Contribution in the Open Source Community

Disha Al Baqui

December 9, 2005

Abstract

Open Source software licenses permit the free sharing and improvement of Open Source software source code for which the community of Open Source software has flourished. However, these communities suffer from a lack of contribution and a profusion of "Free-Riding". Different innovation models and game theoretic principles can thus, be applied to view this phenomenon more logically and even be solved.

1 Introduction

When we think of innovations we imagine employees at large firms, academics at universities, or scientists at research institutions building novel goods. We imagine an average user to consume. However, in many fields of research and industry, consumers have also developed and innovated products. The Open Source community is an example of such a social organization where developers, who are users, come together to innovate new software.

There are two distinct facets of this community that are unique: *open product design*, which refers to users being able to modify software; and *open communication*, which refers to making information and software easily available and accessible [12].

The Open Source software (OSS) community of users are loosely affiliated by common interests: charecterized by voluntary participation, relatively free flow of information, and less hierarchical control and coordination than in firms. However, it does have an inherent social structure for which it is called a community and not a network and the means to share the software innovated is through discretionary databases.

2 Discretionary Databases

Shared databases are used in many organizational communication systems [6].A shared database is *discretionary* if users contribute to the database voluntarily

[7]. Information is *discretionary* if it is initially under the control of one organizational member, who can choose whether or not to make it available to others. There are several problems or social dilemmas, situations that pit the interests of the collective (e.g., group, organization) against self-centered interests of its members, that arise from discretionary databases [6]. A significant social dilemma the OSS community suffers from is under-supply. One way to solve this problem is to make contribution mandatory; however, that can be very difficult and sometimes impossible to achieve. Also, if users are rewarded for the quantity of their contributions, then a negative effect of lesser quality software will be encouraged, degraing the quality of the database. Other dilemmas include information hoarding and social loafing in teams as well as inter-organizational communication failures [6].

3 Open Source software development

An Open Source software is software for which the source code is "open [which] doesn't just mean access to the source code ... [but] must allow distribution in source code as well as compiled form". OSS is under an Open Source license which "... must allow modifications and derived works [to] ... be distributed under the same terms as the license of the original software." [9]. Open Source licenses allow for remodification and redistribution of the software without having to pay the original author [14]. OSS development projects are maintained by Internet-based communities of software developers who voluntarily collaborate to develop software that they or their organizations need [5].

In 1998, Eric S. Raymond, Bruce Perens and partners decided to start the OSS movement which is an offshoot of Richard Stallman's Free software movement ("... give freedom to computer users by replacing software which has restrictive licensing terms with free software") [13]. The two movements have very subtle differences but have large followings who argue on the side of each. Eric S. Raymond, or more popularly known as ESR in the hacker community, is a well known programmer who is currently a spokesperson for the OSS movement. His widely acclaimed book *The Cathedral and the Bazaar* is popular amongst many programmers. In his book, he compares the creation of proprietary software (software that has restrictions on using and copying it by means of software licenses, copyright and patent laws) to that of building a cathedral where "individual wizards or small bands of mages work in splendid isolation, with no beta to be released before its time" [10], whereas OSS development is a Bazaar model where users and developers contribute and consume with different "agendas and approaches".

It appears that the objectives, practices, and technical regimes of OSS developers and software engineers are different. Well-known examples of OSS having many users are the GNU/Linux computer operating system, Apache server software, and the Perl programming language. OSS development seems to entail shorter development times and produces higher quality systems incuring very low costs compared to traditional Software Engineering techniques. There is

very little known about how members in the OSS community develop their software or what organizational contexts are necessary for their success: but it is claimed that the development of OSS is faster, cheaper and better than traditional Software Engineering practices [11]. We shall see one aspect of this community which entails contribution within the discretionary databases by the simple act of participation.

