Action: Public Goods and the Theoryof Groups.* Cambridge, MA: Harvard University Press.

Ostrom, E. (2000) "Collective Action and the Evolution of Social Norms," *Journal of Economic Perspective, 14* (3), pp. 137-158.

Rapoport, A. (1998) "Experiments with N-Person Social Traps I: Prisoner's Dilemma, Weak Prisoner's Dilemma, Volunteer's Dilemma, and Largest Number," *The Journal of Conflict Resolution, 32* (3), p. 457-472.

Raymond, E. (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sabastopol, CA: O'Reilly.

Regan, D. T., J. Totten (1975) "Empathy and Attribution: Turning Observers into Actors," *Journal of Personality and Social Psychology, 32*, pp. 850-856.

Sharp, E. B. (1978) "Citizen Organizations on Policing Issues and Crime Prevention: Incentives for Participation," *Journal of Voluntary Action Research, 7*, pp. 45-58.

Smith, D. H. (1983) "Altruism, Volunteers, and Volunteerism," *Journal of Voluntary Action Research, 12*, pp. 21-36.

Sourceforge.net (2004) *"The world's largest Open Source software development website"*, Available on the web at http://sourceforge.net

Stinson, T. F., J. M. Stam (1976) "Toward an Economic Model of Voluntarism: the Case of Participation in Local Government," *Journal of Voluntary Action Research, 5*, pp. 52-60.

Trivers, R. L. (1971) "The Evolution of Reciprocal Altruism," *Quarterly Review of Biology, 46,* pp. 35-57.

Tullock, G. (1995) "Comment: Rationality and Revolution," *Rationality and Society, 7* (January), pp. 116-120.

de Wall, F. B. M., L. M. Luttrell (1988) "Mechanisms of Social Reciprocity in Three Primate Species: Symmetrical Relationship Characteristics or Cognition?" *Ethology and Sociobiology, 9*, pp. 101-118.

G. C. Williams (1971). *Group Selection.* Aldine-Atherton, Chicago.

D. S. Wilson, E. Sober (1994). Reintroducing group selection to the human behavioral sciences. *Behavior and Brain Science, 17*: 585-654.