Attractiveness of Open Source Projects: A Path to Software Quality

Kay Nelson  
Southern Illinois University

Carlos Santos  
Southern Illinois University

Follow this and additional works at: http://aisel.aisnet.org/amcis2007

Recommended Citation
http://aisel.aisnet.org/amcis2007/490

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Attractiveness of Open Source Projects:
A path to software quality

Carlos Santos, Jr., M. Sc.
Research Associate – CAPES (Brazilian Government Agency)/Fulbright
Southern Illinois University
carlos81@siu.edu

Kay Nelson, Ph.D.
Professor, MIS
Southern Illinois University
ikay@siu.edu

Abstract
The Open Source Software movement is impacting society and organizations in significant ways. This impact can be observed not only economically but also on the way business processes like software development are performed within organizations. The success of open source software is attributed to its practices and organizational structure. Consequently, there is a trend in the corporate environment to copy and adapt some of these practices, such as releasing software source code to the community, giving up proprietary rights. This paper is an attempt to model what are the drivers for success of this specific practice-trend of releasing source code to the community by corporations. Thus, to understand (1) what the specific sponsor’s motivations to engage in projects intended to release source code are, thereby defining success of the practice, and (2) the conditions under which those projects are more likely to generate desirable outcomes become important to both researchers and practitioners. This paper pursues both of these topics, presenting propositions to empirically explain the numbers of contributions and contributors – attractiveness – a corporate project has observed and why. We assume that attractiveness leads to software quality, a condition desired by profit-oriented managers.

Key words: Open source, organizational practices, software development, interdependence, software modules.

Introduction
Open source software has impacted organizations in general and especially the IT industry over recent years. Organizations now have a ‘new company model’ to look at and get inspiration from. They also have the option to engage in open source communities, which have the potential to significantly change both the economic and social structures of the IS organization.

The low costs of communication through the Internet, the availability of people around the globe and their willingness to develop software to fulfill their own and other’s needs have created a scenario that has never existed before.
Volunteers are now developing software to be used in the corporate world.

Open source software is recognized as high quality due to their organizational structure and development practices. Consequently, corporations have tried to adapt and copy the open source software model in order to achieve better results. Of especial interest here is the business practice of releasing software source code to the community, thereby giving up of proprietary rights. A new social norm is emerging. As this trend increases, questions of how and why it occurs have become of interest to researchers and practitioners. Subsequently, judgments about whether and when this practice should or should not be used in the corporate environment have to lose subjectivity. This paper is a theoretical attempt to understand managers’ and/or sponsors’ motivations to engage in such practice. Furthermore, it is an attempt to find out which variables are good candidates for predictors of the number contributions (inputs) a project receives and the number of contributors it has, components of the attractiveness construct we developed.

The number of corporations engaged in open source communities can potentially change the entire social structure of the IS organization, and can lead to major shifts in corporate norms.

Attention has been given in the management literature to the open source volunteers’ motivations for engaging in such activities. The main findings are concentrated on issues like reputation, potential career advantages, social and political motives, status, and so on (Hertel, et. al., 2003).

Some of the open source software are considered to be as good as any other proprietary software. Actually, one of them is the web server market leader Apache (Netcraft, 2006). Thus, if one understands and is able to provide what attracts skillful volunteers out there to develop software, corporate opportunities might be realized. Consequently, a new business strategy (practice) has emerged from this setting: An attempt to attract skillful volunteers to develop and maintain software by releasing the source code. However, when such practice becomes a trend, as stated by West and O’Mahony (2005), competition for the limited population of skillful people willing to devote their time to first study and understand, and second to develop someone else’s source code is expected to increase. Consequently, the number of contributions expected to be received from adopting this practice is reduced, justifying careful analysis of the probability of applying corporate resources wisely.

The recognition of the open source movement impact in the IT industry has resulted in an attempt to bring it closer to the corporate world, requiring the application of resources in the form of investments, which brings risks along. This trend has many similarities with other business ones such as downsizing, outsourcing, TQM, and so forth, meaning that it might pay off or not, depending on the case and on the definition of success. Cases such as IBM of corporations funding open

1 As the case of 7zip, a software that compress archives and has better performance than its competitors, including proprietary software. See [http://www.7-zip.org/](http://www.7-zip.org/)

source developers are not rare anymore. Thus, collaboration between communities, industry, and academia has been discussed.

