Journal of Strategic Information Systems 22 (2013) 26–45

Contents lists available at SciVerse ScienceDirect

Journal of Strategic Information Systems

journal homepage: www.elsevier.com/locate/jsis

The attraction of contributors in free and open source software projects

Carlos Santos [a,*], George Kuk [b], Fabio Kon [c,1], John Pearson [d,2]

[a] University of Brasilia, Department of Management, Caixa-Postal: 4320, 70910-900 Brasilia, DF, Brazil
[b] Nottingham University Business School, Nottingham, UK
[c] Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, 207-C, Rua do Matão, 1010, Cidade Universitária, 05508-090 São Paulo, SP, Brazil
[d] Department of Management Information Systems, College of Business, Rehn Hall, Southern Illinois University at Carbondale, Rehn 210-A, Mail Code 4627, Carbondale, IL 62901, USA

ARTICLE INFO

Article history:
Received 25 October 2010
Received in revised form 11 July 2012
Accepted 31 July 2012
Available online 2 November 2012

Keywords:
Attractiveness
Open source
Free software
Preferential attachment
Contributors
Contributions
Software development

ABSTRACT

As firms increasingly sanction an open sourcing strategy, the question of which open source project to undertake remains tentative. The lack of established metrics makes it difficult to formulate such strategy. While many projects have been formed and created, only a few managed to remain active. With the majority of these projects failing, firms need a reliable set of criteria to assess what makes a project appealing not only to developers but also to visitors, users and commercial sponsors. In this paper, we develop a theoretical model to explore the contextual and causal factors of project attractiveness in inducing activities such as source code contribution, software maintenance, and usage. We test our model with data derived from more than 4000 projects spanning 4 years. Our main findings include that projects' set of conditions such as license restrictiveness and their available resources provide the context that directly influence the amount of work activities observed in the projects. It was also found that indirect and unintended contributions such as recommending software, despite of being non-technical, cannot be ignored for project activeness, diffusion and sustainability. Finally, our analysis provide evidence that higher attractiveness leads to more code-related activities with the downside of slowing down responsiveness to address projects' tasks, such as the implementation of new features and bug fixes. Our model underscores the significance of the reinforcing effects of attractiveness and work activities in open source projects, giving us the opportunity to discuss strategies to manage common traps such as the liability of newness. We conclude by discussing the applicability of the research model to other user-led initiatives.

1. Introduction

Free and open source software projects (FOSPs) comprise groups of developers and users geographically dispersed but connected together through shared values and the Internet (Herbsleb and Mockus, 2003; Stewart and Gosain, 2006). Open source developers have traditionally developed software as a hobby but are increasingly being paid and sponsored by commercial and public organizations (Fitzgerald, 2006). To facilitate the development process and promote widespread adoption, the application and its source code are made available on a website, which provides all the information and tools

* Corresponding author. Address: Universidade de Brasília, Campus Darcy Ribeiro, Departamento de Administração, Asa Norte – ICC Norte – 1° Andar, Caixa-Postal 4320, 70910-900 Brasilia, DF, Brazil. Tel.: +55 61 82126224.
E-mail addresses: carlosdenner@unb.br (C. Santos), george.kuk@nottingham.ac.uk (G. Kuk), kon@ime.usp.br (F. Kon), jpearson@business.siuc.edu (J. Pearson).
[1] Tel.: +55 11 3091 6135.
[2] Tel.: +1 618 453 7802.

0963-8687/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jsis.2012.07.004

needed for the software to be used, adapted and improved by the public. Over the years, several projects, such as the Web server Apache, the operating system GNU/Linux, and the browser Firefox, have become widely adopted, demonstrating the viability of the open source production model and its capacity to create high-quality applications. Consequently, corporations have started opensourcing their software assets, aiming to create and capture new business value from this alternative to a more traditional way of developing software (Agerfalk and Fitzgerald, 2008).

The significance of attracting users and developers has been frequently highlighted in the open source literature (Arakji and Lang, 2007; Koch, 2004; Krishnamurthy, 2002; Sen et al., 2008; von Krogh et al., 2003). Each group of users and developers contributes a unique set of complementary resources to FOSP: users provide inputs including bug reports, suggestions of new features, and translation of documentation; and developers implement new features, fix bugs, and deal with sponsors. These roles are reflected in the ways the success of FOSP has been measured (Crowston et al., 2005; Long, 2006), including developers' contribution to source code modularity (Shaikh and Cornford, 2003), number of lines of code generated (Mockus et al., 2000), velocity of closing bugs (Herbsleb and Mockus, 2003), and the number of downloads (Crowston et al., 2004; Krishnamurthy, 2002; Grewal et al., 2006).

Raja and Tretter (2006), Crowston and Scozzi (2002), and Comino et al. (2007) viewed success as the ability of a project to advance through development phases (e.g., from alpha to beta, and from beta to stable). Koch (2004) and Crowston and Howison (2006) suggested the use of community size (i.e., number of members) as a proxy for success. Additionally, Stewart and Gosain (2006) adopted a dependent variable labelled "effectiveness", composed by the abilities to receive inputs and produce related outputs such as fixing bugs and adding new features to the software. These various measures reflect the roles of both input (e.g., bug reports) and output (e.g., bug fixes) producers in open source software development and success. In short, contributions to FOSP come from various groups; developers need users to inform their practices, and users need developers to implement their requests (von Krogh and von Hippel, 2006). Yet, the literature primarily singles out developers as the main contributor to the success of FOSP (Bagozzi and Dholakia, 2006).

The eye-ball metaphor, underpinning the Linus law, underscores two sources of contributions for the success of FOSP. Besides reviewing each others' source code (co-developers), it highlights the role of users as beta-testers in reporting bugs (Raymond, 1999). Bagozzi and Dholakia (2006) noted that experienced users provide support to less experienced individuals, and Bevan (2006) pointed out that users' input is frequently responsible for usability improvements. Users play an important role in the innovation process of FOSP. Although not generally acknowledged in empirical research, users' contributions are critical to FOSP success (Crowston et al., 2003; von Hippel, 2005; Grewal et al., 2006; Bagozzi and Dholakia, 2006), and their contributions have been observed in related industries (Arakji and Lang, 2007).

Another source of contribution is the role of the visitor to FOSP's Web pages. We suspect that this type of contribution is less frequent, but visitors contribute to FOSP through various activities such as reporting broken links or installation problems, and requesting a version not yet available for a particular operating system (or a missing feature that they wish to have). There is also the possibility of a technically-inclined visitor inspecting the source code and posting a suggestion or referring people to the project. Hence, visitors can contribute to FOSP success, even though they may never use or get directly involved in developing the software.

Ye and Kishida (2003) discuss the roles of passive users, readers, bug reporters, bug fixers, peripheral developers, active developers, core members, and project leaders. Though fairly comprehensive, their model implicitly assumes that the contributing roles can only be performed by users and developers, excluding the visitors as a key actor. We expand this view by stating that visitors, users and developers are key resources to FOSP, and that attracting, retaining and inducing them to contribute are core challenges for success. Yet how their roles are interrelated in open source development is relatively unexplored.

To address this limitation, we introduce a theoretical framework around the construct of "attractiveness", which is defined as an array of project values perceived by its potential and actual visitors, users and developers. Our motivation to define and focus on this construct is based on prior research, which has suggested "the need for the company to market the attractiveness of the project and improve its visibility" (Agerfalk and Fitzgerald, 2008, p. 394). This definition implies that attractiveness is a core construct in obtaining the critical resources brought along by different actors, whose motivations, capabilities and likelihood to contribute are influenced by their perceptions of project values and the available resources to the project. In turn, their contributions may increase project values (such as attractiveness) and perpetuate software development as a virtuous cycle. We argue that contributors, including visitors, users and developers contribute in different ways to the shared-innovation process, and that as a collective promotes wide adoption and software development and maintenance. In developing our model, we seek to identify factors that affect the distribution of these resources and ultimately the relative success among FOSP.

This paper presents a model to examine what attracts contributors, creating an environment likely to improve and promote the project sustainably. We attribute the success of FOSP to a user-driven process, involving complementary contributions from various user groups. Overall, our intent is threefold: to understand what makes certain open source projects preferable for usage and improvement; to tease out the rich-get-richer effect; and notably to provide managers the insights into formulating a strategy to compete for external collaborators and complementors. We organize the rest of the paper as follows. First, we use the extant literature of FOSP to frame and develop our model of attractiveness. Next, we describe the methods and present the empirical testing of our model, which is followed by the results and the implications for theory and practice. Finally, our view on future research and the broader conclusions close the paper.

2. Literature review and model development

What makes certain FOSP preferable to others? Prior research concerns the general appeal of a specific project to developers but relatively less to other user groups. As commercial organizations increasingly sanction an open source strategy (Agerfalk and Fitzgerald, 2008; Fitzgerald, 2006), the user uptake will determine the level of sponsorships (Stewart et al., 2006; West and O'Mahony, 2005), and the sustainability of FOSP. This reflects in the strategic thinking of several large open source coalitions such as the Open Source Initiative and the Linux Foundation. One of their core goals is to encourage the development process to be more liberal, by being less restrictive in the terms of use and commercialization potentials of the derivatives, and attract commercial contributions and exploitation. This highlights the significance of bringing developers and different user groups closer together as a collective rather than a collection of disparate entities.

To release the source code to the public creates opportunities for independent and enterprise developers to co-develop with peers and users. This open approach has proved to be successful and exerted a profound effect on the software industry. Yet, despite the highly cited use cases such as GNU/Linux, only few hundreds FOSP have managed to succeed (Krishnamurthy, 2002; Xu and Madey, 2004; Koch, 2004). With the vast majority failing, it seems that the action of releasing source code alone is insufficient to attract and sustain contribution. This uneven distribution of contributors conforms to the behavioural phenomenon of preferential attachment commonly observed in online communities including open source (Xu and Madey, 2004).

Preferential attachment has been used to explain uneven distribution of resources among objects of systems as diverse and complex as those associated with the Internet, neural networks and academic authorships. It is often used synonymously with the rich-get-richer effect, a characteristic of the Pareto or heavy-tailed distribution (Price, 1976; Clauset et al., 2009; Papadakis and Tsionas, 2010). The tendency of overly concentrating on a few gives rise to a general scale-free, power-law distribution of resources to objects (Simon, 1955), which partially defines the growth mechanism of how such systems develop over time (Barabási and Albert, 1999; Barabási, 2005).

Although the parsimonious and general description offered by preferential attachment is intuitively appealing, it is not clear which set of conditions instigate such distribution. Although prior studies have shown that new nodes tend to connect to highly-connected ones, the underlying reasons (including the contextual conditions) of why certain nodes become established as preferential among other nodes are unexplored. The underlying mechanisms of preferential attachment are markedly different across social and physical systems (e.g., Lee et al., 2006; Newman, 2002). The reasons leading to the concentration of contributors on a few projects are not the same nor directly comparable to physiological systems. In physiological systems, the chemistry guides the interaction among proteins and the attraction is often determined at the molecular level, whereas in social systems the attraction is wired to perceptions and intentions. In relation to open source software development, a patch of code is being reused by developers because of its perceived quality and usefulness. This raises further questions of whether the attachment of new resources to a particular object is solely based on the object's "popularity" and/or its unique set of attributes or characteristics. Additionally, little is known of the underlying change process that can elevate the status of a less preferred object. This paper seeks to examine the set of project conditions and the underlying dynamics of what makes an open source project attractive, aiming to inform practice and generate a theory designed to the open source ecosystem, a kind of theory that, being specific, has been overlooked (Keller, 2005).

The success of open source software has been attributed to many developers working under the Bazaar paradigm. However, only a few FOSP managed to build virtuous and productive Bazaar ecosystems (Krishnamurthy, 2002), giving rise to a scale-free network (Xu and Madey, 2004). Researchers have been trying to understand what motivates FOSP developers and drives user-interest, highlighting the importance of trust, knowledge sharing, employment prospects, sponsorship, coordination mechanisms and communication patterns within the communities (von Krogh, 2002; Crowston and Scozzi, 2002; Stewart et al., 2006; Stewart and Gosain, 2006; Crowston and Howison, 2006; Fershtman and Gandal, 2007; Fang and Neufeld, 2009). Nevertheless, most of the prior research has focused on developers to explain projects' activities and success, and less on the role of users and visitors as contributors to the use-value of FOSP.

