Why is Open Source Software Viable? A Study of Intrinsic Motivation, Personal Needs and Future Returns

Alexander Hars
University of Southern California, hars@bus.usc.edu

Shaosong Ou
University of Southern California, sou@bus.usc.edu

Why Is Open Source Software Viable? - A Study of Intrinsic Motivation, Personal Needs, and Future Returns

Alexander Hars
hars@bus.usc.edu

Shaosong Ou
sou@bus.usc.edu

Marshall School of Business, Univ. of Southern California

Abstract

Traditional business models for software development are currently being challenged by the phenomenon of open source software where communities of programmers leverage the Internet to develop free software without receiving any direct compensation. To understand the success and the prospects of open source software it is necessary to examine the motivation of the participants in open source projects. This paper presents a theoretical model to study the three main factors leading to participation in open source projects: intrinsic motivation, personal need and expectation of future returns. Implications of the model are derived and their significance discussed. We conclude that open source software will be an enduring alternative to traditional software development and that it is possible to combine aspects of both development approaches.

Challenging established business models

In the past 30 years, software has become a major industry. Highly profitable software companies have emerged. All of these companies are based on the premise that software is a proprietary good. Although its reproduction and distribution costs are nearly zero, they regard software primarily as a private good which is the property of the "owner".

This traditional business model is now being challenged by the phenomenon of open source software. In contrast to traditional software, open source software is provided to the public for free. The user does not only have the right to use and run the software; its source code is also provided free of charge and the user explicitly receives the right to make modifications to the source code and to distribute these modifications. Open source software is typically developed by teams of programmers that collaborate via the Internet. In addition, these programmers do not receive direct compensation for their work.

While this model of developing software which treats software essentially as a public good has a history that can be traced back into the 1960s (see Table 1), the phenomenon has received little attention from commercial developers or from academia until recently. The impact of these efforts seemed to be limited to what small bands of idealistic programmers can achieve. However, in the last years one open source project, Linux has matured sufficiently to reach the spotlight. Linux was written by Finnish student Linus Torvalds in 1991. He then released his source code to the public and received continuous support and updates via the Internet from programmers all over the world (Bollinger and Beckman 1999). Linux has rapidly increased its market share and drawn converts in many business. It is now actively supported by many leading software vendors including Corel, Oracle, SAP, IBM, etc. (e.g. Seltzer 1999). The number of installations is estimated to be over 7.5 Million (Comerford, 1999) and the stock market has placed high values on companies in the Linux space.

Table 1. Open Source Timeline (Gonzalez Barahona, Heras Quiros and Bollinger 1999, Seltzer 1999, Comerford 1999)

Year	Event
1950s and 1960s	Software source code is distributed without restrictions in IBM and DEC user groups, ACM's Algorithms Section etc.
1969	Ken Thompson writes the first version of Unix. Its source code is distributed freely throughout the seventies.
1978	Donald Knuth (Stanford) publishes TEX as free software
1979	Following AT&T's announcement to commercialize UNIX, UC Berkeley begins with the creation of its own version of UNIX, BSD (Berkeley Software Distribution). Eric Allmann, a student at UC Berkely develops a program which routes messages between computers over ARPANET. It later evolves into Sendmail.
1983	Stallmann publishes GNU Manifesto calling for free software, and establishes Free Software Foundation.
1986	Larry Wall creates Perl (Practical Extraction and Report Language), a versatile programming language used for writing CGI (Common Gateway Interface) scripts.

1987	Developer Andrew Tanenbaum releases Minix, a version of Unix for the PC, Mac, Amiga, and Atari ST. It comes with complete source code.
1991	Linus Torvalds publishes version 0.02 of a new Unix variant that he calls Linux in a Minix newsgroup.
1993	FreeBSD 1.0 is released. Based on BSD Unix, FreeBSD includes networking, virtual memory, task switching, and large filenames. Ian Murdock creates a new linux distribution called Debian Linux.
1994	Marc Ewing forms Red Hat Linux. It quickly becomes the leading Linux distributor. Bryan Sparks founds Caldera with backing by former Novell CEO Ray Noorda.
1995	The Apache Group builds a new Web server, Apache, based on the National Center for Supercomputing Applications' (NCSA's) HTTPd 1.3 and a series of patch files. It has become the dominant HTTP server today.
1998	Netscape not only gives away Communicator 5.0 (Mozilla) but also releases its source code. Major software vendors, including Computer Associates, Corel, IBM, Informix, Interbase, Oracle, and Sybase, announce plans to port their products to Linux. Sun announces plans to release the source code for Java 2 to developers.
1999	Number of Linux users estimated at 7.5 Million.
2000	More software companies such as Novell and Real release versions of their products which run on Linux.