In this paper, the business practice is represented as an attempt to get help from the community on internal production processes, especially on ones related to software development. Normally, companies interested in receiving this type of help release software source code on-line. And, from that point on, it is assumed that companies expect to obtain inputs from volunteers to increase their software quality, expanding their relationship with potential and current customers. Examples of this phenomenon appear in different sectors such as in the American government, Sun Inc. and SugarCRM.

This paper main goal is to explore the influences of project/software characteristics such as interdependence (cohesion), project sponsor features such as reputation, and processes characteristics such as popularity of programming languages/Integrated Development Environment (IDE) on the achievement of the goal, assumed to be higher software quality, related to the organizational decision to release software source code, thereby giving up proprietary rights.

The questions to be answered are: (1) What motivates managers to adopt such practice? (2) Does the IDE choice influence the project’s appeal to the community? (3) Does the software level of interdependence (cohesion) influence the project’s appeal to the community? (4) Does sponsor’s reputation influence project’s appeal to the community?

This paper is organized as follows. In the next section, a description of the open source communities is provided. Theories of business practices and technologies adoption are discussed next and the first proposition (assumption) is presented. Then, software interdependence, programming languages, IDE, and sponsor’s reputation are presented along with respective propositions. Finally, a brief discussion section is provided.

**Literature Review**

**Open Source Communities and Software**

The Internet relies on open source software and practices surprisingly more than an inattentive user perceives. According to The Economist “[every] time [I]nternet users search on Google, shop at Amazon or trade on eBay, they rely on open source software. More than two-thirds of websites are hosted using Apache, an open source product that trounces commercial rivals.”

An open source community is composed of contributors, in the sense that they are not necessarily paid or employed, when they are not, they are referred as volunteers. These contributors are dispersed geographically and brought together...
through some sort of IT structure, mainly the Internet (Hertel et. al, 2003). They are sometimes broadly referred to as virtual teams, which can be defined as “a group of people who interact through interdependent tasks guided by common purpose[...] across space, time and organizational boundaries with links strengthened by webs of communication technologies” (Lipnack and Stamps, 1997). By the same reasoning, virtual organizations were defined by DeSanctis and Monge (1999) as “[...] a collection of geographically distributed, functionally and/or culturally diverse entities that are linked by electronic forms of communication and rely on lateral, dynamic relationships for coordination.”

Open source communities are composed of “hobbyists” (volunteers), but the number of paid (e.g. by IBM and Sun Inc) contributors developing open source software seems to be increasing. Sometimes these communities have hundreds of collaborators. For instance, Egyed and Joode (2004) stated that “the Apache community roughly comprises 630 contributors of which about 90 belong to the core developer group”. The product (software), as well as its content (source code⁸), produced by those communities are always made available on the Internet free of charge (Hertel, 2003) and the reasons why (motivations) those developers get involved in such activity have also been discussed.

Although individual social-psychological motivations are hardly empirically measured, some studies have found that most of the volunteers’ effort can be explained by the enhanced reputation granted to them by being part of the group/project, which is referred as signaling (O’Mahony, 2003; von Hippel, 2001). Along similar lines, another characteristic attributed to those developers was that they “[were] not driven by monetary rewards but by competitive motives of status and reputation” (Hertel et. al, 2003). Supportive of these statements, Lee and Cole (2003) observed that the proportion of people joining the Linux group that does not, in fact, develop software has grown faster than the ones that do. Moreover, these authors stated that every component (file) of the software comes with a credits file, recognizing and describing the work of each member publicly.

Moon & Sproull (2002) and Hertel et. al (2003) pointed out some characteristics they have observed in open source communities. They have identified (1) “a general culture in which authority comes from competence”, (2) the presence of a “delegative and participative leadership principles combined with clear responsibilities”, (3) “a modular project structure that decreases unnecessary complexity”, (4) “a parallel release policy that simultaneously enables rapid development and a stable working system”, (5) “a motivating credit policy that not only acknowledges the contributions of developers but also, for instance, documentation work”, (6) presence of “clear rules and norms of the community that are communicated online”, and (7) a “simple but reliable communication tools that are available worldwide (e-mail, file transfer, Usenet discussion groups).”

