8-7-2009

OPEN SOURCE SOFTWARE PROJECTS' ATTRACTIVENESS, ACTIVENESS, AND EFFICIENCY AS A PATH TO SOFTWARE QUALITY: AN EMPIRICAL EVALUATION OF THEIR RELATIONSHIPS AND CAUSES

Carlos D. Santos Jr.
University of Sao Paulo, carlosdenner@gmail.com

I wish to thank CAPES and Fulbright for the financial support, SIUC for the structure provided, and Drs. John Pearson and Peter Mykytyn for the always-open door and stable willingness to help me. Also, I would like to thank Dr. Robert Ping for the valuable insights on statistically testing the ideas here developed and the University of Notre Dame for the prompt access granted to the database.

OPEN SOURCE SOFTWARE PROJECTS' ATTRACTIVENESS, ACTIVENESS, AND EFFICIENCY AS A PATH TO SOFTWARE QUALITY: AN EMPIRICAL EVALUATION OF THEIR RELATIONSHIPS AND CAUSES

by

Carlos Santos, Jr.

B.S., Universidade Estadual de Montes Claros (UNIMONTES), 2002.
M.S., Universidade Federal de Minas Gerais (UFMG), 2005.

A Dissertation
Submitted in Partial Fulfillment of the Requirements for the
Doctor of Philosophy in Management Information Systems.

Department of Management
in the College of Business and Administration at
Southern Illinois University Carbondale

August 2009

DISSERTATION APPROVAL

OPEN SOURCE SOFTWARE PROJECTS' ATTRACTIVENESS, ACTIVENESS, AND EFFICIENCY AS A PATH TO SOFTWARE QUALITY: AN EMPIRICAL EVALUATION OF THEIR RELATIONSHIPS AND CAUSES

by

Carlos Santos, Jr.

A Dissertation Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Ph. D.

in the field of Management Information Systems.

Approved by:

Dr. John Pearson, Chair

Dr. Peter Mykytyn

Dr. Marcus Odom

Dr. Scott McClurg

Dr. Greg White

College of Business and Administration at
Southern Illinois University Carbondale

July, 1st 2009.

AN ABSTRACT OF THE DISSERTATION OF

Carlos Santos, Jr., for the Doctor of Philosophy degree in Management Information Systems, presented on July, 1st 2009 at Southern Illinois University Carbondale.

TITLE: OPEN SOURCE SOFTWARE PROJECTS' ATTRACTIVENESS, ACTIVENESS, AND EFFICIENCY AS A PATH TO SOFTWARE QUALITY: AN EMPIRICAL EVALUATION OF THEIR RELATIONSHIPS AND CAUSES

MAJOR PROFESSOR: Dr. John Pearson

ABSTRACT: An organizational strategy to develop software has appeared in the market. Organizations release software source code open and hope to attract volunteers to improve their software, forming what we call an open source project. Examples of organizations that have used this strategy include IBM (Eclipse), SAP (Netweaver) and Mozilla (Thunderbird). Moreover, thousands of these projects have been created as a consequence of the growing amount of software source code released by individuals. This expressive phenomenon deserves attention for its sudden appearance, newness and possible usefulness to public and private organizations. To explain the dynamics of open source projects, this research theoretically identified and empirically analyzed a construct – *attractiveness* – found crucial to them due to its influence on how they are populated and operate, subsequently impacting the qualities of the software produced and of the support provided. Both *attractiveness*' causes and consequences were put under scrutiny, as well as its indicators. On the side of the consequences, it was theoretically proposed and empirically tested whether the *attractiveness* of these projects affects their levels of *activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion*, though not linearly, as *task complexity* could moderate the relationships between them. Also, it was argued at the theoretical level that *activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion* mediate the relationship between *attractiveness* and *software/support quality*.

On the side of *attractiveness*' causes, it was proposed and tested that five open software projects' *characteristics* (*license type*, *intended audience*, *type of project* and *project's life-cycle stage*) impact *attractiveness* directly. Additionally, these projects' *characteristics* were argued to influence projects' levels of *activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion* (and so an empirical evaluation of their associations was performed). The empirical tests of all these relationships between constructs were carried out using Structural Equation Modeling with Maximum Likelihood on three samples of over 4,600 projects each, collected from the largest repository of open source software, Sourceforge.net (a repeated cross-sectional approach). The results confirmed the importance of *attractiveness*, suggesting a direct influence on projects' dynamics, as opposed to the moderated-by-*task complexity* indirect paths first proposed. Furthermore, all four projects' *characteristics* studied were found to significantly influence projects' *attractiveness, activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion* (with the exception of *license type* and *time for task completion*). Besides providing a statistical test of these propositions, this study discovered the direction of the influence of each project *characteristic* on projects' *attractiveness*, *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion.* Lastly, conclusions, limitations, and future directions are discussed based on these findings.

DEDICATION

To those who find usefulness in this research.

ACKNOWLEDGMENTS

I wish to thank CAPES and Fulbright for the financial support, SIUC for the structure provided, and Drs. John Pearson and Peter Mykytyn for the always-open door and stable willingness to help me. Also, I would like to thank Dr. Robert Ping for the valuable insights on statistically testing the ideas here developed and the University of Notre Dame for the prompt access granted to the database.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1 – INTRODUCTION

The “Going Open Source” Software Strategy

For an organization to utilize computer software, it must first decide on whether to develop it internally or rely on external developer(s) and remove itself from the software development function. This is a “make-or-buy” type of decision. The decision to make software internally implies the highest involvement over the development processes, given that the organization must manage its personnel capabilities towards the construction of a software application throughout the entire process. Alternatively, when a decision to buy is made, the relationship with a supplier must be managed instead.

Of course, these two “pure” types of decisions are didactic as hybrid-types are possible, when part of the development function is performed inside and another outside the boundaries of an organization. Recently, this hybrid type of decision on how to deal with software development has become more common, as organizations began “to open source” their software.

