ATTRACTIVENESS OF FREE AND OPEN SOURCE SOFTWARE PROJECTS

Journal:	*18th European Conference on Information Systems*
Manuscript ID:	ECIS2010-0368
Submission Type:	Research Paper
Keyword:	Software project management, Online communities, OSS/FLOSS, Measurement models/methodologies/metrics
Note: The following files were submitted by the author for peer review, but cannot be converted to PDF. You must view these files (e.g. movies) online.	
18 Jan ECIS2010 (Attractiveness of Free and Open Source Projects)--camera-ready.pdf	

ATTRACTIVENESS OF FREE AND OPEN SOURCE PROJECTS

Santos Jr., Carlos Denner, University of São Paulo, Brazil, denner@ime.usp.br

Pearson, John, Southern Illinois University at Carbondale, USA, jpearson@cba.siu.edu

Kon, Fabio, University of São Paulo, Brazil, kon@ime.usp.br

Abstract

Organisations and individuals release source code on the Web to improve their software by attracting peers in the strategic move of "opensourcing" that has created thousands of open source projects (e.g., Eclipse-IBM, Thunderbird-Mozilla and Linux-Torvalds). Nevertheless, most of these projects fail to attract people and never become active. To minimize this problem, we developed a theoretical model around a crucial construct (attractiveness) to open source projects, proposing its causes (project characteristics), indicators (e.g., number of members) and consequences (levels of activeness, efficiency, likelihood of task completion, time for task completion and software quality). We tested this model empirically using 3 samples of over 4600 projects each in a multi-sample SEM analysis. The results confirm the central role that attractiveness plays to guarantee an active and efficient community of software development, shedding new light on whether more developers increase software quality by finding and fixing more bugs and providing upgrades. They also clarify the actual causal structure involving Web page visits, downloads and members, which can be easily mistaken. Moreover, the results can provide useful insights to strategists as we discuss the impacts of license restrictiveness, software development status, type of project and intended audience on attractiveness and its consequences.

Keywords: free and open source software, attractiveness, software engineering, software quality.

1 INTRODUCTION: THE STRATEGY OF OPENSOURCING

Open source (OS) software is one that has its source code available to anyone for inspection, modifications and utilization. Frequently, the toolset and documentation needed for performing these actions are provided on a Web site, enabling users to contribute. When development tools and source code are put together, an OS project is created. An OS project may come to the attention of people, or organisations, with interests that lead them to join the project. When that happens, we refer to a project as a community, which has developers, and contributors in general, directly interested in improving and promoting the project and its software.

OS communities have been cited to explain the success of software such as Linux, Apache, Sendmail, and Firefox. These applications have been adopted by many and widely studied by researchers trying to understand and describe how their success is possible. As knowledge of how OS software were and are developed became available and reached businesses, a trend of creating OS projects was born.

For an organization to use software, it must first decide on whether to develop it internally or externally (i.e., make-or-buy). The decision to make software internally implies the highest involvement over the development processes, with the organization managing the software construction throughout the entire process. Alternatively, when a decision to buy is made, the relationship with a supplier must be managed.

However, these two "pure" types of decisions are didactic as hybrid-types are possible, when one part of the development function is performed inside and another outside organizational boundaries. Recently, a hybrid-type of decision to develop software became common when organisations began "to opensource" their software to an unknown workforce (Agerfalk & Fitzgerald 2008).

The strategy of "opensourcing" is supported by what researchers have called the "open innovation model", where organisations benefit from an open communication channel with hobbyists, improving services and products based on inputs of external contributors (O'Mahony 2007). These benefits are supposed to come even if internal processes have to be revealed to accomplish it.

Attempts to open boundaries to contributors have been made by several organisations. For example, IBM decided to release Eclipse, a platform initially developed inside the organisation, as open source so that volunteers could contribute to it. Nowadays, Eclipse is an "ecosystem of major technology vendors", universities, and individuals that "extend, complement and support the […] platform."[1]

Also, the OS community has supported Mozilla's activities for years, participating not only in its browser production, but also in its email client, Thunderbird. Making it another "maker-and-buyer", Mozilla maintains a small product development team internally to work with thousands of volunteers from around the world, developing, designing and testing its applications. This configuration has enabled Mozilla to successfully compete against corporations such as Microsoft (Internet Explorer).

Finally, Limewire, an organization headquartered in New York City, "invite[s] all users interested in developing the Gnutella Network and its applications to join the LimeWire Open Source Project. Lime Wire LLC hopes to expedite Gnutella research and development by providing the core message passing and file sharing code so that one need not waste time re-writing it."[2] Limewire developed a peer-to-peer file sharing application that "went open source" in October 2001. Since then, Limewire has attempted to recruit volunteers by rewarding active members (contributors) with cash, prizes, internships, hirings, and by advertising these "rewards" on their Web site.

