Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

Vol. 9, No 2

2010

doi:10.5329/RESI.2010.0902

Sumário

Revista hospedada em: http://revistas.facecla.com.br/index.php/reinfo
Forma de avaliação: *double blind review*

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente. Ela voltou a ser diagramada em uma única coluna para facilitar a leitura na tela do computador. Mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*, que gasta menos tinta.

(mapa de palavras com os termos mais frequentes nos artigos desta edição)

MOTIVATION TO CREATE FREE AND OPEN SOURCE PROJECTS AND HOW DECISIONS IMPACT SUCCESS[1,2]

(paper submitted in September, 2010)

Carlos Denner Santos Jr.
Departamento de Ciência da Computação
Universidade de São Paulo (USP)
denner@ime.usp.br

Kay M. Nelson
College of Business
Southern Illinois Univ. at Carbondale (SIUC)
ikay@business.siuc.edu

ABSTRACT

As a consequence of the success of free and open source software such as Linux, organizations started to rethink development practices and opensource their applications. To opensource software means to release its source code open to the general public in the Internet. That has become an increasingly common strategy in the industry over the last years (e.g., Netscape-Navigator, IBM-Eclipse, and CAIXA-Curupira). However, what motivates organizations to commit their resources to publicize proprietary software is yet to be fully understood; and it is only after this motivation is comprehended that it will become possible to define, measure, and study success. This paper fulfills this gap in the literature by proposing a theoretical model that satisfies technical (software quality) and organizational (business value) requirements at the same time, defining what would be, thus, return on investment and how to achieve it. Specifically, the model proposes that, in opensourcing software, organizations should attempt (1) to attract users and developers, and (2) to receive contributions from them, mainly because achieving these goals makes it more likely to expand the user base and build an active community that constantly improves the software. To work towards these goals, this paper develops awareness of how (a) software architecture (modularity and interdependence), (b) programming language and integrated development environment, and (c) sponsor's reputation and degree of commitment can influence a project's dynamics.

Key-words: open source software; free software; software development; technology adoption; business strategy; software quality; software industry; software engineering.

RESUMO

Como resultado do sucesso de iniciativas de software livre e aberto como o Linux, muitas empresas estão repensando suas práticas de desenvolvimento e utilizando software aberto em suas aplicações. A filosofia do software livre envolve a liberação do código fonte de forma aberta para o público em geral na Internet. Isto está se tornando uma prática cada vez mais comum no mercado, ao longo dos anos (ex., Netscape-Navigator, IBM-Eclipse e CAIXA-Curupira). Contudo, o que motiva empresas a comprometer seus recursos e divulgar livremente o seu software ainda precisa ser melhor compreendido. Apenas depois que isso ocorrer será possível definir, medir e estudar o sucesso dessas iniciativas. Este artigo procura preencher esta lacuna na literatura, propondo um modelo teórico que satisfaça os requisitos técnicos (qualidade de software) e organizacionais (valor de negócio) simultaneamente, definindo o que seria o retorno do investimento e como obtê-lo. O modelo propõe que ao adotar software livre as empresas devem tentar (1) atrair usuários e desenvolvedores e (2) receber contribuições deles, principalmente porque, ao conseguir atingir esses objetivos, ficará mais fácil expandir a base de usuários e construir uma comunidade ativa que melhora o software constantemente. O artigo desenvolve a compreensão de como (a) a arquitetura de software (modularidade e interdependência), (b) a linguagem de programação e o ambiente de desenvolvimento e (c) a reputação do patrocinador e seu grau de comprometimento influenciam a dinâmica de um projeto.

Palavras-chave: software aberto; software livre; desenvolvimento de software; adoção de tecnologia; estratégia; qualidade de software; setor de software; engenharia de software.

---

[1] A previous version of this paper was presented at AMCIS, in 2007.

[2] The authors would like to thank FAPESP for the funding, and FLOSS Competence Center (CCSL-IME-USP) for the technical support and structure provided.

1 INTRODUCTION

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 that can be used, modified and distributed by anyone, even if in support of for-profit activities. This type of software is referred to as Free and Open Source Software (FOS), and can be easily found nowadays in the Internet.