Users provide relevant problems to be solved such as bug reports and enable the network externalities that not only increase the popularity of the project but also attract new users and sustain developers' contribution. Similarly, visitors of a project Web page, representing "brand" exposure and commercial success (Grewal et al., 2006), can contribute to the network externalities by enhancing the ranking of the visited project (Muffato, 2006). Moreover, visitors may indirectly contribute to the improvement of FOSP by reporting broken links or engaging in R&D activities. Visitors, users and developers are valuable resources to FOSP as they all are, directly or indirectly, contributing to the projects in terms of R&D, marketing, and technology adoption, improvement and diffusion. Yet their conjoint role in open source software development has been overlooked.

2.1. Project activities: the sources of improvement, software maintenance

We have stated that visitors, users and developers are critical resources to FOSP because they are responsible for contributions, which in turn, are the sources of open source software development, improvement and diffusion (Ye and Kishida, 2003). These contributions can be observed through FOSP activities that take place over supporting online tools, such as forums and bug tracks. We refer to project work activities as the inputs as well as outputs that the community provides to the project.

FOSP attractiveness influences work activities in a variety of ways. First, from the perspective that every problem is obvious to someone in software development (Raymond, 1999; Sharma et al., 2002), a straight forward implication of having more visitors, users and developers is that the probability of receiving contributions (inputs and outputs) increases as more people gravitate around the project. Second, from a theoretical point of view, we argue that project attractiveness influences the community motivation, at the individual level, to contribute to the project. Attractiveness is related to popularity and visibility of a project, which increases the motivation of individuals to showcase their abilities (signalling), and improving their reputation within both developers and business communities (Lerner and Tirole, 2002; Roberts et al., 2006). In short, the development community may expect a higher impact from their contributions, as well as higher returns, and is more inclined to contribute to projects they perceive as highly attractive.

Free software projects, as creative enterprises, have agents embedded in an open ecosystem that can inspire and evaluate their contributions and resulting products (Guimera et al., 2005). This "large social milieu", as Nonneke and Preece (2000, p. 6) put, is a source of motivation to contributors and "has far-reaching consequences", affecting people's posting behaviour. The richer this ecosystem, or larger the social milieu, the better for the project, as it becomes more diverse, fostering innovation (von Hippel and von Krogh, 2003; O'Mahony, 2007). A highly-attractive project that brings a sufficient number of developers and users/visitors together has a higher chance of forming a virtuous, cooperative relationship with the users and visitors sourcing a relevant set of problems for the developers to solve. Higher user-interest leads to more development activity (Stewart et al., 2006). In contrast, the use-value of a project without visitors and users is limited to a few developers and has a lesser appeal to the wider public, particularly affecting community intention to contribute.

To explore these theorised benefits of attractiveness on FOSP activities, we focused on four complementary, yet different, measures. First, intending to capture the amount of any direct activity observed in the projects, we gathered and summed the numbers of bug reports, feature and support requests, and patches submitted, under the label of project "activeness". This measure focuses on the volume of ideas and opportunities for project improvement and maintenance that were suggested by its resources. Then from this total sum, we excluded any requests that the project was unable to address, maintaining only those that were properly taken care of ("closed"). We labelled this second measure "effectiveness", and believe it is key for the long-term success of a project, as its absence would condemn an open source software to an outdated state, progressively distant from market's changing interests and demands.

Further, we calculated the ratio of effectiveness over activeness to explore the effects of attractiveness on the likelihood of a contributor input being properly addressed. This represents a project's overall responsiveness to tasks originated in the community. We also computed the average time projects take to address the inputs they have received. The importance of development speed is practical and generates some level of urgency among developers, in that, "[t]he more readily developers can recognize the needs and problems addressed by the project, the more successful the project" (Crowston and Scozzi, 2002, p.10). Details of the construction and acquisition of these measures are discussed in the methods and results sections.

Having discussed the main project activities related to software maintenance and improvement, we argue that there is a cyclical influence between project activities and attractiveness. Resources, influenced by attractiveness, are recruited and act to maintain and improve software, and their recruiting and actions influence project attractiveness. At an empirical level, we will first assess whether there is a direct influence from attractiveness to project activities, a necessary condition to support our theoretical claim of cyclicality. To do that, we formulate our first four propositions, linking attractiveness to project activities. Accordingly, we have:

Proposition 1: FOSP attractiveness significantly influences activeness.
Proposition 2: FOSP attractiveness significantly influences effectiveness.
Proposition 3: FOSP attractiveness significantly influences likelihood of task completion.
Proposition 4: FOSP attractiveness significantly influences time for task completion.

2.2. The causes of attractiveness

As we have discussed, visitors, users and developers tend "to attach" to few "attractive" open source projects, creating a rich gets richer effect (Krishnamurthy, 2002; Xu and Madey, 2004; Koch, 2004). In the following sections, we focus on specific characteristics of FOSP, which define their "condition" that impacts attractiveness, and relate them later onto the preferential attachment mechanism.

2.3. Set of conditions: FOSP characteristics

To explain involvement in open source development, prior research has focused on contributors' intrinsic and extrinsic motivations, like signalling to potential employees (Crowston and Scozzi, 2002; Stewart and Gosain, 2006). We do not challenge this explanation, but intend to expand it, using project as our unit of analysis (Colazo and Fang, 2009). The focus on project allows the examination of which project set of conditions (a set of contextual factors), influences the likelihood of that project being chosen by potential contributors. Project characteristics have been shown in the past to be linked to contributors' motivations and perceived usefulness, impacting development activities and adoption rates (Comino et al., 2007; Crowston and Scozzi, 2002; Fang and Neufeld, 2009; Sen et al., 2008).

The project set of conditions, including license type and application domain, affect their attractiveness throughout their life-cycles. This in turn influences contributor recruitment and contribution generation, which take place as people browse for and find out about software, as well as consider contributing to a project after they have become "a resource of". Some people may report a failed-attempt to download or install the software; others may post bugs or develop features for applications in initial stages. Moreover, open source advocates might prefer to use, be associated with, and/or refer people to GPL-licensed applications. FOSP's application domain helps define the target population size, benefiting those in larger domains. For example, it is highly likely that there is a smaller demand for compilers in comparison to office suites. Accordingly, we expect that, all other things being equal, compilers are less likely to attract contributors than office suites, impacting the recruitment of contributors, project's activity level and its probability of receiving other indirect contributions. Similarly, application domain relates to contributors' profile, as users of compilers tend to be technically-inclined software developers, and thus are more capable of contributing or inspecting source code than office suite users.

A project's set of conditions affects its ability to attract resources and activity levels via different mechanisms. First, these conditions offer an opportunity for people to prefer a specific application context when choosing what to adopt and contribute to. Second, they specify the boundaries of competition for recruiting a limited amount of resources available in the marketplace. Finally, FOSP set of conditions influences their visitors, users and developers likelihood of contributing, as their community profile (e.g., computer skills) depends on who they target (programmers or end-users). These project conditions work together to influence people's perception of FOSP attractiveness, which affects recruitment of resources and generation of contributions.

Several distinctive and empirically observable elements constitute the project set of conditions, working in tandem with each other as a set to influence people's perceptions of project attractiveness in a cyclical and dynamic manner. As such, the task to hypothesize in advance the direction of influence of all their combinations on our variables of interest would be unmanageably complex. Accordingly, we opted to theorize in an exploratory manner, stating that each project condition has an influence, which is not independent of the state of another condition as they act together as "the context", on project's resource availability and the amount of activity observed. Next, we present our propositions relating each FOSP characteristics to attractiveness and work activities, and then elaborate on how feedback works within the preferential attachment mechanism along with project conditions.

2.4. Type of license

What allows the classification of a project as open source and/or free software is the license. FOSP licenses regulate what can and cannot be done with the software, its source code and derivative works, influencing its range of use and distribution, and notably its intellectual property (Agerfalk and Fitzgerald, 2008; Santos et al., 2011). Under the General Public License (GPL) the source code of the new derivatives have to remain open and are not allowed to be redistributed as proprietary software. Whereas the Mozilla Public License and the Eclipse Public License permit greater interaction with proprietary software and provide greater commercial freedom (Fershtman and Gandal, 2007).

In the literature, open source software licenses are commonly grouped based on their restrictiveness (Fershtman and Gandal, 2007; Lerner and Tirole, 2005; Sen et al., 2008; Stewart et al., 2006). In general, there are three levels of restrictiveness: (1) do not allow combined compilation with proprietary software and force a derivative work to have the same license as the original (Strong-Copyleft); (2) force derivative works to have the same license but allow combined compilation with proprietary software (Weak-Copyleft); and (3) do not impose any of these restrictions (Non-Copyleft). The influence of type of license on FOSP has appeared in different ways. Lerner and Tirole (2005) have examined how license choice is associated with a project's audience, developers or end-users. It has been reported that license choice is associated with the amount of developer activity, user interest, and individual intention to contribute (Fershtman and Gandal, 2007; Stewart et al., 2006; Santos et al., 2011). Sen et al. (2008) pointed out that license restrictions impact perceived usefulness and the visibility of software. Colazo and Fang (2009), using social movement theory, argued that license type, FOSP size, and development speed are linked. These findings together suggest that license type affects project work activities and thus attractiveness of open source software. That is, the attraction of resources is influenced by the license a project adopts. For instance, whilst open source advocates tend to use GPL applications, for-profit organizations tend not to combine their proprietary codes with open source ones because this will effectively give away their property rights. Thus, we have:

Proposition 5.1: Type of license significantly influences FOSP attractiveness.
Proposition 5.2: Type of license significantly influences FOSP activeness.
Proposition 5.3: Type of license significantly influences FOSP effectiveness.
Proposition 5.4: Type of license significantly influences FOSP likelihood of task completion.
Proposition 5.5: Type of license significantly influences FOSP time to complete tasks.

2.5. Type of user

Software aid processes that are managed by different types of users. These types of users can be end-users (e.g., browsing the web), advanced end-users (e.g., developing a database), system administrators (e.g., creating backups), and developers (e.g., writing software). The influence of type of user, or audience, on FOSP activities has been discussed in the literature

(Crowston and Scozzi, 2002; Stewart et al., 2006; Fershtman and Gandal, 2007). FOSP aiming at technically inclined users are more likely to find contributors among their users, and FOSP for less technically inclined users have a larger audience and, thus, more chances of finding users and developers (Comino et al., 2007; Crowston and Scozzi, 2002). Some FOSP "have a greater number of potential developers in the community than others do", says Johnson (2002, p. 664). Additionally, Stewart et al. (2006, p. 136) hypothesized the influence of type of user on both user-interest and development activities, stating that "[t]hose targeted at a developer audience may attract greater development activity or be less appealing to users". Similarly, projects targeted at developers have been found to be more active than those targeted at system administrators, which outperform projects for end-users (Crowston and Scozzi, 2002). Finally, the number of software developed for different types of users are not equal, influencing the amount of available resources in the market and competitiveness for contributors and their contributions.

Although the literature suggests that recruitment and project work activities are affected by type of user, it is not clear how these will affect the conjoint attraction and role of visitors, users and developers in FOSP, nor in the presence of and interacting with the other FOSP characteristics. We formulate the following:

Proposition 6.1: Type of user significantly influences FOSP attractiveness.
Proposition 6.2: Type of user significantly influences FOSP activeness.
Proposition 6.3: Type of user significantly influences FOSP effectiveness.
Proposition 6.4: Type of user significantly influences FOSP likelihood of task completion.
Proposition 6.5: Type of user significantly influences FOSP time to complete tasks.

2.6. Application domain

An application supports certain processes for its users. To cover these processes, FOSP applications are classified according to domains, or project categories, such as genealogy, payroll, chat, browser and games (Crowston and Scozzi, 2002). The application domain restricts, where a software competes for contributors and their contributions; it is the software industry or niche (Jaisingh et al., 2008). For example, Firefox generally does not compete for users with R, as they operate in distinct arenas. Potential contributors typically look for projects to work on in a specific domain (Johnson, 2002), and so it is likely that they will first select one over the others. Raymond (1999) uses email client projects to illustrate how self-selection works in practice, underlining how FOSP are competing among themselves for contributors and contributions within each application domain.