As shown, the strategy of "opensourcing" has been adopted by many organisations, but many others such as governments and non-governmental organisations may still benefit from it. Adopters of this strategy may reduce development time and time-to-market, improve software quality, reduce costs, and increase their hiring pool (Sharma et al. 2002). Nevertheless, not every OS project will become an

[1] http://www.eclipse.org/ - Accessed 03/25/08
[2] http://www.limewire.org/ - Accessed 02/10/2008

active community. Adopters of the strategy need to face the challenge of attracting independent contributors to support their software and assure the project success (Farhoomand 2007).

This paper presents a thesis that any OS project success depends on its level of attractiveness to potential contributors. A project attractiveness has been considered an essential characteristic since primordial OS software appearances. As Raymond (1999) stated, the more contributing members a project has, the easier it is for the software to be improved. Additionally, as more people join the project, the more diverse it becomes; and diversity fosters quality and innovation (O'Mahony 2007).

We present OS project *attractiveness* as a latent construct that "expresses" itself, empirically, in three variables: web page visits, downloads, and members. Furthermore, we propose that *attractiveness* influences directly projects' levels of *activeness*, *efficiency*, *likelihood* and *time of task completion*; which in their turn lead to "quality" and "success". Finally, our model states that four OS projects' characteristics (*license type*, *intended audience*, *type of project*, and *project's life-cycle stage*) are causes of their *attractiveness*, *activeness*, *efficiency*, *likelihood* and *time of task completion*.

The rest of this paper is organized as follows. First, a literature review of the concepts related to OS software projects and their dynamics appears. Next, the theoretical model is developed, stating the relationships between the constructs in the form of propositions. Then, a section on the methods used to test the propositions is presented. Finally, the statistical results of the analysis are presented, providing grounds for the discussion and concluding remarks section. The paper ends with a discussion on what we believe are the contributions to both researchers and practitioners – consumers of this study – shedding new light into previous research to open new possibilities for future studies and generating managerial knowledge to be applied in the marketplace.

2 OPEN SOURCE COMMUNITIES AND SOFTWARE

An OS community is a group of people geographically dispersed and connected through information and communication technologies developing OS software. These communities are composed of hobbyists and, increasingly, of paid "volunteers" (Fitzgerald 2006).

OS communities applications, and their source code, are almost always made available free of charge on a Web site, providing the necessary tools for the software to be improved by anyone willing to contribute to it. Some OS communities have become very well known, such as the ones responsible for the Apache Web server, the GNU-Linux operating system, and the OpenOffice suite.

After these communities and their managerial methods demonstrated their effectiveness producing high-quality software, corporate began to study and mimic their practices to deliver business value. Accordingly, many corporations have released their software to the OS community, opening up internal processes to engage in a strategy dependent on their project's attractiveness to be successful.

2.1 Attractiveness of OS Projects

The importance for OS projects to attract volunteers is established in the literature (Stewart & Gosain 2006). This necessity, combined with the enormous and increasing quantity of OS projects existent, promotes competition in an environment where some projects are chosen at the expense of others (West & O'Mahony 2005). A project's ability to attract people is defined mainly by its attractiveness.

However, although crucial, attractiveness by itself it is not capable of generating all desirable goals for an OS project. For a project to evolve positively, it must be productive, or receive inputs (e.g., bug reports) and generate outputs (e.g., bug fixes). Accordingly, we theorize that attractiveness influences projects inputs and outputs, enabling sponsors to succeed when creating an OS project.

2.2 Success of OS Projects

Previous research has defined OS projects success on a variety of bases though without the achievement of consensus (Long 2006). Among those measures, we were able to identify source code modularity (Shaikh & Cornford 2003), number of lines of code generated (Mockus et al. 2000), velocity of closing bugs (Stewart & Gosain 2006), and the number of downloads (Balijepally et al. 2009). Moreover, Raja and Tretter (2006) and Crowston and Scozzi (2002) viewed success as the ability of a project to advance through development phases (e.g., from alpha to beta to stable), and Crowston and Howison (2006) suggested the use of community size (i.e., number of members) as a representation of success. Finally, Stewart and Gosain (2006) adopted a construct labelled "efficiency" as dependent variable, which is composed of the attraction of inputs and the production of outcomes.

Although these measures are useful individually, we argue that success is achieved by their combination, or by the interaction between *attractiveness*, *activeness*, *efficiency*, *likelihood* and *time of task completion*. All these constructs are needed to achieve success and should be studied together.

3 RESEARCH MODEL

Before an OS project receives contributions, it must be attractive to volunteers, who first join the project and later provide inputs and develop outputs. Correspondingly, *attractiveness* comes before *activeness*, *efficiency*, *likelihood* and *time of task completion*. Additionally, we argue that these five constructs are antecedents of other constructs such as software and support quality.

3.1 Attractiveness

Attractiveness is defined as the project's ability to call the attention of a potential member and fulfil his or her interests, causing him or her to join the project (Stewart & Gosain 2006). We measure this construct based on three of its outcomes, or empirical expressions (indicators). Attractiveness can be captured through 1) the number of people that joined the project, 2) the number of visits the project's Web site received, and 3) the number of times the software was downloaded. Specifically, project A is said to be more attractive than project B during a certain period of time if A has more Web site hits (visits) than project B, everything else constant. Similarly, a highly attractive project is one whose software has been downloaded more times, *ceteris paribus*. Finally, the number of members a project has is influenced by its attractiveness. The more attractive a project is, the more members it has.