Open source (OS) software can be characterized as one that has its source code, as well as the software application, available to anyone for inspection, alteration and utilization (von Hippel & von Krogh, 2003). Frequently, the necessary toolset and documentation (coding practices and manuals) for development, inspection, and utilization are provided on a website, enabling users to contribute. When development tools and software source code are put together, an OS project is created.

An OS project may come to the attention of people, or organizations, with interests that lead them to join the project. When that happens, it can be said that a project has a community of software developers and contributors in general (volunteers or not) directly interested in improving and promoting the project and its software.

OS communities of distributed contributors have been cited to explain the success of software like Linux, Apache, Sendmail, Firefox, and others. These software have been adopted by many and widely discussed by researchers in an attempting to understand and describe how their success is possible. As these ideas of how OS software were and are developed reached businesses, a corporate trend was born.

The organizational strategy of “going-open” is supported by what researchers have called the “open innovation model”, where organizations are assumed to benefit from an open communication channel with hobbyists, improving services and products based on their inputs (O'Mahony, 2007; von Hippel, 2005). The assumption is that organizations should keep their boundaries open to external contributors, even though they must reveal internal processes and possible sources of competitive advantage to accomplish that.

Attempts to open organizational boundaries to contributors have already been made by several organizations. To illustrate how these attempts can be identified, consider the four following examples. First, IBM has adopted Linux on some of its servers, and by that it adopts a software (a “buy” decision) from a third party (the Linux community of developers). Additionally, IBM has been

involved directly with Linux development activities, providing employees to develop this software. Accordingly, through this perspective, IBM "makes" *and* "buys" Linux. Therefore, in reality, a decision to buy, or to make, does not necessarily exclude the other as the expression "make-or-buy" implies. Furthermore, in another instance, IBM decided to release its Eclipse[1], a platform initially developed inside the organization, as open source so that volunteers could contribute to it. Thus, similarly, IBM made Eclipse, which now is developed by external contributors as well.

As a second example, Limewire "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 is an organization headquartered in New York City that developed a peer-to-peer file sharing application that "went open source" in October 2001. After that, Limewire has attempted to recruit volunteers by rewarding active members (contributors) with cash, prizes, internships, hirings, and, of course, by advertising these "rewards" through their website[3]. Given that Limewire has hired volunteers in the past, it also maintains non-volunteers in its activities, relying only partly on volunteers' contributions.

---

[1] "Eclipse is an open source community whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform." http://www.eclipse.org/ - Accessed 03/25/08

[2] http://www.limewire.org/ - Accessed 02/10/2008

[3] http://wiki.limewire.org/index.php?title=Bounties – Accessed 02/10/2008

Consequently, one could study Limewire as any other organization. It has expenses, such as payroll, and thus needs revenues to keep up with these expenses. To deal with that, Limewire sells Limewire PRO[4] to generate revenue. Therefore, framed in this context, one might say that Limewire is another example of an adopter of the special "make-and-buy" strategy based on open source.

Our third example is the producer of the Firefox Web browser, the Mozilla foundation. The OS community has supported Mozilla activities, participating not only in its web browser production, but also in its email client, Thunderbird, for years. Making it another "maker-and-buyer", Mozilla maintains "a small product development team to work with contributors from around the world on Thunderbird software."[5] Mozilla has thousands of unpaid (volunteers) and paid contributors developing, designing and testing its projects' source codes and usability. This configuration has enabled Mozilla to be successful in competing against corporations such as Microsoft with its Internet Explorer. Nevertheless, Mozilla claims to not have business concerns such as profit or stock price as organizational goals, but simply to promote openness and opportunity on the Internet[6].

Fourth and lastly, SAP, through its Netweaver platform, is another adopter of the hybrid-form "make-and-buy". SAP explicitly assumes that customers "may

---

[4] The PRO version of Limewire provides "optimized search results, turbo-charged downloads, and FREE tech support", among other things. http://www.limewire.com/download/pro.php - Accessed 03/25/08

[5] http://open.itworld.com/4915/mozilla-thunderbird-email-080219/page_1.html - Accessed 02/20/08

[6] http://www.mozilla.org/about/

have unique or emerging innovation needs that aren't met by an existing solution."[7] To fill this gap, SAP provides a diversity of environments referred to as SAP communities of innovation where members exchange information about SAP's products and help develop technical solutions. The SAP developer network, one of those communities of innovation, has over one million members[8]. To SAP, these members' contributions result in a code gallery[9] with a large variety of software code, projects, and instructions to be used at SAP's own discretion; possibly benefiting SAP users too, as they test and find them useful. To attract new and reward existing contributors, SAP provides, as a form of recognition, a public rank of top contributors[10]. That rank is based on the number and type of tasks a member was able to accomplish. SAP assigns different points to different types of contributions, according to other members' input.

Although SAP's software has not become an OS, strictly speaking, on its release (because the software source code is available only to customers), SAP makes sure to state that "open source and SAP are not a contradiction"[11], attempting to keep itself attractive to open source advocators' contributions. That is because like any adopter of the open source hybrid-type of decision, SAP needs to face the challenge of attracting a "global army of independent developers" to support and develop its software (Farhoomand, 2007, p.9); and

---

[7] http://www.sap.com/ecosystem/index.epx?source=gawucesys01&kw=sap+netweaver&KW_ID=p72863042 – Accessed 02/12/2008

[8] http://www.sap.com/ecosystem/communities/sdn/index.epx - Accessed 02/12/2008

[9] https://www.sdn.sap.com/irj/sdn/wiki?path=/display/Snippets/Home - Accessed 02/12/2008

[10] https://www.sdn.sap.com/irj/sdn/topcontributors - Accessed 02/12/2008

[11] https://www.sdn.sap.com/irj/sdn/opensource-integration - Accessed 02/12/2008

the OS community is a great source of contributors who are capable of assuring the success of a software project as an organizational strategy.