Additionally, the application domain has been discussed in the context of "technical sophistication" (Comino et al., 2007). As such, more technically sophisticated domains like "compilers" are more likely to receive substantial contributions from its users, as they are developers and thus have the required technical skills (Crowston et al., 2005; Crowston and Scozzi, 2002). In this line of reasoning, there is an overlapping with the effects of the type of user condition we discussed previously. However, it is still important to control for the application domain in addition to the type of user as projects of the same application domain can still target different types of users. For example, one software can provide an intuitive graphical user interface, whereas a different application may require the user to write algorithms and be familiar with a particular language to perform the same tasks through a command-line interface. The software application domain has been hypothesized to be associated with both user-interest and development activity (Stewart et al., 2006). Therefore, FOSP attractiveness and work activities should be affected by their application domain as well as by the characteristics of the other projects operating *within* that domain. Thus, in the context of our model, we have:

Proposition 7.1: Application domain significantly influences FOSP attractiveness.
Proposition 7.2: Application domain significantly influences FOSP activeness.
Proposition 7.3: Application domain significantly influences FOSP effectiveness.
Proposition 7.4: Application domain significantly influences FOSP likelihood of task completion.
Proposition 7.5: Application domain significantly influences FOSP time to complete tasks.

2.7. Stage of development

Software engineers generally classify applications according to their stage of development, in a life-cycle fashion. These stages in the life-cycle are planning, pre-alpha, alpha, beta, production and mature. FOSP make their software stage available to the public, informing their maturity level condition and influencing decisions to adopt and contribute (Crowston and Scozzi, 2002; Raja and Tretter, 2006), as well as the level of development activity (Stewart et al., 2006).

There are known links between stage of development, project size and contributors' technical skills (Comino et al., 2007). Furthermore, stage of development can be used as a strategy by project managers, for a beta release may be seen as an invitation for contributions from the community, as well as for users to try a "new" application. At the same time, it is probable that users prefer mature software over applications at the alpha level. Accordingly, these observations together suggest that the stage of development condition has to be incorporated in our model, influencing how attractive FOSP are and their level of work activities. Thus, we have:

Proposition 8.1: Stage of development significantly influences FOSP attractiveness.
Proposition 8.2: Stage of development significantly influences FOSP activeness.
Proposition 8.3: Stage of development significantly influences FOSP effectiveness.
Proposition 8.4: Stage of development significantly influences FOSP likelihood of task completion.
Proposition 8.5: Stage of development significantly influences FOSP time to complete tasks.

2.8. The feedback effect of project activities, software maintenance

FOSP change their contributor-base and the amount of contributions received over time, distinguishing between initial and sustained contribution. Fang and Neufeld (2009) demonstrated how important is for contributors to learn and construct a shared identity with the project to sustain their motivation to participate over time. Roberts et al. (2006) showed how past performance rankings of developers influence their future motivation to contribute. Ye and Kishida (2003) pointed out that there is a co-evolution in FOSP, with contributors' motivations being affected by their contributions to the project, which can result in status promotion. Yet, this stream of research has been mostly restricted to the contributor (developer) as the unit of analysis, not the project. Adding to this literature, we propose a mechanism through which contributions are sustained at the project level. Specifically, we differentiate direct (project activities) from indirect contributions (attractiveness self-reinforcing effect).

One source of FOSP improvement and software maintenance is through the work activities carried out by developers. These activities, which are direct contributions, can translate into changing project attractiveness. For example, visitors, users and developers might have requested a series of new features that were later implemented, directly affecting the perceived usefulness of the software and the potential benefits to new participants in the long-run (Stewart et al., 2006). In contrast, a "buggy" software, with many unreported and thus not addressed problems, will adversely affect project attractiveness. When a community crowds around a project, direct participation and improvements will be visible to the public. This visibility will enhance the value of the project/software, attracting more visitors, users and developers. Notwithstanding, the effects of a project conditions on attractiveness and work activities do not cease to operate, influencing the likelihood of receiving a contribution in any case.

Our perspective on the relationships of project conditions, attractiveness and work activities over time can be summarized illustratively as follows. To begin with, a recently created project has low attractiveness as it is only in the planning stage of development. The only contributors are the project creators and a few of their colleagues, who could test the application after a release. Knowing that, the creators work by themselves, designing the source code structure and defining the application features, writing code and implementing functionalities, developing the user-interface and managing all other project tasks. By doing that and performing what we are calling here project work activities, they release an alpha version of their application, which then improves the attractiveness of the project via changing the initial set of project conditions. Now, the creators can send an email to their colleagues saying that there is a new software for them to test and give feedback. The colleagues download the application and start using it, spotting many problems and realizing that there is a lot more that the application should do in order to be professionally adopted by them. As the colleagues report their impressions, increasing the project index of activeness, the creators have the opportunity to address these relevant issues and improve the project attractiveness even further based on their communication with the users (work activities). If they are able to do so, the application will advance again into the next stage of development, their colleagues will become actual users and feel motivated to report more desired features and/or potential problems. This forms a virtuous cycle, where attractiveness is enhanced both by the maturing application condition, which positions it better against similar applications, and the addressing of users' demands, which affects software quality and project attractiveness, fostering diffusion.

In short, FOSP attractiveness is enhanced through direct contributions to the project, but their set of conditions have an independent influence on attractiveness as well. This proposition is a core assumption of the open source movement and takes the form of a loop-mediation in our model, from attractiveness to project activities and, then, back to attractiveness, controlling for the set of conditions. More formally, we have:

Proposition 9: Past project work activities significantly influence future FOSP attractiveness above and beyond the effects of projects' set of conditions.

2.9. Attractiveness self-reinforcing effect, indirect and unintended contributions

However important, the role of visitors, users and developers in enhancing their projects attractiveness is not restricted to contributing to its software source code or related material, such as website content, support provision and documentation. Contributions to FOSP occur in a variety of ways, including indirect, and even unintended, ones. Stewart et al. (2006), for example, highlighted the contributing role of users via word-of-mouth recommendations, influencing future user-base size and, potentially, developers' intention to contribute, configuring an important indirect contribution. By extension, we expect a similar behaviour from visitors and developers.

To understand the motivations for this behaviour, Nonnecke and Preece (2003, p.126), studying online groups, pointed out that lurkers, people who do not actively contribute (post), by reading daily messages, develop a strong sense of community. This sense of community leads to the dissemination of information, such as "contacting individuals [...] and

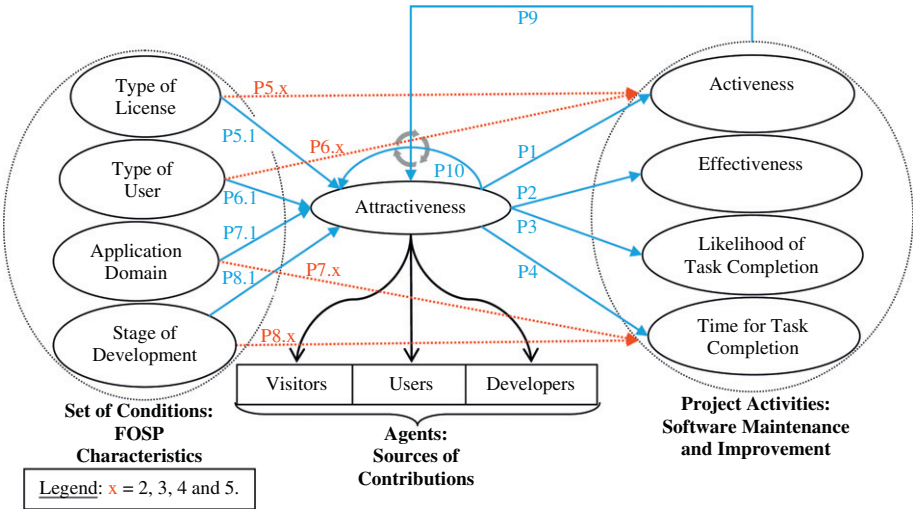

Fig. 1. Theoretical model for the attraction of contributors in FOSP.

introducing others to a group". We find it likely that a similar mechanism takes place in open source projects, as the effort of their contributors have been found to be influenced by common values, beliefs and norms (Stewart and Gosain, 2006). In support to that, Mozilla Firefox's users once contributed their own money to advertise the browser in *The New York Times* (Bagozzi and Dholakia, 2006). Similarly, users may freely create public online content related to open source projects, recommend them to their employers or employees, and wear supporting apparel; visitors can share a project website with their friends on social networks, recommend it to someone by email, donate money, write reviews and rate the project on its repository; and, finally, developers can organize workshops, mentoring programs and "install fests". These represent important, however indirect, contributions to project attractiveness, as they help to build a reputation in the marketplace.

Additionally, there is a more discrete and perhaps unintended type of contribution the community performs. We refer to this as the project density effect, or the silent contribution in Nonneke and Preece's (2000) terms, and its causal chain goes as follows. A visitor, by visiting, and a user, by downloading, enhance a project position in online ranks, increasing its visibility and, thereby, influencing its attractiveness. Especially visitors, by representing website traffic, can help a project find business sponsors, which can bring new developers, and raise money through the sale of ads, which can then be used to publicize the project or hire new developers and sustain work.

All these things together suggest that attractiveness, by affecting the number of available visitors, users and developers and their likelihood of performing indirect contributions, will influence the amount of these sources of contributions in the future, via an effect on attractiveness itself. Again, it is noteworthy that these relationships take place in the presence of projects' set of conditions, which continue to affect attractiveness regardless of an increasing visibility. For example, an organization intending to adopt an open source software may attend a workshop and find out that its license does not allow it to be merged with the organization proprietary applications already in place. Thus, in the context of our model, we formally have:

Proposition 10: FOSP past attractiveness significantly influences future attractiveness above and beyond its effect via project work activities, controlling for projects' set of conditions.

Together, Propositions 9 and 10 represent the specific mechanism through which we claim preferential attachment takes place in open source projects, making the attractive projects even more attractive and leaving the new and unattended project in an difficult situation, struggling to evolve and survive (i.e., liability of newness). With these propositions, we can present of our theoretical model, depicted in Fig. 1, and are now able to move to the discussion of the methods required to evaluate its plausibility, empirically.

3. Methods

The characteristics of our propositions, such as the variety of relationships proposed and the presence of a latent construct, led us to evaluate variables' effects using structural equation modelling (SEM). SEM is a technique that allows researchers to test complex models with simultaneous equations (Chin, 1998; Kelloway, 1998). The structural model developed by us represents a cross-sectional evaluation of variables' theorised influence on the ones that do not represent themselves at a different period in time, effectively testing Propositions 1–8.5 with the data. This model test variables' observed associations at a point in time, a necessary condition for our propositions to find support in the empirical analysis.

As Propositions 9 and 10 predict an association of attractiveness from the past with its future value, we decided to develop an independent test of their plausibility. We discuss the methods for testing these propositions later.

The structural model was tested with data from the largest FOSP repository, SourceForge.net. Data about SourceForge.net projects was readily available at the University of Notre Dame, which allowed us to collect three samples (4693 projects in January of 2006, 4500 in 2007, and 4507 in 2008) to compare and improve our confidence in the results. Data was separated into files by period, and projects were filtered out according to the criteria of: (1) not having inadmissible values on the variables, e.g. negative date of creation; (2) having more than one download and member (to increase confidence that the project contains software and source code posted); and (3) not having "inactive" flagged, which indicates that the project is no longer active, or perhaps has never been.

To capture the amount of available resources to a project at a point in time, we used three specific variables: number of hits the project website had as a proxy for number of visitors; the number of times its application was downloaded as a proxy for the number of users; and the number of registered members the project has as a proxy for developers. Next, a factor or component can be extracted from these three measures, which have been found to be significantly and positively correlated with each other in previous studies (Krishnamurthy, 2002). This factor offers a single and straightforward measure of resources' availability to the project.