Many commercial software companies are currently assessing the impact of open source software on the software market: is it true, as many open source proponents suggest that all software should – and eventually will – be free? One of the key questions in this puzzle regards the sustainability of the open source phenomenon. It is not difficult to imagine that a few projects can get underway where enthusiastic programmers join forces to develop a major software for free. But will it be possible to reach the same momentum repeatedly for many different kinds of software? The answer hinges largely on the motivation that drives the programmers to participate in such a project. In this article, different sources of motivation and their implications for the sustainability and the strengths and weaknesses of open source efforts will be derived.

Sources of motivation

Although prior theoretical research has not directly addressed behavioral issues about open source software programmers, literatures from other disciplines do provide good starting points. Utility maximization theories (e.g., Varian 1999) and investment theories in economics (Becker 1962), for example, are readily available to explain some behaviors and motivations of these programmers. Psychological and motivational studies can also be utilized (Deci 1975). Within the literature on motivation, three core categories of motivations need to be distinguished: those who are intrinsically motivated to write programs, those who program for their personal and work-related needs and those who treat developing open source software as a form of investment and expect future returns. In the following, these three cases will be analyzed in detail:

Intrinsic motivation

The programmers who find programming interesting and attractive fall into this category. According to Deci (1975), human beings are born with the basic and undifferentiated need for feeling competent and self-determining. Not all human behaviors leads to extrinsic rewards (e.g., money, food, clothes etc.), but “intrinsically motivated behavior is the behavior that is motivated by a person’s need for feeling competent and self-determining in dealing with his environment”. The cognitive model of behavior has five elements, stimulus inputs, awareness of potential satisfaction, goals, goal-directed behavior, and rewards (Deci 1975). For these programmers, the existence of software with open source code and the internet are the stimulus inputs, which will invoke the programmer of his awareness of potential satisfaction, i.e., the satisfaction from feeling competent and self-determining by programming. Being intrinsically motivated, he will proceed to set up a goal of programming open source software to realize the satisfaction or rewards. The goal-directed behavior is spending time and effort in programming without receiving extrinsic rewards. The reward to his behavior is feeling competent and self-determining.

These programmers thus consider programming intrinsically motivating. The cost associated with programming (e.g., opportunity cost, physical effort, emotional anxiety) is compensated by the intrinsic rewards (feeling competent and self-determining), which is the underlying reason that sustains their behavior of programming. As long as the pay-off is greater than the cost, programming will continue. Programmers will reduce the effort spent in programming or stop programming when the rewards cannot cover the cost. This implication can be directly derived from basic economics principles. Assuming all agents are economically rational and behave to maximize their own utility levels, their behaviors are determined by the pursuit of rewards.

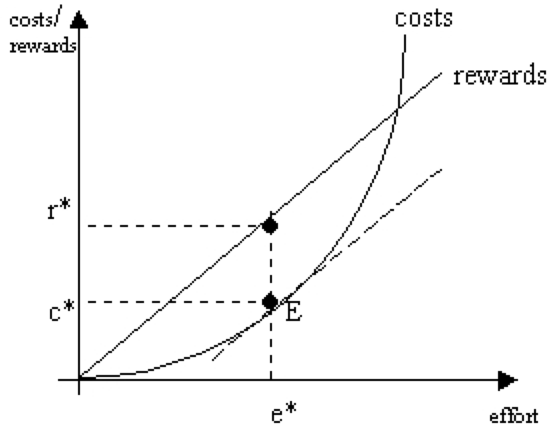
The intrinsic motivations associated with programming open source software may come from two sources, writing programs itself or increasing the welfare of others by programming. We label these groups as hobby pursuers and altruists.

1. Hobby pursuers
Open source software developers are frequently programming enthusiasts. They seem to develop software just for "fun". They love programming like other hobby pursuers love cooking or playing instruments. Their feelings for competent and self-determination can arise from a number of sources, e.g., manipulating computer's performance by writing codes and issuing commands, watching the computer behave in certain ways or appreciating their mastery of programming skills. If open source developers are mainly hobbyists, this has interesting implications for open source projects: The time that each programmer is able to spend on open source development will be limited; due to budget constraints, Among the programmers with the same amount of leisure time, those with low income are likely to spend less effort in programming than high-income ones. In addition hobby pursuing programmers may be difficult to coordinate as they often lack an externally identifiable motivation which allows to predict their behavior and interests.