As shown, this software development strategy has been an adopted by many organizations, but it seems that it is also possible for many others such as governments and non-governmental organizations. This opportunity has already been identified as a strategy to become competitive "by outsourcing parts of the development to a passionate open source crowd."[12] Arguably, organizations that adopt this strategy, if successful, would reduce software development time and time-to-market, improve software quality, reduce costs, and increase the adopter's hiring pool of developers (Sharma, Sugumaran, & Rajagopalan, 2002).

Public administrations such as the governments of Brazil, Denmark, China, Japan, South Korea and South Africa are known to prefer open source instead of proprietary software for acquisition cost reasons and so may become adopters of this hybrid-type of decision in two different forms (O'Mahony, 2007). These organizations might adopt a piece of software from an OS community and get involved in the development of the software (e.g., IBM and Linux), or they may release software developed internally in an attempt to receive external help (e.g., IBM and Eclipse). In either of these cases, the organization may benefit from studies of OS projects' dynamics to explain their likelihood of improving their software, and their organizations through contributors' inputs. This dissertation is an effort in that direction.

---

[12] http://conferences.oreillynet.com/cs/railseurope2007/view/e_sess/13374 - Accessed 01/31/2008

This dissertation presents a thesis that any open project's success depends on its level of *attractiveness* to potential contributors, expressed primarily by the project's number of members. A project's *attractiveness* has been considered an essential characteristic since the earliest OS software appearances. Raymond (1999, p. 27), in a document considered to have provided the foundations for the open source movement, stated that "given enough eyeballs, all bugs are shallow". In other words, the more contributing members a project has, the easier it is for the project to be improved. Additionally, as more people join the project, the more diverse it becomes; and "pluralism can foster quality" and innovation (O'Mahony, 2007, p. 146).

We propose that a project's *attractiveness* influences a project's levels of *activeness*, *efficiency*, *likelihood of task completion*, and *average time to complete tasks*. Additionally, we posit that these relationships are not linear because a project's software *complexity* moderates them. To connect *attractiveness* with "quality" and "success", it is proposed that *activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion* mediate the relationship between *attractiveness* and "software quality" or "project success". Finally, our model proposes that four open software projects' characteristics (*license type*, *intended audience*, *type of project*, and *project's life-cycle stage*) influence *attractiveness*, *activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion*.

The rest of this dissertation presents first a literature review of the concepts related to OS software projects and their dynamics. Next, two

theoretical models are developed in chapter 2. One that contains only the endogenous[13] constructs of the model, and another that contains both endogenous and exogenous[14] constructs, which was empirically tested.

In chapter 3, we present the "methods" to test our developed model. This section proposes a repeated cross-sectional approach using Structural Equation Modeling (SEM) to accomplish the necessary statistical tests. Data from Sourceforge.net, the world's largest repository of OS projects, populated our samples. Details on how the data will be collected are also presented.

In chapter 4, the statistical results of the analysis are presented, providing grounds for the discussion and concluding remarks section, which appears in chapter 5. The dissertation ends with a discussion of what we believe are our contributions to both researchers and practitioners, shedding new light into previous research to open new possibilities for future studies and generating managerial knowledge to be applied in the marketplace.

---

[13] In Structural Equation Terminology, endogenous variables are ones that are caused by other variables in the model.

[14] Exogenous variables are ones that are *not* caused by other variables in the model. They influence other variables, but are not influenced by any.

CHAPTER 2 – LITERATURE REVIEW AND MODEL DEVELOPMENT

This section is organized as follows. First, a brief presentation of open source software and their communities is made. Second, attractiveness is discussed in the context of OS projects, and an argument of how attractiveness is expected to influence OS project outcomes, leading to success or failure, is developed. Third, a discussion of what success has been considered for (open source) software development projects is presented, closing the literature review and opening the model development section.

In developing our theoretical model, the endogenous constructs of the theory with their respective relationships are presented. Next, the exogenous constructs of the model and their respective relationships with the endogenous constructs are presented and added to the first model, generating the complete theoretical model.

Open Source Communities and Software

Many services available on the Internet rely on open source software and practices, but because these software and practices work in the background of those services, they may go unnoticed by many users. Google, Amazon, Ebay, and Wikipedia are a few examples of such services[15]. Furthermore, over fifty

---

[15] The Economist - Edition of March 16th, 2006
http://www.economist.com/displaystory.cfm?story_id=E1_VGNQJQQ

percent of the websites on the Internet use Apache[16], a web server software developed by volunteers and a variety of organizations.

An open source community is a group of developers geographically dispersed and connected together through information and communication technologies based on the Internet to develop software (Herbsleb & Mockus, 2003). Open source communities are composed of “hobbyists”, but the number of paid “volunteers” developing open source software is increasing due to the recent involvement of corporations in the movement (Fitzgerald, 2006).

These communities’ software application, and their source code, are almost always made available free of charge on a website, which provides the necessary conditions for the software to be improved by anyone willing to join the project and contribute to it. Some open source software have become very well known, such as the web server Apache, the operational system Linux, the office automation package OpenOffice.org, and the web browser Firefox.

After these projects and their adopted managerial methods demonstrated their feasibility, producing high-quality software, corporations’ attention turned to studying and mimicking “open source practices” as a way to deliver business value. Accordingly, many corporations have released their software to the open source community, opening up their internal processes and engaging in a strategy dependent on their OS project’s *attractiveness* to be successful.

---

[16] http://news.netcraft.com/ - Accessed on April, 17th, 2008.

Open Source Project's Attractiveness and Its Importance

Open source software has been around for over fifteen years, and its impact on the software market has increased along with the increasing investments of corporations such as IBM and Sun in their projects (Fitzgerald, 2006).