Nevertheless, these proxies as a composite measure of available resources of a project are not without drawbacks. The use of number of visits as a proxy for visitors, downloads for users, and members for developers is not perfect. The proxies for visitors and developers tend to be biased upward, as a visitor may "hit" a website more than once and a member may never directly develop source code, being in charge of an administrative role such as the assignment of tasks to developers, for example. In its turn, number of downloads tend to bias its representation in a more complex way, as software may be downloaded but never used (upward-bias), or a user may acquire the software through some other form, such as from a GNU/Linux distribution, or even download it more than once (downward-bias). However, one cannot be a visitor without visiting, one of the main ways of becoming a user of open source is downloading the software, and most members are source code developers. And given that we are interested in understanding the distributions of these measures rather than their absolute values, we consider them useful for research purposes. In defence of their usefulness, researchers have used downloads to count users and user-interest (Stewart et al., 2006; Wiggins et al., 2009), and members to represent developers (Fershtman and Gandal, 2011). The FOSP literature frequently adopts these measures (e.g., Crowston et al., 2005; Raja and Tretter, 2006; Stewart and Gosain, 2006; Sauer, 2007).

To explore the role of project work activities and software maintenance, we observed activeness as the sum of (1) bug reports, (2) support requests, (3) feature requests, and (4) patches submitted; effectiveness as the sum of (1)–(4) that were closed (resolved/approved); likelihood of task completion as effectiveness divided by activeness; and time for task completion as the average time a project took to close its tasks.

In accordance with the discussion on the skewed-distribution of resources to objects and previous research on FOSP, hits, downloads and members were found to have skewness values outside the range of normality, and standard deviations much greater than their averages (see Table 1, for the 2008 sample characteristics). Activeness, effectiveness and time for task completion were found to have a similar pattern. Likelihood of task completion was not, but we decided to transform it along with the others that were log-transformed for linearisation to keep the results interpretation consistent. Likelihood was transformed into its inverse-sine-square-root, which tends to render more normally distributed data from variables in the form of proportion (effectiveness divided by activeness) (Crowston and Scozzi, 2002). These transformations for linearisation

Table 1
Descriptive statistics.

	Minimum	Maximum	Mean	Std. deviation	Skewness	
					Statistic	Std. error
Hits	1	1423193	3196.07	32693.98	29.15	.04
Ln.hits	.00	14.17	4.9918	2.32	.16	.04
Downloads	7	160385573	230484.93	2884297.85	41.36	.04
Ln.downloads	1.95	18.89	9.35	2.11	.35	.04
Members	2	374	6.43	9.81	14.89	.04
Ln.members	.69	5.92	1.49	.762	1.01	.04
Activeness	1.00	122375.00	146.38	1874.71	61.67	.04
Ln.activeness	.00	11.71	3.13	1.72	.35	.04
Effectiveness	1	119282	112.02	1811.48	63.29	.04
Ln.effectiveness	.00	11.69	2.49	1.83	.57	.04
Likelihood	.01	1.00	.61	.27	−.24	.04
Arc.likelihood	.01	1.57	.74	.42	.56	.04
Average	21.00	165888141.00	11373856.98	15286540.08	3.40	.04
Ln.average	3.04	18.93	15.30	1.93	−2.2	.04
Life.span	1118.90	3010.00	2111.16	500.50	−.059	.04
Ln.life	7.02	8.01	7.62	.25	−.45	.04
N (listwise)	4507					

are frequently reported in open source research (e.g., Comino et al., 2007; Crowston and Scozzi, 2002). We measured the categorical variables, FOSP's set of conditions or characteristics, as dummies.

Sets of dummy variables were created to represent each FOSP condition/characteristic (that is, type of license, type of user, application domain and stage of development). For type of license, we used four dummies, based on Lerner and Tirole (2005): (1) no restriction or non-copyleft; (2) restriction of modification or weak-copyleft; (3) restriction of modification and use or strong-copyleft; and (4) dual-licensing, when a software is licensed on different types of licenses. Type of user required five dummies, application domain required 18, and stage of development 6 (see Table 3). Finally, a control variable was included, life-span measured in days, for its effects on various variables related to FOSP success have been previously reported (Crowston and Scozzi, 2002; Crowston et al., 2005; Stewart et al., 2006; Fershtman and Gandal, 2007).

To generate the results, we used the multisample-SEM capability of EQS 6.1 (Byrne, 2006). Data was entered in its raw form. For the fitting criterion (coefficient estimation), priority should be given to maximum likelihood (ML) when sample size is large (Bentler, 1989; Kline, 1998). Accordingly, we adopted ML, the most common method used by structural modellers (Anderson and Gerbing, 1998; Hair et al., 2006). The results of this statistical analysis are presented after we describe the methods for testing Propositions 9 and 10.

The propositions related to preferential attachment were tested using the process described by Preacher and Hayes (2008) to establish mediation in a single model with multiple mediators and control variables. Their procedure is based on the well-known conceptual description of Judd and Kenny (1981) and Baron and Kenny (1986). However, Preacher and Hayes' procedure has the advantages of testing whether the mediators have an effect as a set and their individual effects in the presence of the other mediators and controls, reducing the likelihood of parameter bias due to the omission of variables (2008). Of course, one should make sure that the mediators included in the model are not conceptually overlapping and highly-correlated. Moreover, Preacher and Hayes' (2008) procedure includes bootstrapping to generate confidence intervals and does not assume normality of the sampling distribution of the indirect effects. According to Preacher and Hayes, these features make the procedure "far superior" to the traditional Sobel test.

To test Propositions 9 and 10, we used data from the Notre Dame 2008 sample and collected additional data of the "future" directly from the SourceForge.net website through a Web crawler our team developed. This new data allowed us to embed some measurement independence in using data of the same variables over time. The final content of the variables in the mediational model is as follows. First, we have the independent variables hits, downloads and members up to 2008. From these three, we calculated one independent variable using the regression weights extracted from the principal component analysis (Mardia et al., 1980) using the statistical package SPSS. Second, the mediators selected for inclusion were effectiveness, likelihood and time for task completion (from the Notre Dame sample). Activeness was left out as it shares a great deal of conceptual overlap with effectiveness, which represents what the community has indeed addressed (source code included, bugs fixed, etc.) and is therefore more interesting to evaluate its effects on attractiveness. Third, there are the dependent variables hits, downloads and members related to the period of January, 2008–December, 2009. Finally, we included all dummies, which represent the projects' set of conditions, and life-span as control variables (covariates). Having gathered all data needed, we used Preacher and Hayes' script (http://www.afhayes.com/public/indirect.sbs) to generate the results, using 5000 bootstrapping resamples to calculate intervals as per their recommendation (Preacher and Hayes, 2008).

4. Results of the structural model

4.1. Descriptive statistics

The Notre Dame SourceForge.net sample is of 149,542 projects in January/2006, 179,867 in 2007, and 143,591 in 2008. After filtering, these numbers were reduced to 4693, 4500 and 4507, respectively. The average project in the 2008 sample received 3196 hits, 230,484 downloads, had six members, and was over 5 years old. Also, the average project produced 146 tasks (inputs received), closing or addressing 112 of them (61%) in an average of 132 days. In their raw form, every variable but life-span and likelihood has a standard deviation greater than the average, and Skewness statistic outside the interval commonly accepted as normal $[-1,1]$. The log-transformations were effective on substantially reducing skewness and returning standard deviations smaller than averages (see Table 1). To illustrate the FOSP characteristics (set of conditions), we noted that the 2008 sample has 1279 projects with licenses that do not impose any restriction to the source code; 2632 are aimed at end-users; 222 were listed under the database application domain; and 1972 projects had their software in the beta stage of development.

4.2. Latent construct reliability

The amount of resources available to FOSP was measured using three variables or indicators and, therefore, its internal consistency had to be assessed. We did so via Cronbach's alpha. The construct scored 0.705 in 2006, 0.711 in 2007, and 0.714 in 2008 (Table 2). Alpha values greater than 0.7 are considered acceptable (Hair et al., 2006; Peterson, 1994; Rutner et al., 2008). Nevertheless, the use of alpha to assess reliability of latent variables is questionable as it requires unrealistically stringent assumptions (Byrne, 2006). Accordingly, we took into consideration EQS reliability coefficient Rho for the overall model as well, which were all above 0.9 and, therefore, indicated appropriate reliability. These results support our claim that

Table 2
Model-to-data fit indices for model selection.

	Model with equality constraints			Model without equality constraints			Comparison
	2006	2007	2008	2006	2007	2008	
Sample size	4693	4500	4507	4693	4500	4507	
Cronbach's alpha	0.705	0.711	0.714	0.705	0.711	0.714	
EQS Reliability coefficient rho	0.952	0.955	0.954	0.952	0.955	0.954	
Chi-Square	3042.7 (585 Degrees of Freedom)		2849.4 (237 Degrees of Freedom)			193.3 (348 d.f.)	
P-value for chi-square	<0.01		<0.01			Same	
Model Fit (CFI)	0.979		0.978			0.001	
B–B Normed fit index	0.974		0.976			−0.002	
Root mean square residual	0.021		0.020			0.001	
RMSEA	0.018 – (90% C.I.: 0.017, 0.018)		0.028 – (90% C.I.: 0.027,0.029)			0.010	
			Chi-square critical value (0.05; 348 d.f.):			392.501	
			Decision (given 193.3 < 92.5):			Favour model with constraints	

hits, downloads, and members are likely to have common causes, or are empirical expressions of, at least, one single construct (e.g., attractiveness).

4.3. Overall model-to-data fit

The equation coefficients were calculated both (1) independently, sample by sample and (2) forced to be equal across samples 2006, 2007 and 2008. This strategy was utilized to reduce sampling fluctuations that may obscure effects and bias results (Maitland et al., 2001). The difference in chi-square between the models is 193 (DF = 348; $p > 0.9$). Therefore, the null hypothesis is not rejected, indicating that the more restrictive model (2) produces at least as good a fit as model (1). Also, the two models have similar model-to-data fit indices (Table 2). Thus, the restrictive model (2) is preferable as it is more parsimonious (Mulaik, 2005). Fit tests and indices check if the pattern of covariances are consistent between the specified model and the data (Dow et al., 2008). A "good" fit is a necessary condition to analyse SEM models, that is, CFI greater than 0.95, RMSEA smaller than 0.05, and insignificant chi-square. The constrained model (2) with RMSEA of 0.018, CFI of 0.979 and chi-square of 3042.7 (DF = 585; $p < 0.01$) is acceptable and has good fit except for the chi-square. Nevertheless, the chi-square test is known for its sensitivity to sample size and number of parameters modelled (Kaplan, 2008). Among available fit indices, "RMSEA is relatively [the] most stable" and insensitive to sample size (Yuan, 2005, p. 141). Therefore, we consider the constrained model proper for further analysis.

4.4. Testing the independent effects of variables

The framework developed and coded in EQS for empirical evaluation of Propositions 1–8.5 requires the analysis of five equations. In regression terms, the first equation has attractiveness (F1) as the dependent variable, explained by 33 dummy variables plus life-span (Table 3). As it turned out, life-span is a positive and statistically significant predictor of attractiveness. Thus, the importance of including life-span in FOSP analysis is supported. Also, out of four dummies used to study type of license, one (dual_licensing) was found to influence attractiveness significantly. To register software under licenses with different restrictions has been popular in FOSP with commercial intentions (Santos, 2008; Watson et al., 2008), affecting attractiveness positively. Therefore, we fail to reject P5.1. In general, our rationale to decide whether to reject propositions was that if at least one dummy (e.g., dual_licensing) of a set (e.g., type of license) was significant, then the effect of type of license would have been detected, supporting the proposition.

In relation to type of user, projects for end-users and developers have higher attractiveness, whereas those aiming at others have lower (fail to reject 6.1). Projects listed as multimedia, printing, security and system (application domain) have higher attractiveness, whereas those in database, education, other, scientific and sociology have lower (fail to reject 7.1). Specifically, projects should avoid being listed as others as it hinders attractiveness the most.

Stage of development significantly influences attractiveness in all its six possibilities (fail to reject 8.1). Results indicate that the initial stages of projects (planning, pre-alpha, and alpha) affect attractiveness negatively, whereas advanced stages (beta, production, and mature) enhance attractiveness increasingly, respectively. So, mature projects are more attractive, and to release software in initial stages tend to be ineffective. This first equation explained 22.4% (in 2006), 17.3% (2007), and 15.2% (2008) of attractiveness' variance.