According to Benkler (2006), the appearance of FOS has impacted the economy broadly, changing deeply the environment of industries in which players rely on information technology (IT) to operate. That impact comes from at least two sources: first, from the provision of software and infrastructure for application development at no acquisition cost; and second, from the publicity of the processes of how production activities take place in FOS (i.e., through peer-review in virtual communities). By adopting FOS tools and understanding these community processes, organizations interested in user-centric initiatives are able to adopt and adapt the 'open source business model', which relies on voluntary contributions to be successful (e.g., Wikipedia). The scope of FOS impact is not likely to change in the near future, for it has been empowered by major corporations, such as IBM, which made FOS part of their businesses, signaling that the socio-economic impact of free software is sustainable (RIEHLE, 2007; FITZGERALD, 2006).

The high quality of FOS is recognized to be a result of the development practices of their communities. Consequently, corporations have tried to mimic FOS practices in order to achieve better results, increasing their proximity with customers. The resulted business practice consists of releasing software source code open to the community in the Internet, sharing knowledge in an attempt to receive contributions. As this practice popularizes, questions of when and how it can succeed (or not) become of interest to researchers and practitioners. This paper is a theoretical attempt to grasp founders' motivations to engage in such practice, so far not fully understood (SANTOS JR., 2008). Furthermore, it is an attempt to explain the amount of contributions a project receives, and the number of contributors it has, both indicators of a FOS project attractiveness (SANTOS JR. *et al.*, 2010).

The number of corporations engaged in FOS communities can potentially change the entire organizational structure of the IS organization and lead to major shifts in corporate norms. So far, attention has been given in the management literature to volunteers' motivations for engaging in such activities. The main findings concentrate on issues like increased reputation that leads to career advantages (HERTEL *et al.*, 2003). But corporate motivations have been overlooked.

Some FOS are considered to be as good as their proprietary competitors. Apache, for example, is the leader in the web server industry

doi:10.5329/RESI.2010.0902004

for over a decade (VON KROGH; SPAETH, 2007). Thus, if one understands and is able to provide what attracts skillful volunteers, corporate opportunities might be realized. This setting gave birth to a new business strategy – opensourcing – which is an attempt to recruit contributors to develop and maintain software and to increase customer base (AGERFALK; FITZGERALD, 2008). However, when such practice becomes a trend, 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 (WEST; O'MAHONY, 2005). Consequently, the number of contributions expected to be received, on average, from adopting this practice, is reduced, creating a managerial problem and justifying careful analysis to apply corporate resources wisely.

The recognition of the FOS impact in the software industry has resulted in an attempt to bring it closer to the corporate world, which required the investment of resources, bringing risks along (FITZGERALD, 2006). The investment in opensourcing has many similarities with other practices such as downsizing, outsourcing, TQM, and so forth. This means that it might pay off or not, depending on the case and on the definition of success.

In this paper, the business practice is represented as an attempt to get help from the community on internal production processes, especially those related to software development. For that, companies release software source code open on a website, and, from that moment on, expect to obtain inputs from volunteers to increase software quality, expanding their relationships with potential and current customers. Organizations that have adopted this model are, for example, the Brazilian government, Sun Inc. (Oracle) and SugarCRM.

Specifically, this paper explores the influences of project characteristics such as overall software interdependence (cohesion), sponsor reputation, and the popularity of the programming language and integrated development environment (IDE) adopted on success. We understand that success is achieved through higher software quality, which would impact software adoption rates positively, benefiting then the sponsor-corporation business.

The research questions discussed are: (1) What motivates corporations to opensource? (2) Does the IDE choice influence the project attractiveness to the community? (3) Does the programming language choice influence the project attractiveness to the community? (4) Does the overall software interdependence (cohesion) influence the project attractiveness to the community? (5) Does the sponsor reputation influence the project attractiveness to the community?

The remainder of this paper is organized as follows: in the next section, a description of free and open source software communities is provided; theories of business practices and technology adoption are discussed afterwards; then, software interdependence, programming

languages, IDE, and sponsor's reputation are examined; and finally, results are discussed.

2 LITERATURE REVIEW

2.1 FREE AND OPEN SOURCE SOFTWARE AND COMMUNITIES

The Internet relies on free and open source software surprisingly more than an inattentive user perceives. According to The Economist (2006), every "time Internet 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 might be paid to contribute, buy may also be volunteers. These contributors are dispersed geographically and brought together through an IT structure, mainly the Internet (HERTEL *et al.*, 2003; MARKUS *et al.*, 2000). These communities are 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; STAMPS, 1997, p. 18). By the same reasoning, virtual organizations were defined by DeSanctis and Monge (1999, p. 694) 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."