2. Altruists
Another variant of intrinsic motivation is the urge to increase other people's welfare by writing useful programs. Developing free software and publicizing the source code can bring convenience to others and increase their utility and productivity levels. Serving people with one's effort in open source software is thus the source of feeling competent and self-determining, i.e., the source of intrinsic motivation. For an altruistic programmer, the better and the more he can serve, the stronger he feels competent and self-determining and the larger are the rewards he receives from programming. A crucial implication is that altruistic programmers prefer to develop software that reaches a large number of users and has high value to these users. In contrast to hobbyists, a team of altruists will be less at risk to become sidetracked in esoteric functionality. They can be expected to focus on that functionality that is most valuable to users. The hypothesis that open source programmers are motivated by altruism, however, has a significant flaw: One would expect that programmers would focus on software usability which is the key determinant of the value of software to an end-user. However, open source software is often characterized by cumbersome and difficult-to use interfaces.

According to the cognitive model of behaviors, rewards of intrinsically motivated activities are more sustainable than other activities because a person is in continual interaction with the environment and has a continual need for feelings of competence and self-determination. So when satisfaction is achieved a new awareness of potential satisfaction will arise which will energize a new sequence of cognitive behavior (Deci 1975). Based on this mechanism, we assume that this group of programmers receives constantly increasing rewards or satisfactions by programming, i.e., the relationship between rewards from programming and the amount of effort spent in programming is linear and positive. That is, a unit increase in effort will increase rewards by the same amount. The behavior also incurs costs. The costs may be associated with their physical or mental expenses, e.g., fatigue and anxiety, or they may include the opportunity cost caused by engaging in a non-profit activity. It is a reasonable move to assume that these costs are positively associated with the amount of their effort. Moreover, we can further assume that the marginal increase in the costs is higher when the effort spent in programming is increased. In other words, increasing programming hours from 10 to 15 hours per week incurs higher unit costs than increasing the hours from 5 to 10. As trading extrinsic rewards for intrinsic ones, extrinsic rewards become more valuable as they become relatively scarce. The same amount of sacrifice in extrinsic rewards therefore incurs extra loss. The relationships between effort and rewards/costs are represented in Figure 1.

Figure 1. Relationship between effort and rewards/costs



As shown in Figure 1, the reward function is linear, but the cost function is convex. Each programmer as a rational economic agent will try to determine the best amount of effort to spend in programming open source software to maximize his or her own utility level, which is the difference between rewards and costs. The utility-maximizing point is at E, where its marginal cost equal to marginal rewards. The corresponding effort level, rewards and costs are point e*, r* and c* respectively. The utility level that the programmer receives is (r*-c*), the difference between rewards and costs.

This model implies that intrinsically motivated programmers will spend certain amount of time and effort in programming as standard software programmers, even without receiving explicit compensation for it. This implication is compatible with the fact that open source software programmers program for free. These programmers will choose to spend e* amount of effort in programming, where their utility level is maximized at (r*-c*). This amount is jointly determined by opportunity costs, physical conditions, leisure time and income (hobby pursuers) and software functionality and popularity (altruists).

Personal Needs

Intrinsic motivation is not the only reason for participating in an open source project. As the history of prominent open source projects shows, many open source projects have been initiated for entirely different reasons. The programming language PERL was created by Larry Wall because he needed to generate web pages programmatically. He found it too cumbersome to write his programs in C and therefore developed simple routines that could be reused and combined. He later shared these routines with other programmers (O'Reilly 1999, p.194). The Apache Server was driven by a similar motivation. In 1995, a large number of web masters were using the NCSA HttpD web server. It had many problems which the web masters circumvented by programming their own patches. Quickly, a core group of web masters formed to share their patches. They rewrote the web server to include more patches and the Apache web server was born (Unknown 1998).

Both cases illustrate an additional motivation that is based on the personal need of a programmer (or a group of programmers) for specific functionality (Behlendorf 1999, p. 159). The existence of personal needs has important implications for open source projects. First, it shows that participants in open source projects may act rationally in their own self-interest, if they provide their software for free as long as long as selling software involves significant transaction costs. Second, it shows that there should be a limit to the amount of effort that a participant in open source project provides for free. The more complex a product becomes, the less its value depends on interactions with other modules of software, the more its contribution can be identified and communicated, the more likely it is that a programmer will sell his software rather than provide it for free. Some cases may already be cited. Eric Allmann, for example, the founder of Sendmail which is one of the most successful email server programs, has started a company which provides an add-on product to Sendmail which simplifies its configuration and administration. This is a larger module which is useful to most adopters of Sendmail and thus can be marketed effectively. The third implication of personal need may be the most important: it shows that the interests of the users and developers of software are often aligned: both are interested in improving the functionality; both are willing to invest in improvements. However, traditional software houses structure their licenses agreements in a way which prevents customers to invest in their software by making modifications, by tweaking and tuning it and by sharing these improvements with others. This would raise the value of a license to prospective buyers and thus increase the revenue stream or market position of the software vendor. However, because of the fear of piracy, software houses give up considerable potential investments which customers would be willing to make in the source code. From the perspective of leveraging a need in improved functionality, this category of motivation thus demonstrates a crucial oversight in the marketing and product evolution strategies of current software companies.