The other four equations indicate how attractiveness influences FOSP work activities. Namely, activeness (F2), effectiveness (F3), likelihood of task completion (F4), and time for task completion (F5). Attractiveness is a significant, and the most important, predictor of the four variables and thus we fail to reject Propositions 1–4. It positively influences activeness and effectiveness, just as Raymond (1999) predicted and many others followed (e.g., Stewart and Gosain, 2006). However, higher attractiveness is associated with smaller likelihood to complete tasks and greater time to complete them. The impact of type of license (both_restrictions) is significant and negative on activeness and effectiveness. Moreover, licenses with both restrictions influence likelihood of task completion positively, suggesting that projects under GPL are more likely to address the

Table 3
Structural equations results.

Exogenous variables	F1-Attractiveness 2006, 2007, and 2008		F2-Activeness 2006, 2007, and 2008		F3-Effectiveness 2006, 2007, and 2008		F4-Likelihood of task completion 2006, 2007, and 2008		F5-Time for task completion 2006, 2007, and 2008	
	Coef.	*T*-Statistic	Coef.	*T*-Statistic	Coef.	*T*-Statistic	Coef.	*T*-Statistic	Coef.	*T*-Statistic
	Endogenous variables									
F1-Attractiveness[a]	–	–	3.699	43.586*	3.568	41.946*	−0.208	−14.36*	1.586	25.056*
F2-Activeness[a]	–	–	–	–	–	–	–	–	–	–
F3-Effectiveness[a]	–	–	–	–	–	–	–	–	–	–
F4-Likelihood of Task Completion[a]	–	–	–	–	–	–	–	–	–	–
F5-Time for Task Completion[a]	–	–	–	–	–	–	–	–	–	–
F6-Life-Span[a]	0.172	19.078*	0.006	.161	0.049	1.196	0.031	2.645*	0.72	14.546*
F7-Type of License(No-Restriction)	−0.013	−1.183	0.028	.609	0.065	1.297	0.023	1.557	0.048	0.784
F8-Type of License(Mod-Restriction)	0.01	1.2	−0.041	−1.165	−0.023	−0.593	0.02	1.777	0.02	0.415
F9-Type of License(Both-Restrictions)	0.014	1.622	−0.136	−3.562*	−0.085	−2.036*	0.028	2.254*	−0.026	−0.505
F10-Type of License(Dual-Licensing)	0.027	2.363*	−0.013	−0.254	−0.052	−0.952	−0.028	−1.788	−0.065	−0.983
F11-Type of User(End-Users)	0.093	14.8*	−0.057	−2.179*	−0.092	−3.239*	−0.011	−1.287	−0.036	−1.027
F12-Type of User(Developers)	0.025	3.982*	0.008	0.294	0.025	0.847	0.009	1.082	0.143	3.918*
F13-Type of User(System-Admins)	−0.01	−1.499	0.024	0.858	0.006	0.191	−0.015	−1.708	0.017	0.436
F14−Type of User(Others)	−0.019	−2.227*	0.048	1.336	0.049	1.24	0.011	0.929	−0.145	−3.010*
F15-Type of User(Advanced-End-Users)	0.007	.667	−0.039	−0.944	0.004	0.088	0.054	4.053*	−0.044	−0.783
F16-Application Domain(Communications)	0.001	.109	0.046	1.081	0.004	0.095	−0.039	−2.862*	−0.022	−0.38
F17-Application Domain(Database)	−0.056	−4.898*	0.083	1.693	0.071	1.314	−0.013	−0.842	0.036	0.546
F18-Application Domain(Desktop)	0.019	1.294	−0.129	−2.067*	−0.113	−1.645	−0.013	−0.652	−0.055	−0.652
F19-Application Domain(Education)	−0.059	−4.015*	0.212	3.388*	0.28	4.067*	0.037	1.83	0.158	1.873
F20−Application Domain(Games)	0.003	0.273	−0.157	−3.49*	−0.146	−2.971*	0.017	1.194	−0.027	−0.453
F21-Application Domain(Internet)	0.011	1.323	−0.042	−1.126	−0.018	−0.431	0.034	2.837*	−0.011	−0.216
F22-Application Domain(Multimedia)	0.061	5.124*	−0.234	−4.437*	−0.283	−4.914*	−0.03	−1.817	0.132	1.879
F23-Application Domain(Office)	0.016	1.224	0.256	4.519*	0.232	3.726*	−0.035	−1.914	0.08	1.055
F24-Application Domain(Other)	0.083	−5.437*	0.022	0.334	0.036	0.505	0.006	0.294	0.238	2.731*
F25-Application Domain(Printing)	0.092	3.262*	−0.227	−1.887	−0.181	−1.364	0.041	1.066	−0.138	−0.85
F26-Application Domain(Religion)	0.025	0.644	0.293	1.754	0.328	1.783	0.048	0.894	0.739	3.284*
F27-Application Domain(Scientific)	−0.039	−3.418*	0.043	0.855	0.099	1.783	0.003	0.201	0.109	1.619
F28-Application Domain(Security)	0.066	4.209*	−0.148	−1.088	−0.038	−0.518	0.002	0.084	−0.005	−.054
F29-Application Domain(Sociology)	−0.131	−3.234*	0.585	3.338*	0.537	2.792*	−0.124	−2.222*	0.536	2.278*
F30-Application Domain(Software-Dev)	0.008	1.034	0.02	0.576	0.027	0.718	−0.007	−0.651	−0.036	−0.782
F31-Application Domain(System)	0.035	3.203*	−0.268	−5.544*	−0.33	−6.268*	−0.034	−2.224*	−0.012	−0.184
F32-Application Domain(Terminals)	−0.003	−0.077	−0.148	−.867	−0.213	−1.138	−0.045	−0.822	0.456	1.997*
F33-Application Domain(Text-Editors)	0.025	1.456	0.053	0.684	−0.078	−0.926	−0.062	−2.549*	0.024	0.236
F34-Stage of Development(Planning)	−0.036	−4.181*	0.02	0.556	−0.005	−0.135	−0.027	−2.311*	−0.024	−0.495
F35-Stage of Development(Pre-Alpha)	−0.084	−8.978*	0.034	0.864	0.077	1.772	0.033	2.592*	−0.265	−4.994*
F36-Stage of Development(Alpha)	−0.024	−3.106*	−0.023	−0.694	−0.015	−0.4	0.009	0.802	−0.152	−3.397*
F37-Stage of Development(Beta)	0.03	4.512*	0.12	4.210*	0.16	5.106*	0.007	0.737	−0.048	−1.242
F38-Stage of Development(Production)	0.162	21.72*	0.12	4.026*	0.231	7.053*	0.05	5.248*	0.161	4.031*
F39-Stage of Development(Mature)	0.186	13.961*	0.097	1.746	0.195	3.194*	0.071	3.979*	0.152	2.037*
	Variance explained per sample									
R-Squared	2006; 2007; 2008		2006; 2007; 2008		2006; 2007; 2008		2006; 2007; 2008		2006; 2007; 2008	
	0.224; 0.173; 0.152		0.448; 0.476; 0.49		0.394; 0.414; 0.416		0.028; 0.025; 0.03		0.136; 0.12; 0.112	

[a] Variable log-transformed.
* Significant at 0.05 level; *T*-value >1.96.

tasks they received. Finally, no impact of type of license on time for task completion was detected. Thus, we fail to reject P5.2–P5.4, but reject 5.5. Type of user (end-users), which "require extensive and costly usability testing" (Johnson, 2002, p. 656), impacts activeness and effectiveness negatively. Likelihood of task completion is positively influenced by type of user (advanced end-users). And time for task completion is negatively affected by type of user (others) and positively by developers. Thus, we fail to reject Propositions 6.2–6.5.

Application domain affects activeness and effectiveness similarly (positively by education, office, and sociology; and negatively by games, multimedia, and system). However, application domain (desktop) affects activeness negatively, but does not affect effectiveness at all. Application domain (communications, sociology, system, and text-editor) affects likelihood of task completion negatively; whereas Internet do so positively. Finally, the domains religion, sociology, other and terminals tend to take longer to complete tasks. Consequently, we fail to reject P7.2–P7.5. Stage of development (beta and production) influences activeness positively, and beta, production and mature influence effectiveness positively. Moreover, the planning stage influences likelihood of task completion negatively, whereas pre-alpha, production, and mature do so positively. Finally, software in pre-alpha and alpha tend to close tasks faster, and those in production and mature tend to do so slower. All that being consistent with the model proposed, we fail to reject P8.2–P8.5.

Altogether, the results of F2's and F3's equations (activeness and effectiveness) have a very similar pattern when it comes to the significant variables. Among the most interesting results, we found that projects licensed under GPL (strong-copyleft), the most common and restrictive license, tend to be less active as well as less effective than projects that do not adopt GPL. This finding is consistent with previous studies that pointed out that GPL restrictions are seen negatively, decreasing people's intention to contribute (Comino et al., 2007; Fershtman and Gandal, 2007; Lerner and Tirole, 2005), and provides a counter-argument to those who suggested that the fear of open source software being "hijacked" into proprietary applications, maximized by non-restrictive licenses, would drive the community away from contributing (Sauer, 2007; Colazo and Fang, 2009).

A few other comments on the structural model results are worth making. First, software targeted at end-users affects activeness and effectiveness negatively (attractiveness positively), but applications of domains such as education, sociology and office, which are supposedly aimed at end-users too, tend to score higher on activeness and effectiveness (lower on attractiveness). This finding can only be sorted out with a specific study of the interactions between these categories, which can freely vary, the team compositions of these projects, and their user-interfaces. But one way to interpret it is with the logic that these projects have a significant learning curve and thus are not targeted at end-users as we have assumed (e.g., LaTeX). As a matter of fact, 65% of the sociology projects in 2008 were aimed at developers, and only 12% were aimed at end-users solely. The specific purpose of these projects and their characteristics would have to be understood in depth to be sure. Additionally, as the number of projects in sociology, for example, is rather small (17 in 2008), our sample could be biased towards another project condition that is prevalent in the statistical analysis (e.g., out of the 17, 12 are GPL and none is mature).

A second comment on the structural model is that life-span is not a significant influencer of activeness and effectiveness, indicating that the number of inputs and outputs do not increase simply because projects are available for a longer period of time. Likely, the community does not take "seniority" into account to decide whether to contribute to the project, but mainly its attractiveness. Third, the results suggest that more attractive projects have smaller likelihood to complete their tasks, indicating that an overload might occur as more tasks are requested in more attractive projects. In a similar pattern, projects under the GPL license are more likely to complete their tasks, as these projects tend to be less active. However, the variance of likelihood of task completion explained by the model is so low (3%) that, from a practical point of view, its interpretation is limited.

Further, we found that higher attractiveness is associated with more time for task completion, suggesting another side-effect of a higher number of requests, reports and posts (activeness). Having more tasks to deal with and a larger community gathered around these tasks, projects tend to slow down their work-pace, creating a positive chain of influences from attractiveness to activeness to time for task completion. Additionally, higher time to complete tasks may be associated with the type of tasks that are being generated by the community. Projects with more contributors are likely to generate more complex and important tasks to deal with, requiring more time but also being more rewarding. Furthermore, stage of development has an interesting pattern of influence on time for task completion. Projects tend to work faster at pre-alpha and alpha and slower at production and mature. This decrease in activities that accompanies project matureness was predicted by Stewart et al. (2006), and fits well with the logic we discussed before that it is "easier" to contribute in the earlier stages of software development.

The structural model explained 45% of activeness variance in 2006, 47.6% in 2007, and 49% in 2008; 39.4% of effectiveness in 2006, 41.4% in 2007, and 41.6% in 2008; 2.8% of likelihood of task completion in 2006, 2.5% in 2007, and 3% in 2008; and 13.6% of time for task completion in 2006, 12% in 2007, and 11.2% in 2008 (see Table 3). In summary, we failed to reject 23 of 24 propositions, explaining a significant part of FOSP work activities and demonstrating how powerful an understanding of attractiveness can be to manage open software development activities.

5. Results of the mediational model

The final sample used in the mediational model was of 4328 open source projects. The Notre Dame 2008 sample was of 4507, but several projects became inactive from January, 2008 to December, 2009 and, therefore, were excluded. The

overlapping sample between the Notre Dame database and the SourceForge.net website, obtained via Web crawler, after applying our filter, was of 4328.