Additionally, Moon and Sproull (2002) point out some characteristics they have observed about open source software communities. They have identified: (1) "a general culture in which authority comes from competence"; (2) the presence of "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) the presence of "clear rules and norms of the community that are communicated online"; and (7) "simple but reliable communication tools that are available worldwide (e-mail, file transfer, Usenet discussion groups)."

Open source communities are composed of hobbyists, but the number of paid (e.g., by IBM) contributors developing open source software is increasing. Some of these communities have hundreds of members (e.g., Linux). For instance, Egyed and Joode (2004, p. 72) 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 code), produced by those communities are always made

available on the Internet free of charge. Contributors' motivations have been the focus of past research (HERTEL *et al.*, 2003).

Previous 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 to as the signaling motivation (O'MAHONY, 2003; VON HIPPEL, 2001). Similarly, it has been found that those developers were not motivated by monetary compensation, but by competitive reasons of status and reputation (HERTEL *et al.*, 2003). To support these findings, Lee and Cole (2003) observed that every component (file) of the software comes with a credits file, recognizing and describing the work of each contributor publicly.

Some open source software are market leaders, such as the Apache web server. Others such as Linux are considered good candidates to substitute proprietary software developed by corporations as big as Microsoft (HERT *et al.*, 2003). The success of Linux, as stated by Lee and Cole (2003, p. 178), "demonstrated the feasibility of a large-scale, online collaboration effort where developers and users can be one and the same". However, most of free and open source projects have no expressiveness or competitiveness.

In this scenario that blends success and failure, one important topic of study is how to make use of the open source software practices as building blocks for delivering business value. As stated by Hertel *et al.* (2003), one of the aspects of open source software that is most compelling to business is its predominant voluntary characteristic, in which contributors are not supported by organizations in the traditional sense. This paper is a step towards understanding what can separate successful projects from the rest.

2.2 ADOPTION OF BUSINESS PRACTICES AND TECHNOLOGY

It is common in the management literature to assume that managers decide under norms of rationality. It is assumed similarly here (i.e., an implemented strategy represents an intention to achieve specified goals). Accordingly, we know that adopters of FOS practices are looking for "desirable" outcomes. In the case of opensourcing, source code releasers intend to receive external feedback on their products and attract as many people as possible, what translates into business value by magnifying brand exposure, user base and software quality (SANTOS JR. *et al.*, 2010).

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 managerial behavior. Abrahamson and Fairchild (1999, p. 709) 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." To act under these conditions increases the

likelihood of failure on the task of achieving a goal. Nevertheless, the imitator would have its expected goal defined by imitation, which would thus be the same of the rational manager.

Despite the difficulty of providing definitive empirical proof, there is an extensive body of research on the theme of fad and fashion in the management literature. A variety of management techniques are available to managers nowadays. Some examples are downsizing, TQM, agile methods, and software source code release. These techniques are not expected to work in any situation, and most of them become less popular over time. Thus, as companies continue to contrive to "follow the crowd", adoption' patterns appear to partly fit theories of fad and fashion too (FICHMAN, 2004; MILLER; HARTWICK, 2002; BENDERS; VAN VEEN, 2001).

Having presented the two main perspectives on practice adoption, this study assumes that managers should attempt to be rational in their choices of releasing software source code to the community. In other words, managers expect that projects gain attention and remain active after public release. Thus, as a starting point, an assumption to be empirically tested by further research is set, allowing the second objective of the paper to be pursued – that is, to study the determinants of the numbers of contributions and contributors of a project.

Proposition 1: Managers release source code open to receive as many external inputs, and to attract as many viewers, users and contributors, as possible; these outcomes are believed to increase software quality, brand exposure and, ultimately, business and/or social value.

3 MODEL DEVELOPMENT

The existence and popularity of a corporate trend to release proprietary hardware design and software as open source is the phenomenon under investigation here (e.g., IBM's Eclipse and Sun's UltraParc). So far, past research has focused on the volunteer side, studying its psycho-sociological motivations. However, research findings suggest that only about a third of the variance of open source software developers' contributions can be explained by motivation variables (LI *et al.*, 2006). Accordingly, we adopt a different set of variables in order to pursue the explanation of the remaining non-explained variance.

The model here developed focuses on characteristics of the software and the overall project. For example, preliminary analysis of the data collected from SourceForge.net suggested 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 affect their ability to have contributors are not fully understood.

The set of variables chosen to explain the corporate-sponsored open source project attractiveness can be divided in three distinct groups or theoretical constructs. First, we discuss a construct called *project*

architecture, which is composed of characteristics such as programming language, IDE, and modularity levels. 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. Finally, these constructs' influences on *project attractiveness* are expanded into a more operational, lower level, model. These proposed relationships are depicted in Figure 1.