We expect these indicators of attractiveness to be significantly correlated with each other, but not highly, given that to visit a project's Web site does not imply downloading its software, and that a "join project" decision is even harder to occur.

3.2 Activeness

After a person became a member, his or her contributions are still "one-step" away. Activeness in the context of OS communities is defined as the project amount of input received (Raja & Tretter 2006). The more inputs a project receives, the more active it is. This should be the second construct of concern of a project sponsor, because this is what creates opportunities for software improvement. A bug cannot be fixed through the project if it is not reported. Therefore, if a project is to evolve, it must be through activeness. We argue that attractiveness is an influencer of activeness.

3.3 Efficiency

Efficiency is a project ability to complete a given task. A task is originated by an input received (activeness), and it might or might not be completed. When a task is completed, it enhances a project's

efficiency. In the OS projects context, bugs reported are closed, and features requested are developed. We logically derive that it is necessary that a member (attractiveness) report a bug (activeness) for it to be solved (efficiency). We argue that the more *efficient* an OS project, the more likely it is to succeed by having higher quality and providing better support to users. Furthermore, the lack of efficiency would condemn a project to its current state driving it away from users' changing demands.

3.4 Task Completion: Likelihood and Time

In production activities and service providing, to be able to solve problems within a reasonable amount of time is essential. In the OS literature, this has appeared as "[t]he more readily developers can recognize the needs and problems addressed by the project, the more successful the project" (Crowston & Scozzi 2002, p.10). Essentially, we argue that the more likely a project is to close its tasks, and the faster it does so, the more successful it tends to be.

3.5 Relationships between Variables

Our main thesis is that *attractiveness* influences all constructs of interest to creators of OS projects. We support the claim that diversity leads to quality through innovation, reducing costs and creating opportunities for the future (von Hippel & von Krogh 2003). In the context of OS, diversity can be translated as number of contributors. Accordingly, the main cause of an OS project success is its attractiveness, which influences activeness, efficiency, likelihood and time of task completion. We have developed four propositions to express these relationships formally (also see Figure 1).

Propositions: A project's attractiveness is a significant predictor of a project's **1)** *activeness*, **2)** *efficiency*, **3)** *likelihood of task completion*, and **4)** *time for task completion*.

3.6 Project Characteristics: The Causes

A variety of variables may influence attractiveness, activeness, efficiency, likelihood and time of task completion. There is a widely accepted paradigm in the OS software literature that explains voluntary contributions based on motivations, especially to receive compensations (Crowston & Scozzi 2002). We recognize this view, but also believe that the soft nature of this construct makes it too difficult to conduct any large empirical study that seeks to develop practical managerial knowledge. Alternatively, we argue that one's motivations drive him or her to a project with specific characteristics that fulfil one's motivations. Additionally, besides motivations, one's technical knowledge can also be captured by project characteristics, explaining their success (Crowston & Scozzi 2002). Thus, we attempt to identify which project characteristics fulfil more, or less, people's motivations.

3.6.1 License Type

Every OS software needs a license attached to it. This license regulates what can and cannot be done with the software and its source code, influencing its range of use and rules for modifications. For instance, no source code under the General Public License (GPL) can be used to produce proprietary software that is further redistributed. Other licenses such as the Mozilla Public are not as restrictive as GPL, permitting an interaction between OS and proprietary software (Fershtman & Gandal 2007).

A project's license influence on its activities has been documented. For example, Lerner and Tirole (2005) have examined how that is related to project's audience. Also, Fershtman and Gandal (2007) proposed that type of license is linked with the number of contributions received as members' willingness to contribute depends on the restrictions and allowances of a license. Thus, we have:

Propositions: A project's license type is a significant influencer of **5.1)** *attractiveness*, **5.2)** *activeness*, **5.3)** *efficiency*, **5.4)** *likelihood of task completion*, and **5.5)** *time for task completion*.

3.6.2 *Intended Audience*

Our motivation to insert intended audience in the model relates to the idea of niche in marketing. The bigger the niche, the higher one should expect sales to be, everything else constant. In the context of OS projects, audience defines the number of potential developers (Johnson 2002). Also, these different audiences attract specific members (e.g., system administrators), which define their expertise and likelihood to contribute. The influence of audience on OS projects has been discussed. For example, Crowston and Scozzi (2002) stated that projects that target developers tend to be more active than ones that target system administrators, which outperform ones targeted at end-users. Thus, we have:

Propositions: A projects' intended audience significantly influences its **6.1)** *attractiveness*, **6.2)** *activeness*, **6.3)** *efficiency*, **6.4)** *likelihood of task completion*, and **6.5)** *time for task completion*.