3.1 Contribution

Eric Raymond reports, in *The Cathedral and the Bazaar*, that participants in OSS development have three basic motives for contribution. First, they may directly benefit from the software and software improvements they develop, because they have a use for them. Second, they may enjoy the work itself. Third, they may gain an enhanced reputation in the eyes of peers from making high-quality contributions to an open source project[10]. Some researchers propose two signaling incentives for contributors consisting of immediate payoffs and delayed payoffs. For the contributor an immediate payoff consists of the improvement in the software developed but also an immediate cost of the time it took to complete the improvement depending on how enjoyable the programmer finds the task is also incurred. The delayed payoff is a career concern and an ego gratification incentive, stemming from a desire for peer recognition. Other researchers list motives to contribute, which includes the beneficial effect of enhancements to ones reputation. A second potential motivator he sees is expectations of reciprocity, a mutual exchange of goods and privileges. The third motivator posited by Kollock is that the act of contributing can have a positive effect on contributors sense of efficacy - a sense that they have some effect on the environment. Finally, he notes that contributors may be motivated by their attachment or commitment to a particular open source project or group [8].

Researchers argue that the rates and effectiveness of discretionary information contribution amongst members in a community will tend to decrease as: (1) participation costs increase, (2) the size of the overall group increases, (3) lower value of information to participants and (4) greater asymmetries in information values and benefits across participants [8].

3.2 Innovation models

OSS falls in the "community-action" model, where each user exchanges a bit of effort or resource in return for benefiting from some collective provision, it is important that users contribute to the community . The signal difference is that someone can cheat in the large-number exchange by "free-riding" on the contributions of others. Contribution within the OSS community is therefore a very large problem because more often a large number of people free-ride by consuming yet not contributing. Even with a large community of users, the effect of free-riding is grave for the rest.[4]

Each and every user has a positive incentive to free-ride. A user's contribution might add substantially to the shared database, but his personal reward of

his own contribution from the database alone is very small.Therefore, every user has an incentive not to contribute and free-ride on the contributions of others which causes the database to be sparse which causes no "riding" to be possible. So the benefit to each user is larger if *everyone* contributes, than having no one contribute. Yet, free-riding still occurs and several solutions or models have been proposed to eliminate this phenomenon from the community.

In the "private-investment" model, innovators or programmers are supported by private investments or contributions and these provide private returns, society thus grants certain private rights to these innovations, for eg. through different forms of intellectual property law mechanisms. However, this model deviates from OSS development as the users of OSS are developers themselves and they would be required to *not* share their software.

In the "collective-action" model, goods are all public and any user who consumes it can not withold sharing the good with others, thus making the contributors relenquish control over the innovation or software, allowing a free flow of knowledge and information. This model places a great deal of emphasis on the importance of recruiting and motivating participants for a successful project in order to increase the attractiveness of contributing. Selective incentives are availabale for members within small groups, as social relations and benefits within this model is obvious, which is a major attraction of this model. However, successful OSS projects do not follow the methodology outlined by the "collective-action" model.

In the OSS community, goal statements for projects vary in detail and depth and as such no active recruitments are done for projects, and even large groups are successful in their endeavors. There are also no moral pressures exercised on any user and thus free-riders benefit on equal terms as contributors do. Thus a "private-collective" model for innovation illustrates the OSS development with the pros of one model cancelling out the cons of the other. This is where we find the solution to the "Free-Rider's" problem, by providing selective incentives to those who contribute.[5]

3.3 Prisoner's Dilemma

The "Free-Rider's" problem has also been modeled as the famous "Prisoner's Dilemma" game. The original game is as follows.

	Cooperate	Free-ride
Cooperate	(gain-gain)=(3,3)	(lose-gain)=(0,5)
Free-ride	(gain-lose)=(5,0)	(lose-lose)=(0,0)

Table 1: The Prisoner's Dilemma

Two suspects A, B are arrested by the police. The police have insufficient evidence for a conviction, and having separated both prisoners, visit each of them and offer the same deal: if one testifies for

the prosecution (turns King's Evidence) against the other and the other remains silent, the silent accomplice receives the full 10-year sentence and the betrayer goes free. If both stay silent, the police can only give both prisoners 6 months for a minor charge. If both betray each other, they receive a 2-year sentence each.

It is only when OSS community members are forced into a prisoner's dilemma that emergence of the database is threatened. The minute you cannot trust your back to the other members is the minute the contribution by members in to the database stops generating a value greater than the sum of its parts. What must a rational member of the community than do? This can be analysed from a game-theoretic point of view by finding the Nash Equilibrium of the game defined above.

A Nash Equilibrium of a game is a set of agent strategies in which any single agent cannot gain by unilateral deviation (that is, considering every other agents strategy as fixed, every agent is playing their best response strategy) [7]. The prisoner's dilemma has a pure-strategy Nash Equilibrium, where all members should follow a strategy with probability 1, which is to always free-ride. However, during unknown numbers of repeated interactions within the community, this strategy causes users to be unhappier than if they were to cooperate all-along [2].