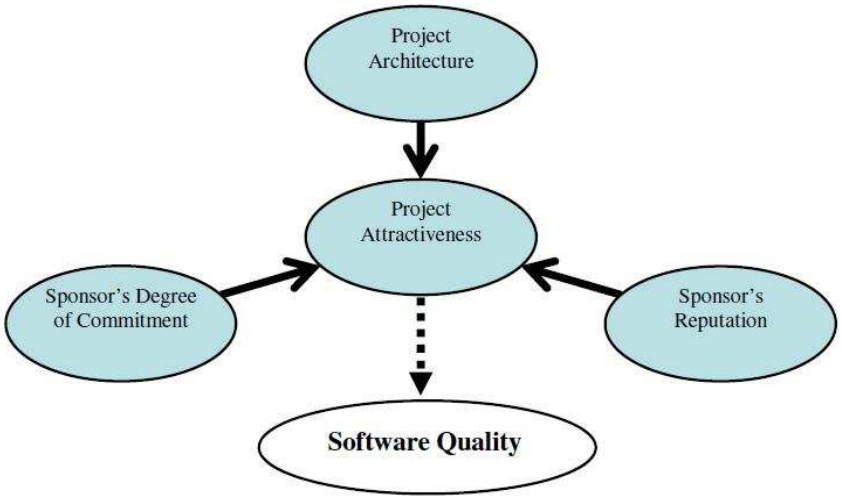

Figure 1. Higher-Level Theoretical Model

Source: the authors

3.1 PROJECT ARCHITECTURE

3.1.1 Software interdependence

In general, Cheng (1983) defined *work interdependence within organizations* as the extent to which a task requires its involved members to exchange information with one another. This definition goes beyond Thompson's (1967) *workflow* definition, which considered only the exchange of materials and objects between units.

To develop software is an organizational task and can be broken down in parts, called modules, which can be understood as chunks of the overall software. The final product (application) is the aggregate of its parts (modules), which in turn can also be seen as an organizational task. Possibly, the social composition of interdependence is beyond the workflow between 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, we discuss technical and social interdependence interchangeably. Modules might be dependent on each other in several ways. It is this degree of this interdependence between modules (or actors) that we refer to as *interdependence*.

Baldwin and Clark (2003) demonstrated that modular projects have advantages in recruiting contributors. The more modules you have, the more opportunities you offer, which enhances your chances of receiving contributions (BENKLER, 2006). However, one thing that cannot be set aside is that the quantity of modules is expected to influence interdependence between them. A trade-off exists here.

Especially in the open source case, where people work geographically dispersed and the main communication tool is e-mail, interdependence is expected to be one of the main factors for maintaining an active community (WEST; O'MAHONY, 2005; MACCORMACK *et al.*, 2006). 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 contribution than those with high interdependence. Low interdependence facilitates the source code inspection function (debugging), software testing, comprehension, maintenance and parallelization (XU *et al.*, 2005; COUNSELL; SWIFT, 2006).

Moreover, volunteers tend to enjoy working independently. So, one would expect that developers would pursue less interdependent modules as much as possible, for a trade-off between interdependence and source code programming learning difficulty exists. That creates a complex scenario; where modularity stimulates contributions, but also favors complexity because one module cannot be built completely independent from the others (DESOUZA *et al.*, 2004a). Therefore, manager's control regarding the number of modules and the degree of interdependence between them is crucial for FOS project founders.

Nevertheless, managerial action on these matters may fail for different reasons, for example: (1) software development activities suffer pressure from customers for new features constantly; (2) market strategies push deadlines (MOCKUS; HERBSLEB, 2002); and (3) commonly, different versions of one software have to be managed in distributed activities, such as open source development.

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

Proposition 2.1: The quantity of modules is positively correlated with the number of contributions received. However, this relationship is moderated by the degree of interdependence among the modules.

Proposition 2.2: The quantity of modules is positively correlated with the number of contributors. However, this relationship is moderated by the degree of interdependence among the modules.

In a nutshell, 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.

3.1.2 Programming language

Software source code is written in a language, and the diversity of programming languages available continues to grow. For example, operating systems such as Linux are normally written in C, web sites might be written in Perl, JavaScript, Java, or a combination of them. Moreover, software applications such as Microsoft Word or Mozilla Firefox might have code in those languages mentioned, as well as Delphi, Visual Basic, C# and others. Software such as the OpenOffice might have source code in various languages.