3.6.3 *Type of Project*

Just as OS projects have an audience, they also focus on a specific area; they are of a specific type. Projects might be related to genealogy, payroll, browsing, games, etc. (Crowston & Scozzi 2002). We argue that the project type influences their activities because we do not believe people choose their projects at random. Rather, they focus on projects they judge to be more "exciting" and related to their work (Johnson 2002). Arguably, available topics guide one to a specific area of interest, accounting for the attraction of particular groups with specific expertise and influencing project's dynamics. Thus:

Propositions: A project's type is a significant influencer of a project's **7.1)** *attractiveness*, **7.2)** *activeness*, **7.3)** *efficiency*, **7.4)** *likelihood of task completion*, and **7.5)** *time for task completion*.

3.6.4 *Life-cycle Stage (Development Status)*

Software can be classified based on its development status, which is often used as a strategy (e.g., beta releases). OS projects maintain their software status readily available to their members. Potentially, this status influences one's decision to join and contribute to a project as one evaluates whether that project is a "good" one to spend his time on. Projects at different stages show different levels of activeness and efficiency, as that affects members' motivations to release a new version (Raja & Tretter 2006; Stewart & Gosain 2006). Based on that, we incorporate development status in our model to test its degree and direction of influence to our endogenous variables. Formally, we have:

Propositions: A project's development status is a significant influencer of a project's **8.1)** *attractiveness*, **8.2)** *activeness*, **8.3)** *efficiency*, **8.4)** *likelihood of task completion*, and **8.5)** *time for task completion*.

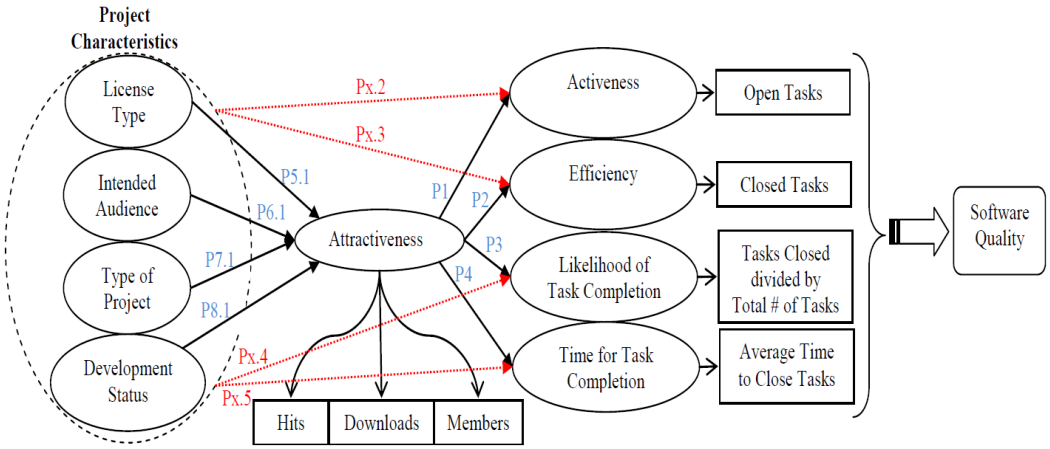

Figure 1. Research and Measurement Model[3].

[3] Propositions 5.2, 6.2, 7.2, and 8.2 are represented in the form of Px.2 in the figure; 5.3, 6.3, 7.3, and 8.3 are Px.3; 5.4, 6.4, 7.4, and 8.4 are Px.4; 5.5, 6.5, 7.5, and 8.5 are Px.5. The arrows pictured with straight lines are related to *attractiveness*.

4 METHODS

The population of OS projects is diverse in domain and audience (Raja & Tretter 2006), requiring large samples to test a theoretical model realistically. We chose to collect data from the largest repository of OS projects, SourceForge.net, which is available for research at the University of Notre Dame. We collected and filtered[4] samples from January of 2006, 2007 and 2008, analysing them in a repeated cross-sectional fashion. Variables were transformed to reduce skewness and kurtosis and for linearisation (Crowston & Scozzi 2002). Thus, *attractiveness* is the log of page views, downloads and members; *activeness* is the log-transformed sum of (1) bug reports, (2) support requests, (3) feature requests and (4) patches submitted; *efficiency* is the log-transformed sum of (1), (2), (3) and (4) that were solved; *likelihood of task completion* is *efficiency* divided by *activeness* in its inverse-sine-square-root; and *time for task completion* is the log-average time required to solve tasks (see Figure 1).

The constructs *license type*, *audience*, *type of project,* and *development status* were measured through a set of dummy variables. *License* required four dummies: no restriction, restriction of modification, of modification and use, and dual-licensing (Lerner & Tirole 2005). *Audience* required 5 dummies, *type of project* required 18, and *development status* 7 (totalling 34). Finally, one control variable, *life-span*, was included to capture the effects of project maturity (Fershtman & Gandal 2007).