5.1. Principal component analysis: conjoint role of resources

We performed a principal component analysis (PCA) to extract factors from hits, downloads and members from the past (2008) and future (2009). This is a necessary step to perform the mediation analysis based on Preacher and Hayes (2008), which is made using one independent, and one dependent, variable, not a latent construct with three indicators. The parameter we adopted to retain factors from variables was the Kaiser criterion of Eigenvalues greater than one (Stevens, 1986). According to this criterion, both sets of three variables, 2008 and 2009, formed only one factor. The first component extracted from the set of the past had Eigenvalue of 1.97, explaining 65.8% of their variance. Similarly, the set of the future had Eigenvalue of 1.85 with 61.5% of variance explained. No other factor had an Eigenvalue greater than 1. Using the weights of the components extracted, we calculated one variable for each set in a multiple regression fashion.

5.2. The self-reinforcing effects

As it should be consistent with the structural model, in the mediational model (Fig. 2), past attractiveness, controlling for projects' set of conditions, significantly influences effectiveness, likelihood and time for task completion, which, as a set of mediators, affect future attractiveness significantly as well, not allowing the rejection of Proposition 9. The signs of the effects from past attractiveness to each construct related to project activities were also consistent with the structural model results. The total indirect, mediating, effect of past attractiveness through activities on future attractiveness was significant and was calculated by summing the product of their coefficients (i.e., past attractiveness on project activities times project activities on future attractiveness). The bootstrapping resamples provided the intervals to decide for significance. Noteworthy is that although the set of mediators significantly affects future attractiveness indirectly, the direct effect of likelihood of task completion was not found significant in the presence of effectiveness and time for task completion. That is, the interval calculated via bootstrapping for the coefficient of likelihood on future attractiveness includes zero.

To be able to retain Proposition 10 and not reject it, the total effect of past attractiveness on future attractiveness would have to be reduced when controlling for project work activities and set of conditions, but not so much as to bring it to zero. Being reduced to zero would mean that the effect of past attractiveness on future attractiveness is fully mediated by project activities and, therefore, would lead us to the decision of rejecting Proposition 10. The results, in contrast, show that the effect of past attractiveness on future attractiveness is only partially reduced with the inclusion of project activities in the model (from .87 to .8). This means that there is a lot more besides work activities that visitors, users and developers, conjointly, do or represent that affects project attractiveness. The influence of contributors is not restricted to the activities performed via the tools adopted by the project that are publicly available in their repositories.

It is interesting to observe in the mediational model that likelihood of task completion does not affect attractiveness as much as attractiveness influences it. These results suggest that the number of contributors reduces the likelihood of tasks being addressed, but that this operational behaviour does not influence how attractive a project is. Most likely, people are not aware of this before they engage in the project's activities or decide to use it. In addition to that, projects with more contributors take more time to complete tasks and that behaviour does affect attractiveness back in the same direction. A possible explanation for this unexpected finding is that projects that take longer to close their tasks are more careful in doing so or may be working on more substantive issues, which require complex coordination mechanisms and take more time but also reward the project more with future value.

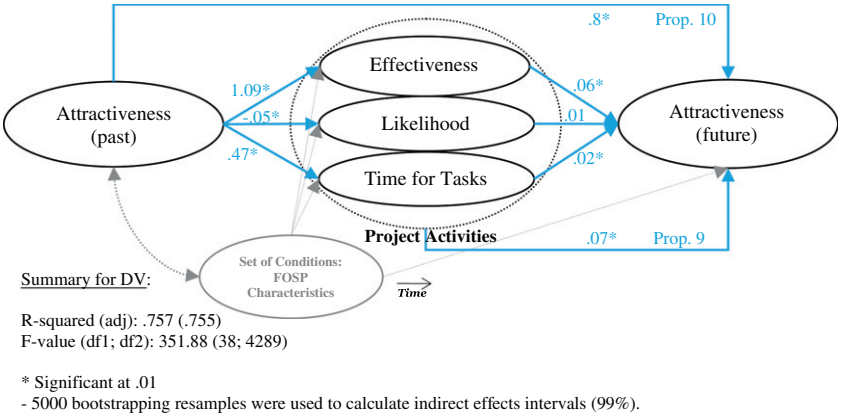

Fig. 2. Results of the mediational model.

These results are consistent with our discussion on preferential attachment and the independent and unique self-reinforcing effect of attractiveness, which happens on top of project work activities after controlling for its set of conditions (contextual factors). The mediational model explains about 75% of the variance of future attractiveness.

6. Discussion: Implications for theory and practice

While prior studies have only indirectly stated the importance of attractiveness to FOSP (Agerfalk and Fitzgerald, 2008; Fang and Neufeld, 2009; Sen et al., 2008), this paper seeks to define, further develop and model the attractiveness construct in relation to its causes, contextual factors and impacts, both theoretically and empirically. By arguing that groups of visitors, users and developers provide FOSP with varying and unique sets of development resources that together promote and improve the project towards the sustainability of recruiting new resources and generating more work activities and other indirect contributions, our findings are important to theory and practice in several ways.

Our model elaborates the observed behavioural aspects of preferential attachment specific to open source projects, providing an explanation for how the highly-skewed distribution of resources among them is generated and sustained. Most visitors, users and developers gravitate around only a few projects because of their tendency to select the "attractive" project to visit, use, join and contribute, which constitute the key actions on improving a project attractiveness. As this mechanism repeats, a reinforcing cycle is sustained, to the benefit of the already preferred. These selection processes are influenced by FOSP's set of conditions, which not only influences decisions of adopting (directly affecting attractiveness) and contributing (indirectly influencing attractiveness), but also defines projects' competition space for resources, limiting their achievable level of attractiveness. In a hypothetical world, where the population of projects and their set of conditions are stable, resources' stream of actions would be concentrated at predictable levels. However, new projects are created all the time, and old ones change their conditions constantly, disturbing the system of projects to influence resources' decisions and making the task of prediction complex. This dynamic and turbulent environment is what creates opportunities for the new to emerge, and threats for the successful to succumb.

As the results suggest, resources' perception of attractiveness is formed based on: certain contextual factors or characteristics of the project, that is, the identity of an object in comparison to the others; its visibility, which is conditioned by the amount of resources that project has and their indirect contributions; and the work activities performed by their resources towards software maintenance and improvement. Together, project conditions and its members' activities interact to form a dynamic and market-wide reputation.

The tendency to act towards "the same" projects set forth their momentum in a reinforcing loop that recursively determines an extreme concentration of resources (making the rich richer). Alternatively, projects can be excluded from resources' decision space for lacking attractiveness and thereby spiral down, making the poor poorer. This process of disturbance could be triggered by a change of sponsors that reconfigures the industry (e.g., OpenOffice in the Sun-Oracle deal), or by a license change (see Santos et al., 2011). This process that FOSP follow to accumulate resources, together with the limited availability of resources in the market and the high competition for them, generates a disparate ecosystem with the scale-free characteristic observed (Xu and Madey, 2004). Nevertheless, as FOSP are social products, there are ways in which project leaders can act to influence how attractive their projects are to the resources they are interested in attracting and motivating.

The findings suggest that new users, visitors and developers are likely to choose the more attractive project in comparison to the others. Yet, new projects can be better positioned, such as by choosing an appropriate license to increase their likelihood of being chosen. Project leaders should be mindful of how certain decisions affect visitors, users and developers perceptions similarly, easing their task of managing the community and software evolution.

New and undifferentiated projects stand a small chance to succeed in a populous and competitive environment, as the influence of project characteristics is relatively modest when comparing to the self-reinforcing force of attractiveness. Nevertheless, there are ways to overcome this "liability of newness". For example, one may reuse available open source code and create derivative projects (forking), or opensource a proprietary software with an established user-base and good reputation, carefully choosing which characteristics to give to the new project (e.g., avoiding being listed in the application domain "others" or choosing a license strategy that differentiates the project from its competitors).

Some of the project conditions affect attractiveness and work activities in opposite directions, such as the GPL license and the dual-licensing strategy. This creates opportunities for the project manager to operate strategically. For example, one may decide to have dual-licensing first, focusing on recruiting resources for the new project to build critical mass, and later on change to non-restrictive licenses, focusing on the generation of work activities, which would improve attractiveness indirectly through other means such as software maintenance. Additionally, project managers should invest higher amounts of their private resources in the initial stages of a software release, as "young" applications (up to alpha) tend not to attract resources from the community. This means that a higher effort on work activities and other indirect contributions such as running ads on key channels is required to overcome the challenge to emerge, until the thresholds of critical mass and "mature application" are reached.

On a theoretical side, our findings suggest that variables such as website hits, downloads and members should be treated as causes of each other or as final dependent variables on studies of FOSP *only* in restricted and well justified situations. We have shown that these variables are highly correlated not because they cause each other, but due to the existence of causes

which lead to their occurrence. One decides to visit a website based on reasons similar to those one considers to download and use, or even to become a member and contribute to open source software. The process may occur in sequence, from visiting to using to developing, but it is people's perception of software value (i.e., how attractive they perceive the project to be) that influences their moving from one step to another (von Krogh and Spaeth, 2007). It is the perception of the project at consideration that impacts people's behaviour. Moreover, the presence of visitors, users and developers alone, or in combination, cannot improve a project. These agents are sources of improvements, but their presence is not improvement and could only sustain a project attractiveness in satisfactory levels temporarily. Therefore, to treat them as independent variables (or mediators) is vital to accurately represent how user-driven innovations can be stimulated by sponsors and social planners in general (Arakji and Lang, 2007).

Our findings shed new light into a dilemma in the Information Systems literature, where one stream claims that the quantity of people increases diversity, which fosters problem-solving and innovation (Raymond, 1999; West and O'Mahony, 2005), whereas the other argues via empirical evidence that the relationships between the number of developers and software quality and project effectiveness and efficiency are not significant (Balijepally et al., 2009; Stewart and Gosain, 2006). Without claiming to close the debate, our results can conciliate these two streams by presenting effects and side-effects of increasing the number of contributors. More people tend to locate and fix more bugs, request and develop more features, creating an environment, where innovations are more likely to occur and higher-quality software generated. However, more people reduce the likelihood to solve an increasing number of bugs reported and features requested at the same time that more time becomes necessary to solve tasks, possibly due to coordination difficulties (Comino et al., 2007). Therefore, with positive and negative effects of an increasing number of contributors, both streams raise valid points and should not be seen as contradictory.

On a related note to the number of contributors in projects, our research provides a new perspective to the free-riding "problem", which is commonly reported as something that threatens the open source model (Baldwin and Clark, 2006). The traditional idea is that private agents are not expected to invest enough of their resources to produce a public good, they would rather free-ride. However, as we have shown, users and visitors, which would qualify as free-riders in the traditional sense, are actually contributing to the project as well, helping it build a critical mass and raising developers' intention to contribute. So, there is nothing open source project creators and managers should do about preventing free-riders, for besides contributing indirectly or unintentionally, they have chosen not to go to the competition and, thereby, improve project's attractiveness instead of competition's. There is no pressing need to worry about attracting free-riders, said to be reducers of the probability of success (Bessen, 2005), as FOSP can actually benefit from them in various ways. Johnson's prediction that "when more individuals are present, the incentive to free-ride is raised" (2002, p. 644), so that contributions become less likely to occur, has no support from our perspective.

Moreover, frequently, members of FOSP communities are developers and users of the software, which broadens their perspectives on software quality to contain technical, functional, and business domain dimensions. This creates an environment, where many are likely to contribute because one's contribution as a developer benefits oneself as a user. That is a recipe for success, giving these communities an advantage over software produced by an organization, where developers and users are independent entities, requiring extra effort to align needs and priorities. Accordingly, more available resources, of any kind, should indeed create an environment favourable to software quality and project success in open source. From this perspective, there is no need to manage what type of resources a project is attracting, as one type helps bring the others, and the various types of contributions come for similar reasons.