To develop software requires “fluency” on the language it is written in. In the case of maintenance, it is the original language that must be used to add new, or fix existing, source code. Programming languages vary in functionality, portability, compatibility, ease of use, and popularity. Some are open source and others are proprietary. Therefore, we should expect the choice of which programming language to adopt in a software development project to influence its dynamics.

In the context of open source, the choice of programming language restricts the population of potential contributors to a smaller group that is familiar with and sufficiently skilled on that language. For example, the TIOBE Index for August 2010 found that Java is the most popular language, with thousands of FOS projects using it (TIOBE, 2010).

The availability of support and easiness of access to documentation are direct consequences of the programming language 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 to open source sympathizers. If the software language is proprietary, the opportunity is not expected to be seen as favorably by the open source community, which is usually composed of people who advocate against proprietary licenses. So, a potential misfit between proprietary languages and projects of this nature might occur. Thus, we have that:

Proposition 3.1: The more popular a programming language is among the population of potential contributors, the more contributions a project receives, when a misfit is not observed.

Proposition 3.2: The more popular a programming language is among the population of potential contributors, the more contributors a project has, when a misfit is not observed.

3.1.3 Integrated development environment

Integrated Development Environment (IDE) is another variable that must be considered when deciding which language to adopt in an open source software project. The programming language choice restricts the options of IDE, and vice-versa. This is expected to influence the likelihood of receiving inputs and attracting volunteers 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 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 IDE supports specific languages, 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.

The use of IDE, especially of one that is popular among developers, is expected to influence productivity. According to Kline and Seffah (2005, p. 607), IDEs vary on functionality, usability and are "difficult to use, learn, and master", which creates costs to develop software with it. Furthermore, Kline and Seffah (2005, p. 625) explain 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." In addition, there is an increasing amount of information available on IDEs, such as expert's opinion, polls, and public awards. Likely, this scenario influences anything with a specific IDE image linked to it. Thus, just as a programming language is expected to influence the project's contributions and contributors, so is the IDE.

Proposition 4.1: The more popular an IDE is among the population of potential contributors, the more contributions a project receives, when a misfit is not observed.

Proposition 4.2: The more popular an IDE is among the population of potential contributors, the more contributors a project has, when a misfit is not observed.

3.2 SPONSOR'S REPUTATION

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 mainly motivated by signaling their skills (HERTEL *et*

al., 2003). Thus, it is implied that they are interested in being hired by open source project sponsors. Also, one should notice that IT job advertisements can be found on open source projects' repositories, such as Sourceforge.net or Github.com.

Accordingly, the potential contributor perception of "value" of a FOS project, which is influenced by its sponsors' reputation in the IT industry, potentially influences the decision of an individual to become a member of a project as well as to contribute to it. Thus, we have that:

Proposition 5.1: The number of contributions a project receives increases with its sponsors' reputation in the IT industry.

Proposition 5.2: The number of contributors a project has increases with its sponsors' reputation in the IT industry.

3.3 SPONSOR'S DEGREE OF COMMITMENT

3.3.1 Quantity of developers paid by sponsor

The more resourceful (i.e., money and reputation) a project's sponsor is, the more visible the project will be. For example, IBM, the main sponsor of the Eclipse project, 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 2nd, 2006 that IBM provides approximately 800 developers to work on open source projects on a full-time basis. Consequently, projects that receive help from paid contributors tend to have a constant level of activity, increasing its visibility, reputation and quality, as judged by potential contributors. Additionally, Hert *et al.* (2003, p. 1168) 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:

Proposition 6.1: The more paid-contributors a FOS project has, the more contributions from volunteers it receives.

Proposition 6.2: The more paid-contributors a FOS project has, the more volunteers it attracts.

3.3.2 Money spent on the project

Finally, the amount of money spent on the FOS project by sponsors for its planning, releasing and advertising is expected to influence positively its dynamics. Many open source projects exist nowadays, increasing competition for skillful contributors and their time. Consequently, it is very unlikely that a developer willing to contribute would consider each and every one available before deciding in which to engage. Visibility is an issue expected to be influenced by investment in the promotion and advertising of the FOS 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 the adoption of more suitable decision-making processes). Thus, we have that:

Proposition 7.1: The more time and money a FOS project's sponsors devotes to its promotion and advertising, the more contributors the project has.