To run the analysis, we applied Structural Equation Modelling (SEM) using the multisample capability and the Maximum Likelihood (ML) estimation criterion of EQS 6.1 (Byrne 2006; Chin 1998; Bentler 1989; Kline 1998). This statistical analysis is presented next in two blocks. The first evaluates the overall model-to-data fit of two different modelling strategies to choose one. Strategy 1 allowed EQS to estimate coefficients freely on each sample independently, and Strategy 2 forced EQS to find only *one* set of estimates for all 3 samples. The second block presents the model equations, providing the empirical tests for propositions 1 to 8.5 and, thus, ground to decide on whether to reject them or not.

5 RESULTS

Sourceforge.net had 149.542 projects in 2006, 179.867 in 2007, and 143.591 in 2008. After filtered, these numbers went down to 4769, 4611 and 4661, respectively. In the 2008 sample, 1327 projects had licenses that would not impose any restriction on the software use or modification, 2721 targeted end-users as an audience, 228 were listed as database, and 2009 projects had their software available in beta stage. The complete descriptive statistics and analytical tests for propositions evaluation (section 5.3) are available in a technical report that will be cited in the final, non-anonymous, version of this paper.

5.1 Attractiveness Reliability

We assessed attractiveness internal consistency across all samples through Cronbach's alpha. It scored 0.705 in 2006, 0.712 in 2007, and 0.714 in 2008. Usually, alpha values greater than 0.70 are considered acceptable (Balijepally et al. 2009; Stewart & Gosain 2006). Thus, *members, downloads* and *page views* are likely to be empirical expressions of a same construct (*attractiveness*).

5.2 Block 1 – Model Choice

Equation coefficients were calculated both (1) independently, sample-by-sample, and (2) forced to be equal across samples and compared. The difference in chi-square between the models is 206 (DF=358; p-value>0.9). Thus, the null hypothesis is not rejected, indicating that model (2), which is more restrictive and parsimonious, produces at least as good fit as model (1), being preferable.

[4] We filtered out projects with less than 2 members, no activity at all, and with invalid data values.

Fit indices check whether the pattern of covariances is consistent between the specified model and the sample data (Dow et al. 2008). A "good" fit is a necessary condition to analyse SEM models. Model (2) has RMSEA of 0.016, CFI of 0.982 and chi-square of 2736.91 (DF=592; p-value<0.01). Chi-square test is known for its sensitivity to sample size and number of parameters (Cheung et al. 2006). Accordingly, researchers accept a model as of "good" fit when at least one index suggests so (Cheung et al. 2006). Among fit indices, "RMSEA is relatively [the] most stable" and the preferable for theory testing (Yuan 2005; Cheung et al. 2006). Therefore, we concluded that the constrained model has an acceptable fit (CFI > 0.90; RMSEA < 0.05). These numbers are summarized in Table 1.

	Model with Equality Constraints			Model without Equality Constraints			Comparison
	2006	2007	2008	2006	2007	2008	
Attractiveness Cronbach's Alpha	0.705	0.712	0.714	0.705	0.712	0.714	
Chi-Square	2736.912 (592 Degrees of Freedom)			2530.101 (234 Degrees of Freedom)			206.811 (358 d.f.)
P-Value for Chi-Square	< 0.01			< 0.01			Same
Model Fit (CFI)	0.982			0.981			0.001
B-B Normed Fit Index	0.977			0.979			-0.002
Root Mean Square Residual (RMR)	0.015			0.014			0.001
RMSEA	0.016			0.026			-0.010
				Chi-Square Critical Value (0.05; 358 d.f.):			403.121
				Decision (given 206.811 < 403.121):			Favor Model with Constraints

Table 1. Model Comparison and Model-to-Data Fit Indices.

5.3 Block 2 – Testing Propositions

The model[5] developed here revolves around *attractiveness*. Having already dealt with *attractiveness* measurement, let us turn to its causes and consequences, which required the analysis of 5 equations. In regression terms, the first equation has *attractiveness* as dependent variable, explained by 34 dummy variables plus *project life-span*. *Life-span* is a positive and statistically significant predictor of *attractiveness*, which increases as projects grow older. Out of 4 dummies used to study *license type*, one (*dual_licensing*) was found to influence *attractiveness*. Registering software under licenses with different constraints is a trend in OS projects with commercial intentions (Santos Jr. 2008), affecting *attractiveness* positively. Thus, we fail to reject P5.1[6]. For *audience*, projects that target *end-users* and *developers* have higher *attractiveness*, and those aiming at *others* have lower (fail to reject 6.1).

Projects listed as *multimedia*, *printing*, *security* and *system* (*type of project*) have higher *attractiveness*, and those in *database*, *education*, *other*, *scientific* and *sociology* have lower (fail to reject 7.1). Specifically, projects should avoid having the projects of the *others* and *sociology* kinds as they hinder *attractiveness* the most. Preferred projects are of the *printing* and *security* types. D*evelopment status* significantly influences *attractiveness* in all seven possibilities (fail to reject 8.1). *Status*, in its initial phases (*planning*, *pre-alpha*, and *alpha*), affects *attractiveness* negatively, whereas advanced phases (*beta*, *production*, and *mature*) enhance *attractiveness*. *Inactive* scares people away. This suggests that releasing software in initial stages is ineffective and should be avoided. This equation explains 21.8% (2006), 17.4% (2007), and 15.8% (2008) of *attractiveness*' variance.