The importance for OS projects to attract volunteers has been established in the literature (Koch, 2004; Krishnamurthy, 2002; Stewart & Gosain, 2006; von Krogh, Spaeth, & Lakhani, 2003). This necessity combined with the enormous and increasing quantity of existent OS projects stimulate competition (West & O'Mahony, 2005). This increasing competition for volunteers creates an environment where some projects, by being more attractive than others, become more likely to be chosen by volunteers. A project's ability to attract volunteers is labeled as its *attractiveness*.

Although *attractiveness* is a necessary condition, by itself it is not capable of generating all desirable outcomes for an OS software development project. Assuming that the goal of a project is its improvement, to receive inputs (e.g., bug reports and feature requests) and generate outputs (e.g., source code) are also essential and necessary conditions for software improvement. Accordingly, the effects of the number of *attractiveness* (number of members) on productivity needs to be further explored (Crowston, Annabi, Howison, & Masango, 2005; Crowston & Scozzi, 2002; Koch, 2004). These abilities to receive inputs and generate outputs are labeled project *activeness* and *efficiency*, respectively.

The need to include these constructs, *activeness* and *efficiency*, in open source studies has been previously discussed in the literature and appears, together with project *attractiveness*, to represent a more complete picture of what a project's sponsor would be looking for when launching an OS project (Stewart & Gosain, 2006). Obviously, in the competitive arena for volunteers' attraction and time devotion, well-established projects such as Linux, Apache, PHP and Mozilla are ahead. These projects are very well known in the marketplace and therefore provide a better opportunity for developers to have their self-interests fulfilled. Also, one might expect developers with higher skills to go after those well-established projects as less-known projects would not add to their marketability.

This scenario leaves smaller and less known OS software projects, which represent the majority of these types of projects, at a disadvantage, justifying our decision to focus on them in our study. Accordingly, this research will exclude the well-known "successful" cases from its analysis as they may be considered outliers. Instead, we will focus on OS projects that do not have a significant reputation in the marketplace to function as a recruiter of volunteers; providing a picture of the cases of interest for this research, where "unknown" organizations (will) attempt to go "open source". Thousands of these projects are found on online repositories such as SourceForge.org.

To study these OS projects, first we need to understand what success means in their context so that we can imagine how it can be achieved.

Open Source Software Development Success

Researchers have pointed out that only a few OS projects such as Linux, Apache, MySQL and PHP have been successful, whereas the majority of the OS projects have failed (Long, 2006; Thomas & Hunt, 2004). However, the basis for reaching that conclusion is frequently not explicitly stated. What makes an OS project (un)successful? For example, why is Azureus, a project that develops a peer-to-peer client, has 26 members, been downloaded over 100 million times[17], been translated in over 25 languages, and has been developed for almost 4 years, considered not successful? Moreover, can an unsuccessful project become successful or vice-versa? Without a bounded definition of success, one is not able to defend an answer to these questions on objective grounds.

Previous research has defined success on a variety of bases though without the achievement of consensus (Crowston et al., 2005; Long, 2006). Among those measures, we were able to identify source code modularity (Shaikh & Cornford, 2003), number of lines of code generated (Mockus, Fielding, & Herbsleb, 2000), velocity of closing bugs (Stewart & Gosain, 2006), and the number of downloads (Crowston, Annabi, Howison, & Masango, 2004; Krishnamurthy, 2002; Stewart & Gosain, 2006). Moreover, Raja and Tretter (2006) and Crowston and Scozzi (2002) see success as the ability of a project to advance through development phases (e.g., from alpha to beta to stable), and

[17] Information collected from the Sourceforge.net website on May, 20th, 2007.

Koch (2004) and Crowston and Howison (2006) suggest the use of community size (i.e., number of members) as a representation of success.

From a different perspective, Weiss and Poland (2006, p.1) defined a project's popularity level "as being proportional to the number of Web pages that mention this project somehow." They hypothesized that a measure of popularity should correlate with the number of reported bugs and features requested, meaning success. However, this predicted relationship was not found significant, suggesting that either their measures of popularity are not good indicators of success, or that there is not, in fact, a correlation between them.

Much closer to our goals, Stewart and Gosain (2006) adopted a construct labeled efficiency as a dependent variable. They divided this variable into two sub-constructs. One is composed of the attraction of inputs from the development community, and the other represents the observable outcomes such as fixing bugs and adding features to the software. Similarly, Herbsleb and Mockus (2003) adopted a dependent variable that measured speed of task accomplishment, touching on items such as *activeness* and *efficiency* indirectly. Their main finding was that distributed software development activities take longer to complete than non-distributed ones due to more people being involved in the decision-processes.

Although all those mentioned above can be useful measures individually, we argue that success, regardless of the chosen definition, for open software projects can be better represented through a combination of them. Accordingly, we propose that the levels of *attractiveness*, *activeness*, *efficiency*, *likelihood of*

task completion, and *time for task completion* can explain success better. Noteworthy is that we do not intend to study success itself, but constructs we argue to be necessary to its achievement, independent of the project's owner/founder/sponsor goals. The composition of these five endogenous constructs is discussed next in the model development section.

Theoretical Model Development

Endogenous Variables

Before an OS project receives contributions of any kind, it has to be attractive to volunteers, who first decide to join the project and only later to provide inputs and/or develop outputs to its improvement. Accordingly, we argue that *attractiveness* comes before *activeness*, *efficiency*, *likelihood of task completion*, and *time for task completion*. Additionally, we argue that these five constructs should be highly desirable things for open software projects because they are antecedents of other constructs such as software/support quality and competitive advantage (i.e., differentiation for attraction of volunteers and inputs). Therefore, there is a need to understand their compositions and relationships before studying their links with other constructs, as either causes or consequences, if one wants to move towards a more complete understanding of these communities' dynamics. Next, each of the five constructs is presented separately, followed by a section on how they are connected together.

Attractiveness

Attractiveness is defined as the project's ability to call the attention of a potential member and, eventually, fulfill his or her self-interests, causing him or her to join the project (Crowston et al., 2005; Stewart & Gosain, 2006). Due to our inability to measure this construct directly, which would imply asking people

randomly on their willingness to join and participate on a series of projects, we measure this construct based on three of its outcomes, or empirical expressions.