Proposition 7.2: The more time and money a FOS project's sponsors devotes to its promotion and advertising, the more contributions the project receives.

Figure 2 depicts all propositions. Next, a discussion section appears with the final remarks of this paper, pointing out implications, limitations and future directions.

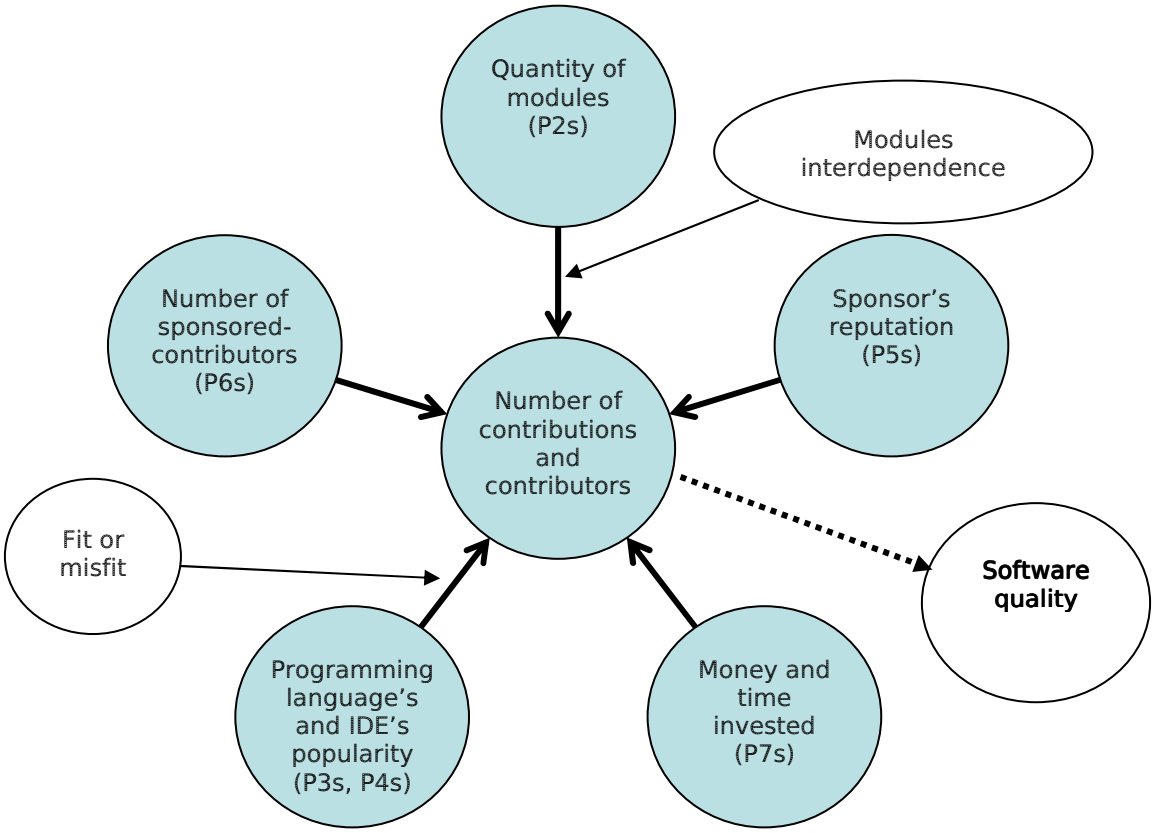

Figure 2. Lower-level theoretical model

Source: the authors

4 CLOSING REMARKS

The adoption of open source software and practices in organizations are reshaping the software industry. A satisfactory understanding of this phenomenon has not been achieved yet, creating a need for exploratory

studies and model development, such as this one in order to create theoretical grounds for future empirical validation.

The open source software recognition of high quality and capability of substituting proprietary software is a consequence of the model of development adopted by the communities. Because of this, organizations have tried to copy and adapt their practices. As technology, such as open source, adoption increases in companies, and understanding how and why this occurs becomes of interest to researchers and practitioners alike (GALLIVAN *et al.*, 2005). This paper is a first attempt to discover which variables are good candidates for explaining the number of contributions (inputs) and the number of contributors that a free and open source software project has. To do so, we focused on sponsors' and projects' characteristics that potentially make them more likely to stand out in competition with others, receiving more inputs from contributors as well as attracting more developers to become members. However, as a first step, our approach is not free of limitations and further research is clearly needed.