Some open source software are market leaders, such as the web server Apache. Others such as Linux and

⁸ “[I].e., the human-readable commands [...] of a computer program.” Hertel et. al (2003)
OSS Projects Attractiveness – Santos & Nelson

OpenOffice.org are seen as good candidates for substitute software developed by giant corporations like Microsoft (Hert et. al, 2003). “As a natural experiment, the Linux project has demonstrated the feasibility of a large-scale, online collaboration effort where developers and users can be one and the same”, suggesting a high-quality product (Lee and Cole, 2003). But many other projects (the majority) have no expressiveness nor competitiveness.

However, whatever the open source communities and developers characteristics are, one the biggest issues in the IT industry and in academia right now is how to make use of open source software practices as a building block for delivering business value. As stated by Hertel et. al (2003): “[o]ne of the most compelling aspects of Open Source Software projects is that they are predominantly based on voluntary contributions from software developers without organizational support in a traditional sense.” As indicated, this scenario might have to, or is about to change. This paper is a first step towards understanding this phenomenon.

Theories of Business Practice and Technology Adoption

It is common knowledge in the business literature to assume that managers act under norms of rationality, and it is so assumed here. That makes one to expect that adopters of open source practices, or of any business practice, intend to observe “desirable” outcomes. Thus, in this case they should expect to receive inputs and attract as many people as possible, which ultimately would translate in better software quality.

However, this scenario of rationality represents one stream of thought used to explain why managers decide to adopt a business practice or technology. The opposite of this situation would be what has been deemed in the literature as fad and fashion, and constitutes the second stream of thought used to describe manager’s behavior.

Abrahamson and Fairchild (1999) defined fad and fashion as a “relatively transitory collective beliefs, disseminated by the discourse of management-knowledge entrepreneurs that a management technique is at the forefront of rational management progress.” Of course the accuracy of the “discourse” is questionable as are the company’s decisions of whether or not to follow the “discourse.” To simply follow managerial trends has the potential to work against the value increasing management of a company.

Despite the difficult and as of yet unachieved task of empirical proof, there is an extensive body of research on the theme of fad and fashion in the business field. A variety of management technologies are made available to managers nowadays. Some examples are downsizing, TQM, ERPs, and – of special interest here – software source code release. These management technologies are not expected to always pay off. Because, despite industry and academic evidence, ERPs and other widely used technologies are not always cost effective. The fact that companies contrive to “follow the crowd” may be
explained through theories of fad and fashion (Fichman, 2004; Miller and Hartwick, 2002; Benders, 2001).

Having said that, this study assumes that managers should be rational in their choices to adopt business practices or technologies, in this case, when releasing software source code to the community. Therefore, we assume that they expect that these projects remain active. Thus, as a starting point, an assumption to be empirically tested by further research is set, considering the exploratory nature of the study, allowing the second objective of the paper to be pursued. That is, to study the determinants of the numbers of contributions and contributors a project (i.e. its attractiveness).

**P1**: Open source model practices were adopted because they were supposed to bring as many inputs from the community on the released source code as possible, as well to attract as many contributors as possible - ultimately generating higher quality software.

**Predictors**

The existence and emergence of a corporate trend to release proprietary hardware design and software as open source (e.g. IBM’s Eclipse and Sun’s UltraParc) is the phenomenon under investigation here. So far, the research endeavor has focused on the developer motivational side (e.g. psychological and sociological) issues. However, “results suggest that [only] 28.7% of the variance of the dependent variable, contribution of OSS developers, can be explained by motivation variables” (Li et al., 2006). Accordingly, we rely on motivational issues but adopt a different set of variables in order to pursue the explanation of the remaining non-explained variance.

The model here developed focus on characteristics of the software and the overall project. Preliminary analysis of data collected by these authors from the SourceForge.org web site has demonstrated that the number of members a project has vary significantly across different groups of projects (e.g. database and financial). However, the specifics of how these groups’ characteristics are able to differentiate them were not fully explored due to the status of the theoretical model here developed. Issues such as measurement are still under development.

The set of variables chosen to predict the corporate-sponsored open source project attractiveness can be divided in three distinct groups, each of which could referred to as a theoretical construct. First, we developed a construct called project architecture, which is composed of characteristics such as programming language, IDE, and modularity. Second, sponsor’s reputation is introduced. Third, the sponsor’s degree of commitment to the project, meaning the number of paid-developers and money invested on it, is presented. These three entities and their influences on project attractiveness are depicted in figure 1. Later on, these three entities as well as the project attractiveness are expanded into a more operational (lower level) set of variables. Note that the arrow leading from attractiveness to software quality is dotted due to that be an assumption of this study, not discussed in detail here.
Project Architecture

Software Interdependence

In a general manner, interdependence within organizations might be described as Cheng (1983) pointed out: “the extent to which the organization’s task requires its members to work with one another”. This definition goes beyond the Thompson (1967) work flow definition, which considered only the inputs and outputs of materials and objects between and among units.