Attractiveness of any project can be captured through the combination of the number of people that joined the project, the number of visits (hits) to the project's website, and the number of times the software or its code was downloaded, up to the point of measurement. Along these lines, a project's attractiveness is a dynamic construct that may change over time, possibly affecting its position on an attractiveness rank, if such was built.

Specifically, project A is said to be more attractive than project B during a certain period of time if A has more website hits (visits) than project B, everything else constant. Similarly, a highly attractive project is one that has been downloaded more times than a less attractive one, again, *ceteris paribus*. Finally, and the most important representation of attractiveness, the number of members a project has should be influenced by its attractiveness. The more attractive a project is, the more members it has, and will have. We expect these three different measures of attractiveness to be significantly correlated with each other, but not highly, given that to visit a project's website is one step before downloading its software, and that these do not necessarily lead to a "join project" decision. Additionally, there is still the fact that one might aim solely to use the software, ending their participation after visiting the website and downloading the software.

Another way to illustrate how the relationships among these empirical measures of attractiveness work is noticing that web sites that host OS projects

advertise ranks of most downloaded and active projects, increasing the visibility of the top ones[18] in an attempt to get them more visits, downloads and members.

An OS project's attractiveness is crucial for many reasons such as the thesis that it indirectly leads to higher level constructs such as quality because more "eyeballs" (contributors) find and fix more bugs, and request and develop more software features. Additionally, empirical studies suggest that an organizations' likelihood to fail decreases as their size (e.g., number of employees) increases (Baum & Oliver, 1991).

Activeness

After a person has become a member, his or her contributions to the project are still "one-step" away. Activeness in the context of open software communities is defined as the project's level of input generation such as feature requests and bug reports (Raja & Tretter, 2006; Stewart & Gosain, 2006). The more a project receives inputs of any kind by its members, the more active it is. This, we argue, should be the second construct of concern of a projects' sponsor/owner/founder, because this is what creates opportunities for software improvement by the project's members. A bug is not fixed through the project if it is not reported, and a feature is first requested and only then, hopefully, implemented. Therefore, if a project is to evolve, it must be through activeness. We argue that attractiveness is an antecedent, and an influencer, of activeness.

---

[18] http://sourceforge.net/softwaremap/

Efficiency

Efficiency is a project's ability to complete or accomplish 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 index of efficiency. In the OS projects context, bugs reported are closed, and features requested are developed (Crowston et al., 2005; Fershtman & Gandal, 2007; Raja & Tretter, 2006; Stewart & Gosain, 2006). Once one has a measure of efficiency of a set of projects at hand, one may rank-order these projects using this measure. Further, one could attempt to relate this variable with another. We logically derive that it is necessary that a member (attractiveness) report a bug (activeness) for it to be solved (efficiency). Therefore, in the very first "lap" of a project's development activities, efficiency is the last of the three to express itself.

It is the main thesis of this research that the more *efficient* an OS software project is, the more likely it is to succeed by having higher quality software and providing better support to the software users. Briefly, the success of the "going-open" strategy is dependent on efficiency, closing bug reported on time and providing new features to users as needed. Additionally, the lack of efficiency would condemn a software to its current state driving it away from the changing users' demands. That, nevertheless, is valid only in the case where there are no software development activities taking place out of OS project's boundaries. But even if there are software activities outside the project, that does not negatively affect our study because we are evaluating the OS project per se (i.e., whether

the project “works” or not, or whether it provides a useful environment for improvement of the software or not).

As a project resolves its raised issues (e.g., bugs) and develops its index of efficiency, two other pieces of useful information become available: the project’s overall likelihood of solving its tasks and the average amount of time spent to solve them. Next, we present these two as the last endogenous constructs.

Likelihood of Task Completion and Time for Task Completion

When one is interested in production activities or service providing of any kind, to be able to solve problems within a reasonable amount of time is an essential ability for success. Specifically, in the open source literature this has appeared as, for example, “[t]he more readily developers can recognize the needs and problems addressed by the project, the more successful the [OS software] project” (Crowston & Scozzi, 2002, p.10). Accordingly, we include these two as endogenous variables hoping to uncover patterns able to benefit adopters of the “going-open” strategy.

Essentially, we argue that the more likely a project is to get open tasks closed, and the faster it does so, the better off it would be. Therefore, the study of patterns and influencers of a project’s likelihood of task completion and time for task completion is justified from a managerial perspective. We will now present our five endogenous constructs connected together in the form of a model.

Relationships between Endogenous Variables

As stated previously, this paper's main thesis is that the level of *attractiveness* of a project should correlate with all main constructs of interest to managers/founders of these open source initiatives. We agree with the literature's assumption that diversity leads to quality through innovation, reducing costs and creating new opportunities in the future (O'Mahony, 2007; von Hippel & von Krogh, 2003). Diversity in the context of a project can be translated as its number of contributors, influencing its inputs and outputs. Accordingly, we have as the main cause of success of an open project its attractiveness. We argue that attractiveness is capable of influencing projects' activeness, efficiency, likelihood of task completion, and time for task completion. Therefore, the impact of attractiveness on these constructs is capable of driving a project to either success or failure. To express these relationships formally, we have developed four propositions, which are presented next and depicted in Figure 1.

Proposition 1: A project's attractiveness is a significant predictor of a project's *activeness*.

Proposition 2: A project's attractiveness is a significant predictor of a project's *efficiency*.

Proposition 3: A project's attractiveness is a significant predictor of a project's *overall likelihood of task completion*.

Proposition 4: A project's attractiveness is a significant predictor of a project's *average time for task completion*.