Robert Axelrod, in his book "The evolution of co-operation", discovered that when these encounters were repeated over a long period of time with many players, each with different strategies, "greedy" strategies tended to do very poorly in the long run while more "altruistic" strategies did better.
Nice: The most important condition is that the strategy must be "nice", that is, it will not defect before its opponent does.
Retaliating: However, Axelrod contended, the successful strategy must not be a blind optimist. It must always retaliate.
Forgiving: Another quality of successful strategies is that they must be forgiving. Though they will retaliate, they will once again fall back to cooperating if the opponent does not continue to defect.
Non-envious: The last quality is being non-envious, that is not striving to score more than the opponent.

Therefore, Axelrod reached the Utopian-sounding conclusion that selfish individuals for their own selfish good will tend to be nice and forgiving and non-envious. [15].

Ofcourse there are other strategies, the most famous of which remains Tit-for-Tat(TFT), where an agent using this strategy will initially cooperate, then respond in kind to a previous opponent's action. If the opponent previously was cooperative, the agent is cooperative. If not, the agent is not. In the OSS, a user is not always able to selectively interact in this fashion with other members of the community. However, it is proved that Generous TFT, a "Nice" version of this strategy (even third defections are forgiven) is sometimes better [2].

4 Related Work

Currently some work is being done which studies how the OSS community operates, why the development of such software is successful and how these methods can be reflected in current software engineering practices. Many game theorists are interested in the equilibrium which exists within the OSS community and are developing several theories borrowed from mechanism design and auction theory as well to formalize the model of OSS more accurately.[3, 1]

5 Conclusion

People must be free to use, modify and distribute software. The Open Source community allows for this to happen. There are many successful projects which have been launched because of a single programmer's dream or necessity and have now grown into multi-developer software systems. However, the OSS community suffers from the lack of contribution. Several models and solutions for the problem of "Free-riding" have been presented, and these features when implemented are sure to negate this issue.

References

[1] Stefan Behringer. The provision of a public good with a direct provision technology and large number of agents. In *Free/Open Source Research Community*. MIT Sloan School of Management, January 2005.

[2] Daniel Eckert, Stefan Koch, and Johann Mitlohner. Using the iterated prisoner's dilemma for explaining the evolution of cooperation in open source communities. In *Proceedings of the First International Conference on Open Source Systems*, July 2005.

[3] J. Feller and B. Fitzgerald. A framework analysis of the open source software development paradigm. In *Proceedings of the International Conference on Information Systems*, December 2001.

[4] Russell Hardin. The free rider problem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, Center for the Study of Language and Information, Summer 2003.

[5] Eric Von Hippel and Georg Von Krogh. Open source software and the private-collective innovation model. *Organization Science*, 14(2), March-April 2003.

[6] Michael Kalman, Janet Fulk, Peter Monge, and Rebecca Heino. Motivations to resolve communication dilemmas in database-mediated collaboration. *Communication Research*, 29(2):125–154, April 2002.

[7] Mike Klaas. Over contribution in discretionary databases. In *Second Annual workshop in the Economics of Peer-to-Peer Systems*, 2004.

[8] Karim Lakhani and Eric von Hippel. Executive summary: How open source software works: 'free' user-to-user assistance. Working paper, May 2000.

[9] Bruce Perens. The open source definition, 2005. [Online; accessed 9-December-2005].

[10] Eric Steven Raymond. The cathedral and the bazaar, September 2000. [Online; accessed 9-December-2005].

[11] Walt Scacchi. When is free/open source software development *Faster, Better, and Cheaper* than software engineering? chapter, Institute of Software Research, University of California, Irvine, June 2004. supported by grants from the National Science Foundation.

[12] Sonali Shah. Open beyond software. *Open Sources 2*, 2005-forthcoming.

[13] Wikipedia. Free software movement — wikipedia, the free encyclopedia, 2005. [Online; accessed 9-December-2005].

[14] Wikipedia. Open-source license — wikipedia, the free encyclopedia, 2005. [Online; accessed 9-December-2005].

[15] Wikipedia. Prisoner's dilemma — wikipedia, the free encyclopedia, 2005. [Online; accessed 9-December-2005].