Having established the need for organizations involved in FOSP to improve their projects' attractiveness and visibility along with prior research (e.g., Agerfalk and Fitzgerald, 2008), this study is informative for the strategic software development practice. Our study can guide organizations on matters to be faced when opensourcing software, or deciding which one to sponsor, by providing insights on how attractive a software would be, and how to influence it (e.g., through the selection of an specific combination of licenses or spending more on advertising towards the beginning). Specifically, organizations may use the results strategically: (1) to identify among their software, according to application domain and type of user, the ones more likely to succeed if opensourced; (2) as a guide to design and position a project more effectively, managing its attractiveness to attain desired goals; (3) to help decide on when to release source code (stage of development), adding objectivity to the subjective advices previously discussed that software should be released in "later stages" (Johnson, 2002; Raymond, 1999); (4) to plan on what to expect from the community as the software evolves, managing better the evolution process by adjusting coordination methods and marketing efforts accordingly; and (5) and to judge which project is preferable and more appealing to developers, visitors and users, both to adopt and get involved, and thus strategically choose, where to place sponsorship resources.

Finally, as the open source model of user-driven innovation resembles the knowledge production in science and has been adopted in other fields (von Krogh and Spaeth, 2007), the research model proposed here can be adapted to help us understand other public projects of collective production as well, especially those that fit into the category of open innovation. One particular evident case outside the software industry is the production and improvement of knowledge that takes place in the public encyclopedia Wikipedia.org. By analogy, we can associate a page or article in the encyclopedia with an open source software project, and say that articles have readers (visitors), content users (e.g., those who cite), and writers (developers). As far as the distribution of these resources to articles, we expect it to be highly skewed as well, towards the popular culture and topics in fashion, for example (contextual factors). Accordingly, we can begin to construct a similar model, considering articles' set of conditions that influence resources behaviour (e.g., stage of development, language, type of content, etc.), gathering different types of contributions to articles such as orthographic corrections, addition of paragraphs,

usage by citing the article in academic papers, so on and so forth. Probably, this model of collective production would generate new insights on how to design articles that are more likely to attract an independent and voluntary community of contributors around them, sustaining knowledge production and dissemination by and to public. A similar theoretical exercise could be performed to generate models for blogs, social networks' profiles, open innovation problem solving, email list threads, etc. In conjunction, these various models would create a body of knowledge useful for organizations and individuals to operate strategically and so more effectively in their Internet-based strategic endeavours of communicating and collaborating with peers and customers.

7. Limitations

Research limitations can be divided into internal and external. Internal limitations are related to how the data was collected and the analysis performed. First, on capturing the effects of license, we could not classify all licenses available to FOSP according to their restrictiveness. We ignored licenses not classified by Lerner and Tirole (2005) and thus lost their effects. Second, although we collected data over time, it was analysed in the structural model using a cross-sectional approach. In doing so, we were not able to control for auto-correlations. However, this limitation is a cost of using a public secondary data in need of validation. To some extent, this limitation was addressed by our mediational model, which provided further support for the claims that FOSP characteristics and the number of contributors influence together how a project grows, changes, and gains momentum over time.

Amid the external limitations, a variety of potentially important variables were omitted from the analysis. There are other candidate explanations for attractiveness that were not included in our model. Namely, we identified: (a) members' technical knowledge, directly affecting the types of tasks generated and addressed; (b) level of community trust on the project and its sponsors, affecting people's willingness to contribute (Agerfalk and Fitzgerald, 2008); (c) existence of sponsored developers, who would keep a project active on a regular basis, having a formal commitment to address tasks; (d) the possibility of an open source project be included in official GNU/Linux distributions, influencing directly its accessibility; (e) the effects of the programming language adopted as the availability of developers depends on that (Stewart et al., 2006); and (f) the role of usability and source code metrics on project attractiveness (Rajanen and Iivari, 2010; Meirelles et al., 2010). This list of causes is exemplary and many other possible influencers exist, but given that empirical research is constrained by mundane restrictions, we believe that our model is useful for it communicates the message and provides ground for further verifications and refinements.

Additionally, we believe that the "self" effect of attractiveness is a proxy for many mediators, which could not be measured. For example, when users spend their money and time promoting the project, the actual chain of causality is from attractiveness, to finding a user, to receiving a contribution (money and time) and, finally, to attractiveness. The attractiveness to attractiveness proposition is simply a shortcut, useful for empirically verifying the theoretical argument and for prediction purposes. Future research should measure these unobserved mediators.

Moreover, there is the fact that a project may change its characteristics over time, which our empirical model assumed to be stable. However, we believe that in such case, the theoretical model we laid out would still be valid. When a project changes its conditions, for example, its type of license, we expect that the change would trigger an effect on the project attractiveness and activities in the terms we have already discussed. Previous empirical research supports this claim, showing that the decision to change license alters project attractiveness (Santos et al., 2011).

As a final limitation, we admit that "a volunteer-based community [...] may not behave strategically", as Jaisingh et al. (2008, p. 260) stated. In that case, our study would lose some of its explanatory power in exchange to strengthening its prescriptive nature. Nevertheless, with higher prescriptions, this study demonstrates how opportunities to operate strategically in the (open source) software market exist and may be used to maximize social welfare (Jaisingh et al., 2008). For the latter, governments may reduce software development costs by effectively opensourcing their applications and choosing the most attractive ones to adopt, and countries with less capability to develop software would have easier access to the knowledge embedded in the source code and thus be better equipped to provide services to their populations by relying on what others have made available on the Web.

8. Future work

The best way to address research limitations is with follow-up studies. A wide variety of studies can be derived from our results and conclusions, covering topics related to both content and method. On the content side, as the phenomenon of dual-licensing was just recently identified and discussed (Jiang and Sarkar, 2009; Watson et al., 2008), and, to the best of our knowledge, this is the first study to empirically assess the impacts of this choice, replications would be beneficial. Moreover, Agerfalk and Fitzgerald (2008) pointed out that the recruitment of developers from an open source project by sponsors could erode the "unknown" aspect of the project, affecting trust levels and innovation rates. Their proposition relates to the limitation of not statistically controlling for sponsored-contributors and reinforces the need to add it in future studies.

Additionally, we encourage the development of competing models of attractiveness to be compared with this one, which is exploratory in nature. For example, we have stated that project activities, as well as users and developers, are consequences of attractiveness and decided to group visitors, users and developers because they are the contributors. Thus,

a competing model with all of them as indicators of one latent construct should also be evaluated. However, we have performed preliminary analysis and found that the addition of the variables related to project activities generates a principal component analysis solution with two factors (1-Eigenvalue = 3.38, variance explained = 48%, 2-Eigenvalue = 1.08, variance = 15%). The complete solution has seven components and thus supports our original proposition that only the sources of contributions, and not their contributions, should be grouped together.

Similar to prior research in code reuse (Maillart et al., 2008), our paper addresses attractiveness from the preferential attachment perspective. But this does not preclude other plausible mechanisms including the 'like attracts like', especially during the earlier stage of project formation. It is likely that the like attracts like mechanism (as predicted in script theory and social status hierarchy) presents a critical resource at the initial stage, in that a small number of highly resourceful developers of similar status will work together initially (Kuk, 2006). Once the critical mass is reached, the rich-gets-richer effect takes over (Lee et al., 2006) as the emergent networks characterized by peripheral contribution will be less costly to maintain (Hu and Wang, 2009). Future research can seek to map out these longitudinal changes of what explains the preferable projects.

Furthermore, as previously shown, the concept of attractiveness can be adapted and used in other fields of research, being useful to any user-centric effort or collaborative work performed on the Web (e.g., see Comino et al., 2007; Wagner and Majchrzak, 2007). In addition, the relative strategic value of opensourcing software, when compared to the traditional outsourcing and insourcing approaches (Qu et al., 2010), should also be studied. Finally, the theoretical framework developed in this paper can be extended by incorporating results of previous research, such as the explanation of what determines license choices (Sen et al., 2008). In doing that, we can visualize a more complete model that explains what determines the license choices, which impacts user adoption and contributor recruitment, project activities, and, ultimately, software maintenance and improvement, all of that under the perspective of FOSP attractiveness.

On the method side, variables' effect-sizes were not calculated by us. Accordingly, we do not know how strongly a specific project activity variable influences project attractiveness, for example. Also, the results reported here could be further evaluated in an attempt to predict projects' future attractiveness at various distances in time (lags). Thus, without future research, we will not know for how long a state of attractiveness lasts or is capable of sustaining recruitment, participation and engagement.

9. Conclusions

Given the increasingly common model of developing software by dissolving organizational boundaries and decentralizing the work activities to interact with globally distributed workers and users, this paper started by identifying and describing the distributional characteristics of FOSP key-resources in order to build a theoretical explanation for their behaviour. That is, we relied on the existent literature that demonstrated that most users and developers of open source are concentrated in a few projects for reasons yet unidentified but that conform with the preferential attachment phenomenon. We proposed that FOSP attractiveness is the fundamental reason for this observed behaviour, and provided an explanation for how the majority of visitors, users and developers come to select only a few projects to adopt and contribute, ignoring the rest (Xu and Madey, 2004).

Additionally, we stated that the notion of contributor and contribution so far adopted in the FOSP literature has been mostly restricted to source code developers and, only rarely, users or visitors. In doing so, many types of contributions have been neglected along with their role in project improvement and software maintenance via project activities and other less direct, or even unintended, modes of contributing.

After developing the theoretical framework, this paper empirically analysed data on thousands of projects from different sources to unfold patterns in their internal activities, which are consistent with the framework developed, according to complementary and independent statistical techniques. In summary, the results inform us about FOSP on a variety of areas. In the model background, there is a pool of FOSP created as a result of opensourcing initiatives, sometimes utilized "as a marketing technique" (Jiang and Sarkar, 2009, p. 208); and there is a community interested in using, studying and contributing to these projects. Thus, our concerns were to explore what drives people to specific projects, or what types of projects are more attractive to people, understanding how that impacts project activities and, consequently, attractiveness, cyclically and continuously.

We found out that attractiveness may indeed be a strong driving force of FOSP dynamics, working as magnetic core that influences how several relevant variables are related to each other in a systemic, reinforcing, way. In a nutshell, the conclusion is that the influx of resources and contributions in FOSP depends on project attractiveness, which is a product of contributions of many kinds that come more frequently and with higher intensity depending on projects' set of conditions. Accordingly, the theme of highest value to organizations interested in releasing or sponsoring open source software as a strategic choice is how to set up and manage a project to influence its attractiveness, selecting, designing and coordinating it to that end.

Acknowledgments

This research was funded by FAPESP (www.fapesp.br, process: 2009/02046-2), and the Horizon Digital Economy Research Institute at the University of Nottingham (http://www.horizon.ac.uk/). The sponsors had no influence on the decision to publish or in the content of the paper.

References

Agerfalk, P.J., Fitzgerald, B., 2008. Outsourcing to an unknown workforce: exploring opensourcing as a global sourcing strategy. MIS Quarterly 32, 385–409.

Anderson, J.C., Gerbing, D.W., 1998. Structural equation modeling in practice: a review and recommended two-step approach. Psychological Bulletin 103, 411–423.

Arakji, R., Lang, K., 2007. Digital consumer networks and producer–consumer collaboration: innovation and product development in the video game industry. Journal of Management Information Systems 24 (2), 195–219.

Bagozzi, Richard P., Dholakia, Utpal M., 2006. Open source software user communities: a study of participation in Linux user groups. Management Science 52 (7), 1099–1115.

Baldwin, C.Y., Clark, K.B., 2006. The architecture of participation: does code architecture mitigate free riding in the open source development model? Management Science 52 (7), 1116–1127.

Balijepally, V., Mahapatra, R., Nerur, S., Price, K.H., 2009. Are two heads better than one for software development? The productivity paradox of pair programming. MIS Quarterly 33, 91–118.

Barabási, A.L., 2005. Network theory – the emergence of creative enterprise. Science 308, 639.

Barabási, A.L., Albert, R., 1999. Emergence of scaling in random networks. Science 286, 509–512.

Baron, R.M., Kenny, D.A., 1986. The moderator–mediator variable distinction in social psychological research: conceptual, strategic and statistical considerations. Journal of Personality and Social Psychology 51, 1173–1182.

Bentler, P.M., 1989. EQS Structural Equations Program Manual. BMDP Statistical Software Inc., Los Angeles, CA.

Bessen, J.E., 2005. Open Source Software: Free Provision Of Complex Public Goods. Technical Report. <http://ssrn.com/abstract=588763>.