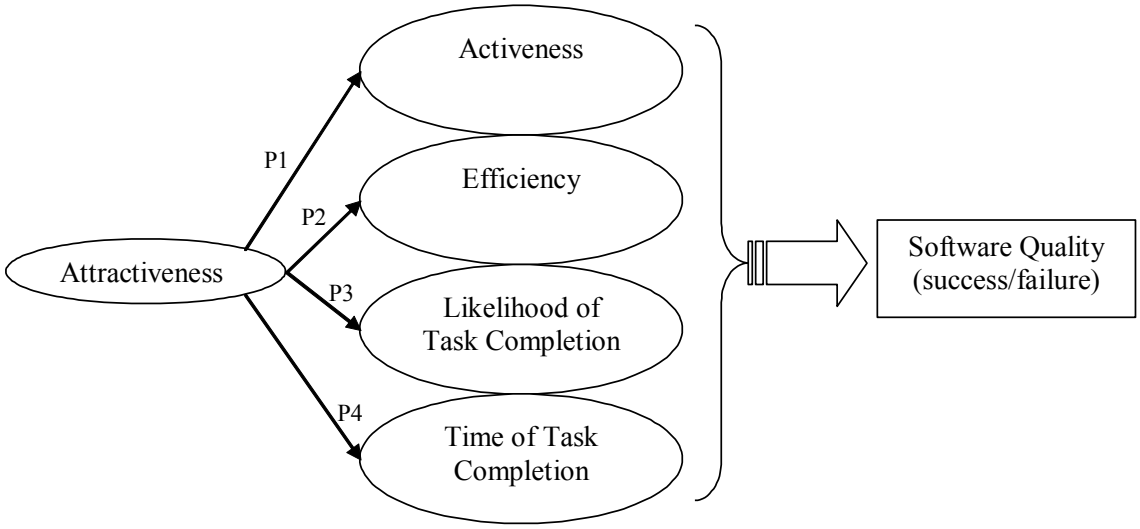


Figure 1 – Endogenous Constructs Theoretical Model

The following section describes, one-by-one, the exogenous variables (project's characteristics) proposed to influence directly each of the identified endogenous constructs. The section finishes with a revision of the first four propositions to account for the expected moderation effect of task complexity. All propositions developed in the next section are depicted in Figure 2.

Exogenous Variables (Endogenous' Influencers)

We recognize that a variety of exogenous variables may influence attractiveness, activeness, efficiency, and likelihood and time of task completion. There is a widely accepted paradigm in the open source software literature that explains voluntary contribution based on contributor's intrinsic motivations, especially to receive compensations such as job offers (Crowston & Scozzi, 2002; O'Mahony, 2007). We do not challenge this view. We recognize it, but also believe that the "soft" nature of this construct makes it too difficult to conduct any

large empirical study that seeks practical managerial knowledge development. Alternatively, we argue that one's intrinsic motivation drives him or her to a project with such characteristics, which we study, that fulfill one's intrinsic motivations. Additionally, we believe that besides one's intrinsic motivations, one's technical knowledge (expertise) also plays an important role in explaining his likelihood to join and contribute to a project, as well as at what speed.

Based on that, we focus on "hard variables" and attempt to identify which project's characteristics are able to fulfill more, or less, intrinsic motivations than others. These "characteristics of the application" have been suggested in the literature to be linked with OS project's likelihood of success in the past (Crowston & Scozzi, 2002).

Below we present four project characteristics capable of influencing the endogenous variables of focus to this dissertation directly, and another variable said to moderate the relationships between attractiveness and the other endogenous variables.

Project Characteristics

License Type

Every software part of a “going-open” strategy 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 alterations. For instance, no source code under the General Public License (GPL) can be used in commercial software. Other licenses such as the Mozilla Public and the Eclipse Public are not as restrictive as GPL, permitting a better interaction between open and proprietary software (Fershtman & Gandal, 2007). To date, there have been no studies focused on describing their similarities and differences in details. Nevertheless, the interested reader might visit www.opensource.org/licenses for a thorough presentation of each license recognized by the open source software foundation (total 71), their scope, and characteristics.

The influence of a project’s chosen license (e.g., GPL, Berkeley Software Distribution, or Sun Industry Standards Source License) on its activities has been documented in the literature (Stewart, Ammeter, & Maruping, 2005). For example, Lerner and Tirole (2005) have examined how this choice is related to the project’s target population, developers or end users. Moreover, Fershtman and Gandal (2007) have found that the type of license is associated with the number of contributions received by an OS project.

The rationale behind this argument is that members’ willingness to voluntarily contribute should be linked with the restrictions and allowances of

each license type, which could go, for example, against one's intrinsic motivation by restricting the visibility of one's work. A negative correlation between one's motivation and a restrictive ("less commercial potential") license is suggested, given the assumption that volunteers' intrinsic motivation is to promote their work (Fershtman & Gandal, 2007).

The influence of type of license on a project's outcomes has been discussed at the individual level, affecting a member's likelihood to contribute (Fershtman & Gandal, 2007), but not at the project level. We propose that this choice should have similar impacts at the project-level. We will not classify different types of licenses according to anything such as restrictiveness, given the lack of such information for all of them (Fershtman & Gandal, 2007). Accordingly, we will simply classify projects according to their type of license and do the appropriate analysis, leaving any inference on why they differ or not to a later moment, after empirical analysis, possibly suggesting hypothesis that are more specific. For now, the following propositions will guide our empirical analysis.

Proposition 5.1: A project's license type is a significant influencer of its attractiveness.

Proposition 5.2: A project's license type is a significant influencer of its activeness.

Proposition 5.3: A project's license type is a significant influencer of its efficiency.

Proposition 5.4: A project's license type is a significant influencer of its overall likelihood of task completion.

Proposition 5.5: A project's license type is a significant influencer of its average time to complete tasks.