Given that this is a theoretical paper, its main limitation is the lack of empirical data to support the propositions. As such, its first proposition and main assumption might not hold; that is, it might be the case that receiving inputs on software development from external contributors and having a large number of members in the project are secondary and considered only a desirable side-effect. In that case, those would not be the main goals of launching software open to the public as we have assumed.

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 successful contributions (by a volunteer) result in less need for training and a faster learning curve for the hired volunteer. Only empirical research is able to answer this question and examine these possibilities. Furthermore, the explanatory variables presented here are arbitrary and, could, potentially, be accompanied of others such as contributors' ideology level or source code metrics, for example (STEWART; GOSAIN, 2006; MEIRELLES *et al.*, 2010). Nevertheless, this paper sets the necessary background to perform future field studies to expand and revise the model, which should then be encouraged.

By presenting a potential set of causes for attracting contributors and receiving contributions to organizational free and open source software projects, this paper begins to define the dynamics of how virtual organizational structures and existing 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 the open source community. 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 organizations to enter in the open source world and what enticements they can (and are willing to) offer to obtain participation.

REFERENCES

ABRAHAMSON, E.; FAIRCHILD, G. B. Management fashion: Lifecycles, triggers, and collective learning processes. *Administrative Science Quarterly*, v. 44, p. 708-740, 1999. doi:10.2307/2667053

AGERFALK, P.; FITZGERALD, B. Outsourcing to an unknown workforce: exploring opensourcing as a global sourcing strategy. *MIS Quarterly*, v. 32, p. 385-409, 2008.

BALDWIN, C.; CLARK, K. Does code architecture mitigate free riding in the open source development model? *Working paper*, Harvard Business School, June 1, 2003. Available at: http://www.people.hbs.edu/cbaldwin/DR2/BaldwinClark.ArchOS.Jun03.pdf. Accessed: Oct. 12, 2010.

BENDERS, J.; VAN VEEN, K. What's in a Fashion? Interpretative Viability and Management Fashion. *Organization*, v. 8, n. 1, 33-53, 2001. doi:10.1177/135050840181003

BENKLER, Y. *The wealth of networks*: how social production transforms markets and freedom. New Haven and London: Yale University Press, 2006.

COUNSELL, S.; SWIFT, S. The interpretation and utility of three cohesion metrics for object-oriented design. *ACM Transactions on Software Engineering and Methodology*, v. 15, n. 2, p. 123-149, 2006. doi:10.1145/1131421.1131422

CHENG, J. Interdependence and coordination in organizations: a role-system analysis. *Academy of Management Journal*, v. 26, n. 1, 1983. doi:10.2307/256142

DESANCTIS, G.; MONGE, P. Introduction to the special issue: communication processes for virtual organizations. *Organization Science*, v. 10, n. 6, p. 693-703, 1999. doi:10.1287/orsc.10.6.693

DESOUZA, C.; REDMILES, D.; CHENG, L.; MILLEN, D.; PATTERSON, J. Sometimes you need to see through walls: a field study of Application Programming Interfaces. In: CSCW'04, Chicago, Illinois, USA, November 6–10, 2004a.

DESOUZA, C.; REDMILES, D.; CHENG, L.; MILLEN, D.; PATTERSON, J. How a good software practice thwarts collaboration: the multiple roles of APIs in

software development. In: SIGSOFT'04/FSE-12, Newport Beach, CA, USA, Oct. 31–Nov. 6, 2004b.

EGYED, T.; JOODE, R. Standardisation and other coordination mechanisms in open source software. *International Journal of IT Standards and Standardization Research*, v. 2, n. 2, 2004.

FICHMAN, R. Going beyond the dominant paradigm for information technology innovation research: emerging concepts and methods. *Journal of the Association for Information Systems*, v. 5, n. 8, p. 314-355, 2004.

FITZGERALD, B. The transformation of open source software. *MIS Quarterly*, v. 30, n. 3, p. 587-598, 2006.

GALLIVAN, M.; SPITLER, V.; KOUFARIS, M. Does information technology training really matter? A social information processing analysis of coworkers' influence on IT usage in the workplace. *Journal of Management Information Systems*, v. 22, n. 1, p. 153-192, 2005.