The other 4 equations focus on consequences of *attractiveness*; namely, *activeness*, *efficiency*, *likelihood* and *time of task completion. Attractiveness* is a significant predictor of all four variables and thus we fail to reject propositions 1 to 4. It positively pushes up *activeness* and *efficiency*, just as Raymond (1999) predicted. However, these effects have the side-effects of making a project less *likely to complete tasks* and slower at *completing* them.

[5] The model has each category (e.g., end-user) as one exogenous variable with error variance set to 0 (zero), *attractiveness* as a latent construct with three indicators, plus the control variable *project life-span* with one indicator and error variance set to 0. All covariances between exogenous factors were estimated. EQS code and covariance matrices are available on request.

[6] If at least one of a construct (e.g., license type) dummies (e.g., dual_licensing) is significant, it follows that the categorical construct is significant as well. A construct impact is non-existent only if none of its dummies is significant.

The impact of *license* (*both_restrictions*) is significant and negative on *activeness* and *efficiency*. Moreover, *both_restrictions* influences *likelihood of task completion* positively, suggesting that projects under GPL are more likely to complete tasks. Finally, no impact of *license* on *time for task completion* was detected. Thus, we fail to reject P5.2 to P5.4, but reject 5.5.

Targeting *end-users* impacts *activeness* and *efficiency* negatively because that requires extensive usability testing (Johnson 2002). *Likelihood* is positively influenced by *audience* (*advanced-end-users*). And *time* is negatively affected by *intended audience* (*others*) but positively by *developers*. Thus, we cannot reject propositions 6.2 to 6.5.

Type of Project affects *activeness* and *efficiency* similarly: positively by *education*, *office*, and *sociology*; and negatively by *games*, *multimedia*, and *system*. However, *desktop* affects negatively *activeness* but not at all *efficiency*. *Communications*, *sociology*, *system*, and *text-editor* affect *likelihood* negatively; whereas *education* and *internet* do so positively. Finally, projects listed as *other*, *religion*, *sociology* and *terminals* take longer *to complete tasks*. Thus, we fail to reject P7.2 to P7.5.

Life-cycle status *(beta*, *production*, and *mature*) influences *activeness* and *efficiency* positively. Moreover, the stage of *planning* influences *likelihood of task completion* positively, whereas *pre-alpha*, *production*, and *mature* do so negatively. Finally, software listed as *pre-alpha*, and *alpha* tend to close tasks faster, as opposed to those in *production* and *mature*. Thus, we fail to reject P8.2 to P8.5.

Among the most interesting results, we found that projects *licensed* under GPL, the most common license, tend to be less *active* and less *efficient* than projects that do not have GPL license. This result is consistent with previous studies that stated that GPL restrictions decrease people's intention to contribute to the project (Lerner & Tirole 2005), and provides a counter-argument to those who suggested that the fear of OS software being "hijacked" into proprietary applications, maximized by non-restrictive licenses, would keep people from contributing (Sauer 2007). Also, 1) although targeting *end-users* affects *activeness* and *efficiency* negatively, projects listed under *education*, *sociology,* and *office*, which are supposedly aimed at *end-users*, tend to score higher on *activeness* and *efficiency*; 2) projects listed as *inactive* have higher scores of *activeness*, but these bugs and features (tasks) reported tend to not be closed, as the effect of *inactive* on *efficiency* is not significant; 3) *life-span* is not a significant influencer of *activeness* and *efficiency*, indicating that the number of reports and requests do not increase as projects *mature* and suggesting that software quality may truly increase with time as less problems tend to be found; and 4) *attractiveness* decreases *likelihood to complete tasks*, indicating that an overload may occur as more tasks are requested in more *attractive* projects. In a similar pattern, projects under the GPL are more *likely to complete their tasks*, as these projects tend to be less *active*. Nevertheless, the variance of *likelihood of task completion* successfully explained by the model is so low that, from a practical point of view, the model interpretation is uninteresting.

Finally, we found that higher *attractiveness* is associated with more *time for task completion*, suggesting another side-effect of an increasing number of requests (*activeness*) generated by higher levels of *attractiveness*. Having more tasks to deal with and more members 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*. Also, projects tend to work faster at *pre-alpha* and *alpha* and slower at *production* and *mature*. Moreover, projects rush for task closure at *inactive*.

Statistically, our model explains 45% of *activeness* variance in 2006, 46.9% in 2007, and 47.6% in 2008. It explains 39.4% of *efficiency* in 2006, 40.6% in 2007, and 40.2% in 2008. Additionally, the *likelihood* explained variance is of 2.9%, 2.5% and 2.9%. Finally, the *time for task completion* explained variance is of 13.1% in 2006, 11.2% in 2007, and 10.4% in 2008.