Software is similar to any other product; therefore, can be broken down in parts, called modules, which can be illustrative thought of as chunks of the overall software. The final product (software) is the summation of its parts (modules), which in turn can be seen as one “organization’s task”, as in Cheng (1983)’s definition of interdependence. It is known that the composition of interdependence is beyond the flow of work between and among the actors. However, in the case of software development, the interdependence of the members has been considered a consequence of the technical interdependence of the modules. Thus, in this paper, we discuss technical and social interdependence interchangeably, assuming that the other three dimensions are built-in. These modules might be dependent on each other in several ways. It is this degree of this interdependence between and among modules and/or actors that we refer to as interdependence or cohesion.

Baldwin et. al. (2003) demonstrated that modular projects have advantages in recruiting contributors. They grounded their argument on the quantity of modules or slices: the more you have, the more opportunities you offer, thus enhancing the potential contributors’ motivations. However, one thing that cannot be set aside is that the quantity of modules is expected to
influence interdependence. A potential trade-off exists here.

Especially in the open source case, in which people work geographically dispersed and the main communication medium is e-mail or list-servers, interdependence might be expected to be one of the main factors for attractiveness (West and O’Mahony, 2005). Also, the relationship between interdependence and contributions was studied by DeSouza et. al. (2004a) and DeSouza et. al. (2004b). They demonstrated that software with low interdependence is more likely to receive collaboration than those with high interdependence. That is because it facilitates the source code inspection function (debugging), software testing, comprehension, maintenance and parallelization. (Xu et al., 2005; Counsell and Swift, 2006). Also, volunteers tend to like to be able to work independently. So, at first glance, one would expect that managers would pursue less interdependent modules as much as possible. But, besides the fact that one module cannot be built completely independent from the others (DeSouza et. al., 2004a), a trade-off is expected to occur due to the relationship between interdependence and source code programming learning difficulty. Therefore, manager’s decision regarding the degree of interdependence is crucial and complex and so a high degree of rationality should be expected.

Nevertheless, rationality may fail to prevail for different reasons: (1) software development activities are expected to suffer pressure from customers for richness, (2) market strategies might push deadlines (Mockus and Herbsleb, 2002), and (3) an alternate version of the software might already have been developed when the decision to launch it has been taken.

In sum, two factors are expected to influence each other and, consequently, the number of contributions received: quantity of modules (quantity of opportunities offered), and degree of interdependence. Thus, we have that:

**P2.1**: The quantity of modules is expected to be positively correlated with the number of contributions received. This relationship is moderated by the degree of interdependence among the modules, controlling for the number of contributors across projects.

**P2.2**: The quantity of modules is expected to be positively correlated with the number of active contributors in a project. This relationship is moderated by the degree of interdependence among the modules.

Put differently, the number of contributions received and the number of contributors are expected to increase along with the quantity of modules up to a certain degree of interdependence, where the costs of ‘understanding’ the whole software becomes prohibitive and makes the relationship between the factors inversely proportional.

**Programming Language**

Software source code is written in a language, and the diversity of programming languages available continues to grow. For example, operational systems such as Linux are normally written in C or C++, web sites might be written in Perl, JavaScript, Java, etc. or a combination, and software applications such as Microsoft Office or Mozilla might also be written in any of those languages mentioned, as well as Delphi, Visual Basic and others. Some software, such as OpenOffice.org,
accepts the inclusion of source code in different languages (Java, Python, StarBasic or JavaScript\(^9\)).

To develop software requires knowledge of the language it is written in. In the case of maintenance, it is the original language(s) and the languages it accepts that must be used when adding new source code. Those languages vary in functionality, portability, compatibility, ease of use, popularity, and the license type, and some are open source and others proprietary. Therefore, we should expect that the choice of which language to adopt in a software development project will influence some of its outcomes.