HERTEL, G.; NIEDNER, S.; HERRMANN, S. Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. *Research Policy*, v. 32, p. 1159–1177, 2003. doi:10.1016/S0048-7333(03)00047-7

KLINE, R.; SEFFAH, A. Evaluation of integrated software development environments: challenges and results from three empirical studies. *International Journal Human-Computer Studies*, v. 63, p. 607–627, 2005. doi:10.1016/j.ijhcs.2005.05.002

KONSYNSKI, B.; KOTTEMANN, J.; NUNAMAKER, J.; Stott, J. PLEXSYS-84: An Integrated Development Environment for Informational Systems. *Journal of Management Information Systems*, v. 1, n. 3, 1984.

LEE, G.; COLE, R. From a firm-based to a community based model of knowledge creation: the case of Linux kernel development. *Organization Science*, v. 14, n. 6, p. 633-649, 2003. doi:10.1287/orsc.14.6.633.24866

LI, Y.; TAN, C.; TEO, H.; MATTAR, A. Motivating open source software developers: influence of transformational and transactional leaderships. In: Proceedings of the 44th ACM International Conference on Computer Personnel Research (SIGCPR), Claremont, California, USA, 2006.

LIPNACK, J.; STAMPS, J. *Virtual teams*. New York: John Wiley and Sons, Inc., 1997.

MACCORMACK, A.; RUSNAK J.; BALDWIN, C. Exploring the structure of complex software designs: an empirical study of open source and proprietary code. *Management Science*, v. 52, n. 7, p. 1015-1030, 2006. doi:10.1287/mnsc.1060.0552

MARKUS, M.; MANVILLE, B.; AGRES, C. What makes a virtual organization work? *Sloan Management Review*, v. 42, n. 1, p. 13-26, 2000.

MEIRELLES, P.; SANTOS Jr., C.; TERCEIRO, A.; MIRANDA, J.; CHAVEZ, C., KON, F. A Study of the relationships between source code metrics and

attractiveness in free software projects. In: Brazilian Symposium on Software Engineering (SBES), 2010, Salvador, Brazil. doi:10.1109/SBES.2010.27

MILLER, D.; HARTWICK, J. Spotting management fads. *Harvard Business Review*, v. 80, n. 10, p. 26-27, 2002.

MOCKUS, A.; HERBSLEB, J. Why not improve coordination in distributed software development by stealing good ideas from open source? In: ICSE '02 Workshop on Open Source Software Engineering, p. 35-37, Orlando, FL, 2002.

MOON, J.; SPROULL, L. Essence of distributed work: the case of the Linux kernel. *First Monday*, v. 5, n. 11, 2000.

O'MAHONY, S. Guarding the commons: how community managed software projects protect their work. *Research Policy*, v. 32, n. 7, p. 1179-1198, 2003. doi:10.1016/S0048-7333(03)00048-9

O'MAHONY, S. Competing on a common platform. In: Eclipse Members Meeting, Chicago, Illinois, USA, September 22, 2005.

RIEHLE, D. The economic motivation of open source software: stakeholder perspectives. *IEEE Computer*, v. 40, n. 4, p. 25-32, 2007.

SANTOS Jr., C. Understanding partnerships between corporations and the open source community: a research gap. *IEEE Software*, v. 25, n. 6, 2008.

SANTOS Jr., C.; PEARSON, J.; KON, F. Attractiveness of free and open source software projects. In: European Conference on Information Systems (ECIS), 2010, Pretoria, South Africa.

STEWART, K.; GOSAIN, S. The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly*, v. 30, p. 291-314, 2006.

TIOBE. Programming Community Index for October 2010. 2010. Available at: : http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Accessed: Oct. 10, 2010.

THE ECONOMIST. Open-source business: open, but not as usual. March 16th, 2006. Available at: http://www.economist.com/node/5624944. Accessed: Oct.10, 2010.

THOMPSON, J. *Organizations in Action*. New York: McGraw-Hill, 1967.

VON HIPPEL, E. User toolkits for innovation. *Journal of Product Innovation Management,* v. 18, pp. 247-257, 2001. doi:10.1016/S0737-6782(01)00090-X

VON KROGH, G.; SPAETH, S. The open source software phenomenon: characteristics that promote research. *Journal of Strategic Information Systems*, v. 16, p. 236-253, 2007. doi:10.1016/j.jsis.2007.06.001

XU, B.; QIAN, J.; ZHANG, X.; WU, Z.; CHEN, L. A brief survey of program slicing. In: *SIGSOFT Softw. Eng. Notes*, v. 30, n. 2, p. 1-36, 2005.

WEST, J.; O'MAHONY, S. Contrasting community building in sponsored and community founded open source projects. In: *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Waikoloa, Hawaii, 2005. doi:10.1109/HICSS.2005.166