6 IMPLICATIONS FOR THEORY AND PRACTICE

Previous studies have suggested the importance of *attractiveness* to OS projects, but this is the first one to directly model it as a multifaceted latent construct and investigate empirically its causes and

consequences. For that uniqueness, this research contributes to theory and practice in many ways. For theory, our results show that variables such as number of *page views*, *downloads* and *members* should not be treated as causes of each other or as *final* dependent variables. They are correlated with each other not because they cause each other, but due to the existence of a common cause to all of them, *attractiveness*. Moreover, these variables by themselves cannot improve software. They lead to improvements, but none of them *is* improvement. Therefore, to study what the consequences of these independent variables (or mediators) are is vital to understand how success is achieved in OS projects.

Furthermore, the results shed new light on a dilemma in the IS literature, where one stream claims that quantity of people increases diversity, facilitating problem-solving and innovation; and another, that points out empirical evidence suggesting that the relationships between the number of members and software quality and efficiency are not significant (Balijepally et al. 2009; Stewart & Gosain 2006). This contradiction disappears by noticing the effects and side-effects of more *attractiveness* (members). More people tend to generate more innovative activity like requesting and developing features or reporting and fixing bugs, but they also reduce the *likelihood to solve* this higher *number of tasks* and require more *time* to solve them. Thus, both streams have reason from this point of view.

OS projects must manage *attractiveness* to improve visibility. Thus, the results are informative for practice by glimpsing on how attractive a software, given its characteristics, would be if opened. Strategists can use the results 1) to identify among their software the ones more likely to succeed; 2) to design and position an OS project more effectively; 3) to decide on matters of license and of when (life-cycle) to release code; and 4) to know what to expect from the community as the project evolves.

7 LIMITATIONS

Research limitations may be divided in internal and external. External ones relate to what could have been included but was not; and the internal relate to how things done in the study could have been done differently. Among the internal limitations, we could not classify all *licenses* available to projects as we focused on the ones classified by Lerner and Tirole (2005); and, we analysed our data cross-sectionally, not longitudinally. In doing so, we could not control for constructs change over time or autocorrelations.

Amid the external limitations, a variety of potentially important variables were omitted from the analysis. To name a few: a) members' technical knowledge; b) level of trust among members, affecting their likelihood to share information; and c) existence of sponsored members ("employees").

8 FUTURE RESEARCH

The best way to address the limitations we identified is with follow-up studies, covering topics related to both content and method. On the content side, a more complete study of the influence of *licenses* is encouraged, where one would classify all *licenses* available to OS projects. Also, as the phenomenon of *dual-licensing* is new and this is the first study to empirically assess its impacts, replications are beneficial. Moreover, studies aimed at understanding what exactly the categories we utilized (e.g., *end-users*) mean to members are encouraged. This research may clarify counter-intuitive findings such as the negative effect of *end-user,* and the positive one of *office*, on *activeness* (it seems that *office* software aims at *end-users*). Also, Agerfalk and Fitzgerald (2008) pointed out that the recruitment of members by sponsors could erode the "unknown" aspect and affect trust levels and innovation rates, reinforcing the need to add it in future studies. Finally, 1) the causal direction between *attractiveness* and its indicators needs second look, as it has been suggested that users might adopt an OS software based on the number of *members* its project has (Baldwin & Clark 2006); and, 2) we posited that *activeness, efficiency*, *members* and *downloads* are all consequences of *attractiveness*. Thus, an obvious competing model is one that has all of them as indicators of *attractiveness*. Other competing models may also be developed toward more parsimony and theoretical understanding.

On the method side, to adopt a longitudinal approach would make the statistical tests more robust and interesting. A longitudinal study could test, for example, the relationship between *activeness* and *software quality*: where *activeness* peaks when software is at low-quality and decreases as software quality increases. Moreover, effect-sizes of variables were not calculated. Thus, we do not know how practically relevant a particular project trait is on its dynamics. Future studies may address this. All these possibilities remain open and by keeping the area active may lead to more knowledge available to the population in the form of software source code.

9 CONCLUSIONS

This paper empirically analysed OS projects to discover patterns in their dynamics. We saw that there is a pool of OS projects created from "opensourcing" initiatives, and that there is a population interested in contributing to these projects. But, what drives people to specific projects? Or, what types of projects are more attractive to people? We addressed these questions by focusing on *characteristics* capable of "positioning" projects better to the OS community, influencing their *attractiveness*. We also posited that attractiveness represents only partially OS projects goals, as they ultimately pursue not just adoption and use, but also the creation of an active community. Thus, we also sought to understand whether *attractiveness* leads to improvements (*activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*), which directly influence *software* and *support quality*.

Commonly, members of OS projects are developers *and* users of the software, which broadens their perspectives on quality to contain technical and functional issues (Santos & Gonçalves 2008). Moreover, these members are highly motivated to contribute for one's (developer) contribution also benefits one (user). That is a recipe for success, giving these communities an edge over software produced by an organization where developers and users are separate entities. Accordingly, more members should indeed lead to higher software quality. Johnson's (2002) prediction that more individuals increase the incentive to free-ride so that contributions become less likely has no support from our analysis. Thus, the theme of highest value to organisations interested in "opensourcing" is how to set up and run a project to maximize its attractiveness; and the best way to improve our knowledge-base on this theme is promoting the science of how to design and manage OS projects.