In the context of the open source practice studied here, the choice of the software’s programming language restricts the population of potential contributors to a smaller group that is familiar with and sufficiently skilled in that language. For example, the TIOBE Programming Community Index for April 2007\(^{10}\) found that Java is the most popular language, with thousands of projects using it. Thus, high availability of support, easy exchange of information and easy access to documentation are natural consequences of the programming language choice adopted. Therefore, a balance between the language characteristics needed or desired in a project and the popularity of the language among potential contributors is likely to increase the probability of receiving contributions and attracting contributors. This is especially true in the case of a project that intends to be attractive for open source sympathizers. Therefore, if the software language is proprietary and needs a license, the opportunity is not expected to be seen as favorably by the open source community, which is usually composed of people that advocate against proprietary type of licenses. So, a potential misfit between proprietary languages and projects of this nature might occur. Thus, we have that,

\[ \text{P3.1: The more popular a programming language is among the overall population of potential contributors (market share), the more contributions a project receives, when a misfit is not observed.} \]

\[ \text{P3.2: The more popular a programming language is among the overall population of potential contributors (market share), the more contributors a project has, when a misfit is not observed.} \]

**Integrated Development Environment (IDE)**

Integrated Development Environment (IDE) is another variable that must be considered when deciding which language to adopt in a software development project. The programming language choice restricts the range of IDE available, and vice-versa. This is expected to influence the likelihood of receiving inputs and attracting volunteers (i.e. project attractiveness) due to their familiarity with and willingness to learn a specific language/IDE adopted in a project.

IDE was firstly and broadly defined by Konsynski, et. al (1984, p. 67) as “a complete and unified set of concepts, techniques, and tools that covers the entire development process.” Later, and precisely related to software development, it was described by Kline and Seffah (2005, p. 608) as a “computer software that generally consists of a source code editor, a

\(^{9}\) [http://contributing.openoffice.org/programming.html](http://contributing.openoffice.org/programming.html)
\(^{10}\) [http://www.tiobe.com/tpci.htm](http://www.tiobe.com/tpci.htm)
compiler or interpreter (or both), build-automation tools, and a debugger. Examples of IDEs for Java programming include Eclipse, Netbeans, Forte (Sun Microsystems), VisualAge for Java (IBM), JBuilder (Borland), Visual Cafe (Symantec), and Visual J# (Microsoft), and examples for C++ programming include Visual C++ (Microsoft) and C++ Builder (Borland).” So, as it was stated, each IDEs support a specific language and a specific language is supported by a smaller group within the population of available IDEs. Thus, priority must be set on which decision to make first, the language or the IDE.

Use of IDEs, especially when one is a standard among developers, is expected to influence productivity. Studies have found that IDE tools often vary on functionality, usability and are “difficult to use, learn, and master”, according to Kline and Seffah (2005, p. 607), implying the existence of learning and time costs to develop software with it. Kline and Seffah (2005, p. 625) advise that “developers should be provided with IDEs that offer functionalities in more rational, less visually complex formats that reinforce the relation between a specific functionality and the software artifacts on which that functionality acts.” Furthermore, the increasing advertising surround IDEs such as expert’s opinions\(^{11}\), polls\(^{12}\), or awards like IDE of the year\(^{13}\), is likely to influence anything with the IDE’s image linked to it. Thus, just as a programming language is expected to influence the project’s contributions and contributors, so is the IDE.

**P4.1**: The more popular an IDE is among the overall population of contributors (market share), the more contributions a project will receive, when a misfit is not observed.

**P4.2**: The more popular an IDE is among the overall population of contributors (market share), the more contributors a project will have, when a misfit is not observed.

**Project Sponsor’s Reputation in the IT Industry**

It has been shown in the literature that open source project volunteers are partly motivated by the opportunity to work on state-of-the-art software, but are also motivated by signaling (Hertel et. al, 2003). Thus, it is implied that they are interested in promoting themselves to open source project sponsors. Also, one might notice that IT jobs advertisements can be found on open source projects’ host websites such as Sourceforge.net or Slashdot.org.

In summary, factors such as the potential developers perception of the quality of a project, the sponsor reputation in the IT industry, and a large amount of downloading and activity\(^{14}\) a project has are potential influences on the decision of an individual to become involved in a project as well as to contribute to it, assuming that they are (also) signaling. Consequently, we have that:

**P5.1**: The number of contributions a project receives from volunteers increases with their perceived level of sponsor’s reputation in the IT industry.