Bevan, N., 2006. Practical issues in usability measurement. Interactions 13 (6), 42–43.

Byrne, B., 2006. Structural Equation Modeling With EQS: Basic Concepts, Applications, and Programming, Multivariate Applications Series. Lawrence Erlbaum Associates.

Chin, W., 1998. Issues and opinion on structural equation modeling. MIS Quarterly 22.

Clauset, A., Shalizi, C.R., Newman, M.E.J., 2009. Power-law distributions in empirical data. SIAM Review 51, 661–703.

Colazo, J., Fang, Y., 2009. The impact of license choice on open source software development activities. Journal of the American Society for Information Science and Technology 60 (5), 997–1011.

Comino, S., Manenti, F.M., Parisi, M.L., 2007. From planning to mature: on the success of open source projects. Research Policy 36 (10), 1575–1586.

Crowston, K., Howison, J., 2006. Hierarchy and centralization in free and open source software team communications. Knowledge, Technology, and Policy 18, 65–85.

Crowston, K., Scozzi, B., 2002. Open source software projects as virtual organizations: competency rallying for software development. IEE Proceedings Software Engineering 149, 3–17.

Crowston, K., Annabi, H., Howison, J., 2003. Defining open source software project success. In: 24th International Conference on Information Systems (ICIS), Seattle, WA.

Crowston, K., Annabi, H., Howison, J., Masango, C., 2004. Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development. WISER Workshop on Interdisciplinary Software Engineering Research, SIGSOFT, Newport Beach, CA.

Crowston, K., Annabi, H., Howison, J., Masango, C., 2005. Towards a portfolio of FLOSS project success measures. In: 26th International Conference on Software Engineering, Edinburgh, UK.

Dow, K.E., Jackson, C., Wong, J., Leitch, R.A., 2008. A comparison of structural equation modeling approaches: the case of user acceptance of information systems. Journal of Computer Information Systems 48, 106–114.

Fang, Y., Neufeld, D., 2009. Understanding sustained participation in open source software projects. Journal of Management Information Systems 24 (4), 9–50.

Fershtman, C., Gandal, N., 2007. Open source software: motivation and restrictive licensing. International Economics and Economic Policy 4, 209–225.

Fershtman, C., Gandal, N., 2011. Direct and indirect knowledge spillovers: the "social network" of open-source projects. The RAND Journal of Economics 42, 70–91.

Fitzgerald, B., 2006. The transformation of open source software. MIS Quarterly 30, 587–598.

Grewal, R., Lilien, G.L., Mallapragada, G., 2006. Location, location, location: how network embeddedness affects project success in open source systems. Management Science 52, 1043–1056.

Guimera, R., Uzzi, B., Spiro, J., Amaral, L.A., 2005. Team assembly mechanisms determine collaboration network structure and team performance. Science 308 (5722), 697–702.

Hair, J.F., Black, W.C., Babin, B.J., Anderson, R.E., Tatham, R.L., 2006. Multivariate Data Analysis. Pearson Education Inc., Upper Saddle River, NJ.

Herbsleb, J., Mockus, A., 2003. An empirical study of speed and communication in globally distributed software development. IEEE Transactions on Software Engineering 29, 481–494.

Hu, H., Wang, X., 2009. Disassortative mixing in online social networks. A letter. Journal Exploring the Frontiers of Physics 86.

Jaisingh, J., See-To, E., Tam, K., 2008. The impact of open source software on the strategic choices of firms developing proprietary software. Journal of Management Information Systems 25 (3), 241–275.

Jiang, Z., Sarkar, S., 2009. Speed matters: the role of free software offer in software diffusion. Journal of Management Information Systems 26 (3), 207–239.

Johnson, J.P., 2002. Open source software: private provision of a public good. Journal of Economics & Management Strategy 11, 637–662.

Judd, C.M., Kenny, D.A., 1981. Process analysis: estimating mediation in treatment evaluations. Evaluation Review 5, 602–619.

Kaplan, D., 2008. Structural Equation Modeling: Foundations and Extensions. Sage, London.

Keller, E.F., 2005. Revisiting "scale-free" networks. BioEssays 27, 1060–1068.

Kelloway, E.K., 1998. Using Lisrel for Structural Equation Modeling. International Educational and Professional Publisher, SAGE Publications, CA.

Kline, R.B., 1998. Principles and Practice of Structural Equation Modeling. The Guilford Press, New York, NY.

Koch, S., 2004. Profiling an open source project ecology and its programmers. Electronics Markets, Special Section: Open Source Software, 14.

Krishnamurthy, S., 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. First Monday, 7 (6). <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/960/881>.

Kuk, G., 2006. Strategic interaction and knowledge sharing in the KDE developer mailing list. Management Science 52, 1031–1042.

Lee, S.H., Kim, P.J., Jeong, H., 2006. Statistical properties of sampled networks. Physical Review Letters 73.

Lerner, J., Tirole, J., 2002. Some simple economics of open source. Journal of Industrial Economics 50 (2), 197–234.

Lerner, J., Tirole, J., 2005. The scope of open source licensing. Journal of Law, Economics and Organization 21, 20–56.

Long, J., 2006. Understanding the role of core developers in open source software development. Journal of Information, Information Technology, and Organizations 1.

Maillart, T., Sornette, D., Spaeth, S., von Krogh, G., 2008. Empirical tests of Zipf's law mechanism in open source Linux distribution. Physical Review Letters 101.

Maitland, S.B., Dixon, R.A., Hultsch, D.F., Hertzog, C., 2001. Well-being as a moving target: measurement equivalence of the Bradburn affect balance scale. Journal of Gerontology: Psychological Sciences 56 (2), 69–77.

Mardia, K., Kent, J., Bibby, J., 1980. Multivariate Analysis. Academic Press, NewYork.

Meirelles, P., Santos Jr., C., Terceiro, A., Miranda, J., Chavez, C., Kon, F., 2010. A study of the relationships between source code metrics and attractiveness in free software projects. In: Brazilian Symposium on Software Engineering (SBES), Salvador, Brazil.

Mockus, A., Fielding, R.T., Herbsleb, J., 2000. A case study of open source software development: the Apache server. In: Proceedings of the 22nd International Conference on Software Engineering.
Muffato, M., 2006. Open Source: A Multidisciplinary Approach. Imperial College Press, London, UK.
Mulaik, S.A., 2005. Parsimony/Occham's Razor. Encyclopedia of Statistics in Behavioral Science.
Newman, M.E.J., 2002. Assortative mixing in networks. Physical Review Letters 89, 34–234.
Nonnecke, B., Preece, J., 2003. Silent participants: getting to know lurkers better. In: Lueg, C., Fisher, D. (Eds.), From Usenet to CoWebs: Interacting with Social Information Spaces. Springer Verlag.
Nonneke, B., Preece., J., 2000. Lurker demographics: counting the silent. In: Proc. SIGCHI Conf. Human Factors Comput. Systems (ACM), New York, pp. 73–80.
O'Mahony, S., 2007. The governance of open source initiatives: what does it mean to be community managed? Journal of Management & Governance 11, 139–150.
Papadakis, E.N., Tsionas, E.G., 2010. Multivariate Pareto distributions: inference and financial applications. Communications in Statistics – Theory and Methods 39 (6), 1013–1025.
Peterson, R.A., 1994. A meta-analysis of Cronbach's coefficient alpha. Journal of Consumer Research 21, 381–391.
Preacher, K.J., Hayes, A.F., 2008. Asymptotic and resampling strategies for assessing and comparing indirect effects in multiple mediator models. Behavior Research Methods 40, 879–891.
Price, D.J.de S., 1976. A general theory of bibliometric and other cumulative advantage processes. Journal of the American Society for Information Science 27, 292–306.
Qu, W.G., Oh, W., Pinsonneault, A., 2010. The strategic value of IT in sourcing: an IT-enabled business process perspective. Journal of Strategic Information Systems 19 (2), 96–108.
Raja, U., Tretter, M., 2006. Investigating open source project success: a data mining approach to model formulation, validation and testing. In: Proceedings of SUGI 31, San Francisco, California.
Rajanen, M., Iivari, N., 2010. Traditional usability costs and benefits - fitting them into open source software development. In: 18th European Conference on Information Systems (ECIS), Pretoria, SA.
Raymond, E.S., 1999. The Cathedral and the Bazaar. O'Reilly, Sebastopol, CA.
Roberts, Jeffrey A., Hann, Il-Horn, Slaughter, Sandra A., 2006. Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the apache projects. Management Science 52, 984–999.
Rutner, P.S., Hardgrave, B.C., McKnight, D.H., 2008. Emotional dissonance and the information technology professional. MIS Quarterly 32, 635–652.
Santos Jr., C., Cavalca, M., Kon, F., Singer, J., Ritter, V., Regina, D., Tsujimoto, T., 2011. Intellectual Property Policy and Attractiveness: A Longitudinal Study of Free and Open Source Software Projects. ACM Computer Supported Cooperative Work (CSCW), Hangzhou, China.
Santos Jr., C., 2008. Understanding partnerships between corporations and the open source community: a research gap. IEEE Software 25.
Sauer, R.M., 2007. Why develop open-source software? The role of non-pecuniary benefits, monetary rewards, and open-source licence type. Oxford Review of Economic Policy 23, 605–619.
Sen, R., Subramaniam, C., Nelson, M., 2008. Determinants of the choice of open source software license. Journal of Management Information Systems 25 (3), 207–239.
Shaikh, M., Cornford, T., 2003. Version Management Tools: CVS to BK in the Linux Kernel. COSPA Knowledge Base. <http://pascal.case.unibz.it/retrieve/2770/shaikhcornford.pdf>.
Sharma, S., Sugumaran, V., Rajagopalan, B., 2002. A framework for creating hybrid-open source software communities. Information Systems Journal 12, 7–25.
Simon, H.A., 1955. On a class of skew distribution functions. Biometrika 42, 425–440.
Stevens, J., 1986. Applied Multivariate Statistics for the Social Sciences. Hillsdale, NJ.
Stewart, K., Gosain, S., 2006. The impact of ideology on effectiveness in open source software development teams. MIS Quarterly 30, 291–314.
Stewart, K., Ammeter, A., Maruping, L., 2006. Impact of license choice and organizational sponsorship on success in open source software development projects. Information Systems Research 17 (2), 136–144.
von Hippel, E., 2005. Democratizing Innovation. MIT Press, Boston, MA.
von Hippel, E., von Krogh, G., 2003. Open source software and the "Private-Collective" innovation model: issues for organization science. Organization Science 14.
von Krogh, G., 2002. The communal resource and information systems. Journal of Strategic Information Systems 11 (2), 85–107.
von Krogh, G., Spaeth, S., 2007. The open source software phenomenon: characteristics that promote research. Journal of Strategic Information Systems 16, 236–253.
von Krogh, G., von Hippel, E., 2006. The promise of research on open source software. Management Science 52, 975–983.
von Krogh, G., Spaeth, S., Lakhani, K.R., 2003. Community, joining, and specialization in open source software innovation: a case study. Research Policy 32, 1217–1241.
Wagner, C., Majchrzak, A., 2007. Enabling customer-centricity using Wikis and the Wiki way. Journal of Management Information Systems 23 (3), 17–43.
Watson, R.T., Boudreau, M.-C., York, P.T., Greiner, M.E., Wynn Jr., D., 2008. The business of open source. Communications of the ACM 51, 41–46.
West, J., O'Mahony, S., 2005. Contrasting community building in sponsored and community founded open source projects. In: Proceedings of the 38th Annual Hawai International Conference on System Sciences, Waikoloa, Hawaii.
Wiggins, A., Howison, J., Crowston, K., 2009. Heartbeat: measuring active user base and potential user interest in FLOSS projects. In: Proceedings of the Fifth International Conference on Open Source Systems (OSS), pp. 94–104.
Xu, J., Madey, G., 2004. Exploration of the open source software community. In: Proceedings of North American Association for Computational Social and Organizational Science (NAACSOS), Pittsburgh, PA, USA.
Ye, Y., Kishida, K., 2003. Toward an understanding of the motivation Open Source Software developers. In: Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, pp. 419–429.
Yuan, K.-H., 2005. Fit indices versus test statistics. Multivariate Behavioral Research 40, 115–148.