Acknowledgments

We thank three agencies, the Brazilians CAPES and FAPESP, and the American Fulbright, which together fully funded this project. Also, this research was benefited by the database access that the University of Notre Dame granted us, providing an opportunity to empirically test our research model. Finally, we are in great debt with Dr. Robert Ping for the valuable insights on statistically testing the ideas here developed.

References

Agerfalk, P.J., and Fitzgerald, B. 2008. "Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy," *MIS Quarterly* (32), pp 385-409.

Bentler, P.M. 2007. "On Tests and Indices for Evaluating Structural Models," *Personality and Individual Differences* (42:5), pp 825-829.

Byrne, B. 2006. *Structural Equation Modeling with Eqs: Basic Concepts, Applications, and Programming (Multivariate Applications) (Multivariate Applications Series)*. Lawrence Erlbaum Associates.

Cheung, M., Leung, K., and Au, K. 2006. "Evaluating Multilevel Models in Cross-Cultural Research," *Journal of Cross-Cultural Psychology 37, no. 5*.

Chin, W. W. 1998. Issues and Opinion on Structural Equation Modeling. MIS Quarterly, 22, 1.

Crowston, K., and Howison, J. 2006. "Hierarchy and Centralization in Free and Open Source Software Team Communications," *Knowledge, Technology & Policy* (18:4), Winter, pp 65-85.
Crowston, K., and Scozzi, B. 2002. "Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development," *IEEE Proceedings — Software Engineering* (149:1), pp 3-17.
Dow, K.E., Jackson, C., Wong, J., and Leitch, R.A. 2008. "A Comparison of Structural Equation Modeling Approaches: The Case of User Acceptance of Information Systems," *Journal of Computer Information Systems* (48:4), Summer2008, pp 106-114.
Farhoomand, A. 2007. "Opening up of the Software Industry: The Case of Sap," *Management of eBusiness, 2007. WCMeB 2007. Eighth World Congress on the*, p. 8.
Fershtman, C., and Gandal, N. 2007. "Open Source Software: Motivation and Restrictive Licensing," *International Economics and Economic Policy* (4:2), pp 209-225.
Fitzgerald, B. 2006. "The Transformation of Open Source Software," *MIS Quarterly* (30:3), pp 587-598.
Johnson, J.P. 2002. "Open Source Software: Private Provision of a Public Good," *Journal of Economics & Management Strategy* (11:4), pp 637-662.
Kline, R. B. 1998. Principles and Practice of Structural Equation Modeling. The Guilford Press, New York.
Lerner, J., and Tirole, J. 2005. "The Scope of Open Source Licensing," *Journal of Law, Economics and Organization* (21), pp 20–56.
Long, J. 2006. "Understanding the Role of Core Developers in Open Source Software Development," *Journal of Information, Information Technology, and Organizations* (1).
Mockus, A., Fielding, R.T., and Herbsleb, J. 2000. "A Case Study of Open Source Software Development: The Apache Server," in: *Proceedings of the 22nd International Conference on Software Engineering*.
O'Mahony, S. 2007. "The Governance of Open Source Initiatives: What Does It Mean to Be Community Managed?," *Journal of Management & Governance* (11), pp 139-150.
Raja, U., and Tretter, M. 2006. "Investigating Open Source Project Success: A Data Mining Approach to Model Formulation, Validation and Testing." Working Paper, Texas A&M University, College Station, Texas.
Raymond, E.S. 1999. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. Sebastopol, CA.: O'Reilly.
Santos Jr., C. 2008. "Understanding Partnerships between Corporations and the Open Source Community: A Research Gap," *IEEE Software* (25:6).
Santos Jr., C., and Gonçalves, M.A. 2008. "A Resource-Based Explanation for the Apache Consistent Dominance in the Web-Server Industry," *Associação Nacional dos Programas de Pós-Graduação em Administração (ANPAD)*, Rio de Janeiro, Brasil.
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* (23:4), Winter2007, pp 605-619.
Shaikh, M., and Cornford, T. 2003. "Version Management Tools: Cvs to Bk in the Linux Kernel," in: *Working Paper*.
Sharma, S., Sugumaran, V., and Rajagopalan, B. 2002. "A Framework for Creating Hybrid-Open Source Software Communities," *Information Systems Journal* (12:1), pp 7-25.
Stewart, K., and Gosain, S. 2006. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," *MIS Quarterly* (30:2), pp 291-314.
von Hippel, E. and von Krogh, G. 2003. Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. Organization Science, 14, 2 (March-April).
West, J., and 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.
Yuan, K.-H. 2005. "Fit Indices Versus Test Statistics," *Multivariate Behavioral Research* (40:1), 01, pp 115-148.