---


\(^{13}\) [http://jdj.sys-con.com/read/171303_3.htm](http://jdj.sys-con.com/read/171303_3.htm)

\(^{14}\) Open source software’s host web sites classify projects in big groups, such as finance, network, etc., and present them, automatically, sorted by quantity of downloads the software had and how active the project is. Thus, the more active a project is and the more downloads it had, the more visible a project will be.
P5.2: The higher the sponsor’s reputation in the IT industry, the more contributors a project has.

Sponsor's Degree of Commitment to Project

Quantity of Developers Paid by Sponsor(s)

The more resourceful (i.e. money and reputation) a project’s sponsor is, the easier it will be for the project to be visible. For example, IBM, the main sponsor of the Eclipse project\textsuperscript{15}, as demonstrated by O’Mahony (2005), is the major contributor of bug reporting and bug fixing, showing how representative the role of paid contributors can be. Also, it was stated at the Open Source Workshop at the University of Texas at Austin on May 2, 2006 that IBM has provided approximately 800 developers to work on open source projects. Consequently, projects that receive help from those developers tend to have a constant level of activity, an increase in its visibility, as well as its reputation and quality in potential contributor’s minds. Additionally, Hert et. al (2003) pointed out that “the more developers were paid for their Linux-related work the more time they spent [on the activity]”. Thus, we have that:

P6.1: The more corporate sponsored contributors a project has, the more contributions from volunteers it receives.

P6.2: The more corporate sponsored contributors a project has, the more volunteers it has.

Amount of money spent on the project (e.g. advertising and planning)

Finally, the amount of money spent on the project for its planning, release and the publicizing of its objectives is expected to be reflected positively in some of its outputs.

Many open source projects exist nowadays, increasing the competition for volunteers. However, it is very unlikely that any developer willing to volunteer for one of them will consider each and every one available before deciding in which to engage. Accessibility is an issue, and this is expected to be influenced by investment in the advertising of the project. Moreover, the more time (workforce) that is devoted to activities such as the planning or evaluation of a project, the more improvement is expected to be observed (e.g. better balance of the quantity of modules and the interdependence among them or faster decision-making processes). So, we have that,

P7.1: The more time and money a sponsor devotes to develop a project, the more contributors (volunteers) a project is expected to have.

P7.2: The more time and money a sponsor devotes to develop a project, the more contributions from volunteers a project is expected to have.

All propositions discussed so far are depicted in the figure 2.

\textsuperscript{15} \url{www.eclipse.org}
**Discussion**

The adoption of open source software and their practices in corporations has reshaped the software sector. A satisfactory understanding of this phenomenon has not yet been achieved, creating a need for studies such as this.

The open source software recognition of high quality and capable of substituting proprietary software is a consequence of the model of development adopted by the communities. Because of this, corporations have tried to copy and adapt their practices. As open source adoption increases in companies, how and why this occurs becomes of interest to researchers. This paper is a first attempt to discover which variables are good candidates for predicting the number of contributions (inputs) received and the number of contributors that a project based on the open source software model has had – project attractiveness – and what sponsors’ and projects’ characteristics make them more likely to receive inputs from contributors as well as attract them.

Since this is a theoretical paper, its main limitation is the lack of empirical data. Given that, its first proposition and main assumption might not hold. It might be the case that real inputs on software development projects from open source community contributors and the number of contributors are secondary and considered only a desirable side-effect, but are not the main goals of launching a project to the public. Perhaps managers’ real intention when soliciting open source ...
contributions is motivated by something else. Possible conflicting explanations could include the development of a vehicle of advertisement or the reduction of future employee hiring costs since confirmed contributions (by a volunteer) result in less need for training and a faster learning curve. Further empirical research is needed to examine these possibilities.

By presenting the potential indicators of attracting contributors and contributions to corporate open source projects, this paper begins to define the dynamics of how virtual social structures and existing corporate social structures meet and collide. Software developers may no longer be recruited through traditional channels; rather they could be “auditioned” via their open source contributions. As a result, the culture of organizations may also become significantly more competitive. Employees will no longer compete only with each other; they will have to work harder to keep their skills current because their work will be compared to that of open source contributors. Additionally, technical skills will become more easily replaceable, but business domain knowledge may become the key to stable employment. The first step in understanding this phenomenon is to uncover what motivates corporations to enter in the open source world and what enticements they can (and are willing to) offer to obtain participation.

References


13