Intended Audience

Every software project has a stated purpose or specific goals. It exists to support an organizational process or a set of processes. These benefited processes, especially inside an organization, affect the work of a person or of a group of people. This group benefited directly with the use of the software is called the "intended audience". Our motivation to insert this variable in our model is similar to the idea of niche in marketing. The bigger the niche, the higher one should expect sales to be, everything else constant. Or, in the context of open source projects, some of them "have a greater number of potential developers in the community than others do" (Johnson, 2002, p. 664, p. 664). Similarly, these different types of audiences should attract different numbers of members as well as specific types of members (e.g., system administrators) which we argue to be linked to one's expertise and then one's likelihood to contribute.

The influence of intended audience on OS projects has been discussed in the literature. Crowston and Scozzi (2002) pointed out that projects that have their software targeted to developers tend to be more active than ones that are targeted to systems administrators, which in their turn, tend to outperform ones

targeted to a non-technical audience (users). Additionally, these same authors pointed out that there are more OS projects with software aimed at fulfilling developers' and end users' than system administrators' needs. Therefore, these different projects' population (niche) sizes are likely to influence their competitive dynamics for recruiting contributors.

Furthermore, Fershtman and Gandal (2007, p.222) have observed that "[o]utput per contributor in projects oriented towards end users […] is significantly lower than that in projects for developers." They explained this by pointing out that perhaps software aimed at end users is of less commercial value than those aimed at developers, once again grounded in the member's intrinsic motivation of signaling assumption (Fershtman & Gandal, 2007).

In the OS software projects context, the intended audience of a project can be (1) end user, (2) systems administrator, (3) developer, or any combination of them. We will take the same inductive approach we did for license type for intended audience. Accordingly, we do not hypothesize on the directionality of this influence, although we expect it to occur, as previous research has suggested. Nevertheless, the following propositions will guide our empirical analysis.

Proposition 6.1: A projects' choice of intended audience significantly influences its attractiveness.

Proposition 6.2: A projects' choice of intended audience significantly influences its activeness.

Proposition 6.3: A projects' choice of intended audience significantly influences its efficiency.

Proposition 6.4: A projects' choice of intended audience significantly influences its overall likelihood of task completion.

Proposition 6.5: A projects' choice of intended audience significantly influences its average time for task completion.

Type of Project

Just as OS projects have an intended audience, they also have focus on a specific area. Projects are of a specific type, as Sourceforge.net classifies them. A project's topic is capable of giving another clue of what a project is all about. Projects might be related to genealogy, payroll, online chatting (e.g., ICQ), browser, games, and many others. They can also be of a combination of them, which creates a methodological challenge due to the difficulty of grouping them (Crowston & Scozzi, 2002).

Once again, we argue that the project type is capable of influencing a project's activities because we do not believe people choose their projects at random. Arguably, available topics guide one to a specific area of interest, and they may even account for the attraction of specific groups of people with specific types of expertise, influencing directly project's activities.

The type of project has been discussed in the open source literature. For instance, Crowston and Scozzi (2002, p.18) have pointed out that "projects

(software) dealing with topics that are familiar with developers (such as Internet or communication topics) will be more active and in a more advanced development status than those that address very specific needs, such as religion or scientific/engineering." According to Crowston and Scozzi (2002), the type of project is a significant influencer of project success (i.e., number of downloads and views, development status, and intensity of work).

Furthermore, the influence of project type has been argued to be derived from its link with the project's targeted population, developers or end users (Crowston et al., 2005; Crowston & Scozzi, 2002). Additionally, empirical evidence has shown that the distribution of projects by type is not a balanced one, raising possibilities of different levels of competition in each, given that these project types are capable of separating them by audience. Most projects are of software development or are related to systems topics. Minorities are composed of office or business topics (Crowston & Scozzi, 2002).

We take the approach we did for the other predictors to assess the degree of influence, if any, of type of project. We argue that there should be an influence on our endogenous variables as described in the literature, but we opt to not hypothesize on the directionality of this influence because no attempt to date was made to measure each topic influence on project's activities. Accordingly, it is not possible to guess the direction of such influence for each type of project. Nevertheless, propositions to guide empirical analysis were developed and are presented below.

Proposition 7.1: A project's choice of topic is a significant influencer of a project's attractiveness.

Proposition 7.2: A project's choice of topic is a significant influencer of a project's activeness.

Proposition 7.3: A project's choice of topic is a significant influencer of a project's efficiency.

Proposition 7.4: A project's choice of topic is a significant influencer of a project's overall likelihood of task completion.

Proposition 7.5: A project's choice of topic is a significant influencer of a project's average time for task completion.

Development Status (Project's Life-cycle Stage)

Any software can be classified based on its current state in what the literature refers to as the software life-cycle. This status of the software is often used as a strategy. For example, corporations often release beta versions of their software so that people can evaluate it and provide feedback to them. In doing so, the organization safeguards its image in case bugs are found, given that an assumption of a beta version is that it is a prototype and not a ready-for-the-market piece.

Most OS projects maintain their software status readily available to their members. Potentially, this status might influence one's decision to join and contribute to a project as one evaluates whether that project would be a "good"

one to have his or her image attached to. Additionally, projects at different stages might show different levels of activeness and efficiency, as it might affect its members' motivation to release a new version, for example. Based on that, we decided to incorporate a project's development status in our model and empirically test its degree and direction of influence in our endogenous variables.

The relationship of the software development status (e.g., beta, stable, or production) with the success of an open software project has been previously discussed (Crowston & Scozzi, 2002; Raja & Tretter, 2006; Stewart & Gosain, 2006). Mainly, these studies discussed a project's ability to advance throughout its development statuses as success, being predicted by variables such as project audience and developer ideologies. That stream of research argues that a more mature project is more successful. Our view differs from this in that we argue that these changes in a project's development status may influence attractiveness, activeness, and efficiency, but do not necessarily indicate success, given that this influence, theoretically, might be negative to them. Furthermore, most of the software that has been considered successful (such as Windows or Linux) has achieved the last status on their life cycle (i.e., stable); nevertheless they are still (and have to be) active and efficient, as understood in this paper, to maintain or enhance their market positioning.