Expectation of future returns

A third category of motivation is based on the expectation that participation in an open source project will lead to future revenue. Participation thus is treated as an investment for which a payback can be expected at a later time (DiBona, Ockman & Stone, 1999, p. 13). The economics of such investments are well understood. The question that remains specific to the open source community is the nature of such rewards. Three types of investments models must be distinguished:

1. Commercialization of related products and services

One of the traditional strategies of generating business is to provide one product for free which then generates follow-up sales opportunities. This has been demonstrated in many industries. In the open source domain, companies such as Red Hat, Caldera, Corel, etc. are investing in the development of Linux to ensure that they can provide services like consulting, training, implementation and distribution etc. But it should be noted that participating in an open source project by a company that markets services around this project may be a double-edge sword. The company may be naturally biased against increasing the usability of the software because of the concerns of loosing service opportunities.

2. Investment in human capital

According to human capital theories (Becker 1962), human skills, capabilities and knowledge are special capital forms – human capital. Investment in human capital can take forms such as schooling, training and learning and is anchored with expected future returns (higher output, better performance, more profits etc.).

Open source software programmers who aim at improving their skill of writing programs are human capital investors – investors that invest in their own human capital stock. Open source software has large number of users, open source code and collaborating developers. Thus the community of the open source software developers provides an excellent environment to extend one's skills of writing programs.

3. Advertisement

Some programmers may regard working for open source software as an effective way to demonstrate their capability and skillfulness in programming. Claims of competence in programming can be well reinforced by the achievements in open source projects. Participating in open source projects therefore can be a good advertising channel to publicize one's skillfulness and capabilities. Advertisement is also associated with future returns.

The fact that every open source project leaves a record of the commitment of each participant has an important implication: The larger the contribution of an individual, the more likely it is that a commercial developer will recognize the value of the individual and the larger the incentive will become for this individual to apply his skills in a paid position. Thus the openness of open source projects may work to some against open source projects. It may help to lure the best programmers and most productive minds away from these projects into much more highly paid commercial development.

Summary

This article has shown that characterizations of open source developers as being motivated by intrinsic motivation and altruism may neglect other equally or even more important motivations of open source software developers which are rooted in self-interest. These motivations are the self-interest in the improvement of software functionality and the investment into marketing, growth and sales opportunities. The open source phenomenon thus may have an economically much stronger foundation than previously thought. The analysis of motivations also imply that commercial software companies have more alternatives in changing their marketing and product evolution strategies in reaction to the open source phenomenon. Software companies, for example, could find partially open approaches which accommodate their customers' needs to improve and extend standard software and at the same time increase the overall utility and market value of their software products. As a consequence, open source must be seen as a lasting, sustainable phenomenon. Rather than replacing commercial development, however, it will increase the variety of business models in the software industry.

References

Becker, G. S., "Investment in Human Capital: A Theoretical Analysis", *Journal of Political Economy* (70 supplement), 1962, pp. 9-49.

Behlendorf, B., "Open Source as a Business Strategy", in DiBona, Chr., Ockman, S. and Stone, T. (eds.) "Open Sources", Sebastopol, CA (O'Reilly), pp. 149-170.

Bollinger, T. and Beckman P. "Linux On The Move", *IEEE Software* (January/February), 1999, pp. 30-35.

Comerford, R. "The Path to Open-Source Systems", *IEEE Spectrum* (May), 1999, pp. 25-31.

Deci, E. *Intrinsic Motivation*, Plenum Press, New York, NY, 1975.

DiBona, Chr., Ockman, S. and Stone, T. (eds.) "Open Sources", Sebastopol, CA (O'Reilly).

Gonzalez Barahona, J., Heras Quiros, P. and Bollinger, T. "A Brief History of Free Software and Open Source", *IEEE Software* (January/February), 1999, pp. 32-33.

O'Reilly, T. "Hardwarde, Software and Infoware", in DiBona, Ockman and Stone (1999), pp. 189-196.

Seltzer, L. "Milestones in the Open-Source Movement", *PC Magazine* (March), 1999.

Unknown, "Linux and Apache", *Computer* (October), 1998, pp. 12.

Varian, H. *Intermediate Microeconomics A Modern Approach*, W. W. Norton & Company, New York, NY, 1999.