Motivation of Open Source Developers: Do License Type and Status Hierarchy Matter?

Mark R. Allyn, Montclair State University, USA
Ram B. Misra, Montclair State University, USA

ABSTRACT

The motivational drivers of open source software developers have been researched by various investigators since about 2000. This work shows that developers are motivated by different extrinsic and intrinsic drivers, among them community aspirations, reciprocity and fairness, creative impulses, and monetary and career ambitions. There has been some work done in studying whether the profile of developer motivations is constant across open source projects or is sensitive to project organizational design. Among the many factors that could influence the mix of motives of OS developers is the license under which the work is performed. Licenses range in openness between those such as the GNU GPL that severely restrict the freedom of developers to mingle their OS code with proprietary code to those such as BSD licenses which allow programmers much greater latitude in integrating open source code with proprietary code. In addition to formal rules, meritocracies emerge to reward effort and performance, and also to direct, coordinate, and control other participants. The authors discuss these variables and how they may be related to motivations.

Keywords: Code, Hierarchy, Motivation, Open Source Licenses, Open Source Software

INTRODUCTION

Open source software projects have become ubiquitous. Sourceforge.net open currently hosts more than 200,000 projects, from small to large. In the open source environment, many programmers work without direct monetary compensation and yet they create high quality products that result from their willingness to share their knowledge (code and algorithms) with other programmers. Setting aside the problem of how their efforts are directed and coordinated, these open source projects present a fascinating instance of knowledge sharing without compensation and with only limited safeguards on exploitation by renegades who can hijack the code for their own ends. Sharing knowledge and code is costly for the contributor, both in terms of current effort and foregone opportunities. What is the reward?

DOI: 10.4018/jossp.2009100104
INCENTIVE SYSTEMS

Beginning with Hars and Ou (2002), many investigators have asked this question and proposed various answers supported by differing amounts of empirical evidence. The answers have fallen into two broad categories, and it appears from the evidence that no single driver is the impetus for contributing, but a mix of motives is present in all projects that have been studied. In an important survey, Krishnamurthy (2006) summarizes the major threads of the panoply of motivators, and makes the important observation that the empirical evidence indicates that both extrinsic and intrinsic motives can coexist in an OS context.

A. Extrinsic Motivation

Lerner and Tirole (2000) group extrinsic motivations into two clusters, immediate payoff and delayed payoff. The immediate payoffs may come simply because the code must be written to solve an existing problem. The delayed reward payoffs derive from ego gratification, applause from an appreciative and sophisticated audience, or career incentives. Career concerns stem from future anticipated job opportunities.

As distinct from penalties arising from infringing licenses, there are norms that punish those who defect from (for example, “forking”) the community and rules that determine status and benefits for making substantial contributions to the product. One norm that is mentioned often is “generalized reciprocity”, the willingness to share with no expectation of immediate benefits. Sharing widely and freely what one knows can be a powerful signaling device that shows others, in particular employers, one’s level of competence in a domain in which it may be difficult and costly for the employer to discern skill levels.

Hann, Roberts, Slaughter, and Fielding (2002) show that employer compensation is linked with rank in an OS project, but contributions to the project are not correlated with pay. As noted by these authors, leaders provide valuable services to the project in terms of “vision”, architecture decisions, and enforcing the rules of project. Thus, a hierarchy rather naturally develops in these projects if they are to sustain themselves and higher employer pay is a compensation for the leadership behaviors of the leaders.

It is a common illusion (see, for instance, Benkler, 2002) that open communities are flat, decentralized “full mesh” peer-to-peer networks (“bazaars”, as Raymond tells it). In fact, there is strong evidence that a substantial and powerful hierarchy emerges, typically starting with a founder such as Linus Torvalds (1999), who controls the final release of each point version of the Linux program. In addition to the founder, additional bureaucratic levels (“user”, or “bug fixer”, “programmers”, administrators, (Lattemann & Stieglitz, 2005) appear in mature open source environments. In one of the few social sciences investigations, Stewart (2005) examined an open source community of about 5000 programmers and found a very distinct four-level status hierarchy.

Leadership is vital to the survival of an open software environment (Lerner & Tirole, 2000). The leader is responsible for providing what Lerner & Tirole call “vision”, do the job of dividing the project into modules, attract other programmers, prevent fragmentation (“forking”) of the project into renegade camps. Hierarchy in an OS project provides this leadership.

Open systems projects employ self-selection as the method of assigning tasks to workers. On the basis of the information about a particular module a worker makes voluntary contributions to the module. However, although individual programmers make decisions about which tasks to undertake, the hierarchy determines which “patches” will be included in the working version of the code.

There are many ways programmers can profit directly (Koenig, 2004; Watson et al., 2008) from creating an open source project or contributing to one. Some firms and individuals collect revenue through complementary services such as consulting, selling subscriptions for maintenance and upgrades, hosting applications, and so forth. Firms that have com-
Related Content

Developing a Dynamic and Responsive Online Learning Environment: A Case Study of a Large Australian University
[www.igi-global.com/article/developing-dynamic-responsive-online-learning/41952?camid=4v1a](www.igi-global.com/article/developing-dynamic-responsive-online-learning/41952?camid=4v1a)

An Approach to Mitigate Malware Attacks Using Netfilter's Hybrid Frame in Firewall Security
[www.igi-global.com/article/an-approach-to-mitigate-malware-attacks-using-netfilters-hybrid-frame-in-firewall-security/206886?camid=4v1a](www.igi-global.com/article/an-approach-to-mitigate-malware-attacks-using-netfilters-hybrid-frame-in-firewall-security/206886?camid=4v1a)
A Quantitative Study of the Adoption of Design Patterns by Open Source Software Developers
[www.igi-global.com/chapter/quantitative-study-adoption-design-patterns/18722?camid=4v1a](www.igi-global.com/chapter/quantitative-study-adoption-design-patterns/18722?camid=4v1a)

To Fork or Not to Fork: Fork Motivations in SourceForge Projects
[www.igi-global.com/article/fork-not-fork/68147?camid=4v1a](www.igi-global.com/article/fork-not-fork/68147?camid=4v1a)