Simply put, an inactive or inefficient project leads to failure in the long run from a marketing perspective no matter what the project current life-cycle stage is. Additionally, we do not expect that these changes from one life-cycle status to another will influence a project's attractiveness, activeness, efficiency, task

completion, and time for task completion equally. Therefore, we will analyze each possible life-cycle stage separately, looking for clues on whether a project status influences its dynamics and to which direction.

This approach can potentially provide hints to managers on what to expect as their projects evolve throughout different phases. We believe that the directionality of this influence should be analyzed on a case-by-case basis, and therefore opt to not hypothesize on the directionality of these relationships, adopting an inductive approach for theory development. Nevertheless, propositions were developed to guide the empirical analysis.

Proposition 8.1: A project's development status is a significant influencer of a project's attractiveness.

Proposition 8.2: A project's development status is a significant influencer of a project's activeness.

Proposition 8.3: A project's development status is a significant influencer of a project's efficiency.

Proposition 8.4: A project's development status is a significant influencer of a project's overall likelihood of task completion.

Proposition 8.5: A project's development status is a significant influencer of a project's average time for task completion.

The Moderation Effect of Task Complexity

Social-cognitive factors such as affective trust were found to significantly predict an OS project's size (i.e., number of members) and their level of activeness (Stewart & Gosain, 2006). However, contrary to our argument that attractiveness leads to activeness and efficiency, attractiveness (represented by the number of a project's members) was not able to explain task completion or team effort significantly (Stewart & Gosain, 2006). Therefore, although the presence of members is a necessary condition to generate outcomes, this finding suggests, at least, that a linear relationship is not likely to exist between the number of members and a project's activeness and efficiency levels. Accordingly, the inclusion of other variables in the analysis is necessary to explore this issue further.

We argue that the reason Stewart and Gosain (2006) did not find a significant relationship between number of members and a project's outcomes is because of a factor not accounted for in their paper, the complexities of the software (tasks). In an attempt to clarify this unexpected finding, which goes against what we predicted in propositions 1 to 4, we include this complexity measure in our model. The complexity of the software can be partially captured by the degree of interdependence among software tasks (C. R. B. DeSouza, Redmiles, Cheng, Millen, & Patterson, 2004a; C. R. B. DeSouza, Redmiles, Cheng, Millen, & Patterson, 2004b).

Stewart and Gosain (2006) proposed that the number of members does not influence a project's activities because of differences in expertise among volunteers, or to an unbalanced distribution of expertise across projects. Accordingly, we have included in our model a variable that separates projects based on different categories, audiences, and license types. We argue that one should expect to find similar members' expertise within each of these groups of projects (e.g., clustering or genealogy), also separated by license and intended audience. Furthermore, the high number of unexpected findings in Stewart and Gosain's (2006) study is encouraging of replications and alternative answers for the same question of what predicts activeness, efficiency, task completion, and time for task completion.

Software modules (tasks) might be dependent on each other in several ways. It is the degree of this interdependence among modules that is referred to as interdependence or cohesion in the literature. Baldwin, Carliss and Clark (2003) demonstrated that modular projects have advantages in recruiting contributors. They grounded their argument on the reasoning that the more modules or slices one project has, the more opportunities it offers, increasing members' likelihood to contribute. However, one thing that cannot be set aside is that the quantity of modules is expected to influence interdependence, which is assumed here to be a proxy for task complexity, thereby inverting the relationship. A potential trade-off exists here.

Especially in the open software case, in which people work geographically dispersed and the main communication medium is e-mail or list-servers,

interdependence might be expected to be one of the main factors for attractiveness (West & O'Mahony, 2005). Moreover, the relationship between interdependence and contributions was studied by DeSouza et. al. (2004a) and DeSouza et. al. (2004b). They demonstrated that software with low interdependence is more likely to receive contributions than those with high interdependence. That is because low interdependence facilitates the source code inspection function (debugging), software testing, comprehension, maintenance and parallelization (Counsell & Swift, 2006; Xu, Qian, Zhang, Wu, & Chen, 2005). Furthermore, Koch and Schneider (2002), in their study of the GNOME project, found that developers worked in isolation on different modules, which is arguably essential for parallel and distributed activities such as in the case of OS projects (Fitzgerald & Feller, 2002). Finally, Stamelos et al. (2002) have discussed the importance of modularity and of a well structured piece of software on open source type of endeavors. According to Fitzgerald and Feller (2002), this is not a surprising conclusion, given the connection of modularity and productivity is part of the basic principles of software development.

In accordance with the literature, we argue that volunteers prefer to be able to work independently and on simpler tasks. So, at first glance, one should expect administrators of "going-open" strategies to build systems with modules (tasks) interdependences as low as possible because of the relationship between interdependence and source code programming and learning difficulty. Thus, a manager's decision regarding the degree of interdependence is crucial and complex. And for being complex, rationality may fail to prevail for different

reasons in such a decision. First, features are requested by members and therefore are not in control of the manager (software development activities are expected to suffer pressure from users for richness). Second, market strategies might push deadlines (Mockus & Herbsleb, 2002). And third, a version of the software might have been already developed when the decision to release it open is taken.

In sum, two factors, quantity of modules (quantity of opportunities or tasks offered) and their degree of dependencies on other modules, which is used as proxy for task complexity, are expected to influence each other and, consequently, the number of contributions received (i.e., activeness and efficiency) due to the increasing costs of 'understanding' the task. Thus, we revise our first four propositions to the following form and show them in Figure 2.

Proposition 9: The overall complexity of a project's tasks moderates the relationship between attractiveness and activeness.

Proposition 10: The overall complexity of a project's tasks moderates the relationship between attractiveness and efficiency.

Proposition 11: The overall complexity of a project's tasks moderates the relationship between attractiveness and overall likelihood of task completion.

Proposition 12: The overall complexity of a project's tasks moderates the relationship between attractiveness and average time of task completion.

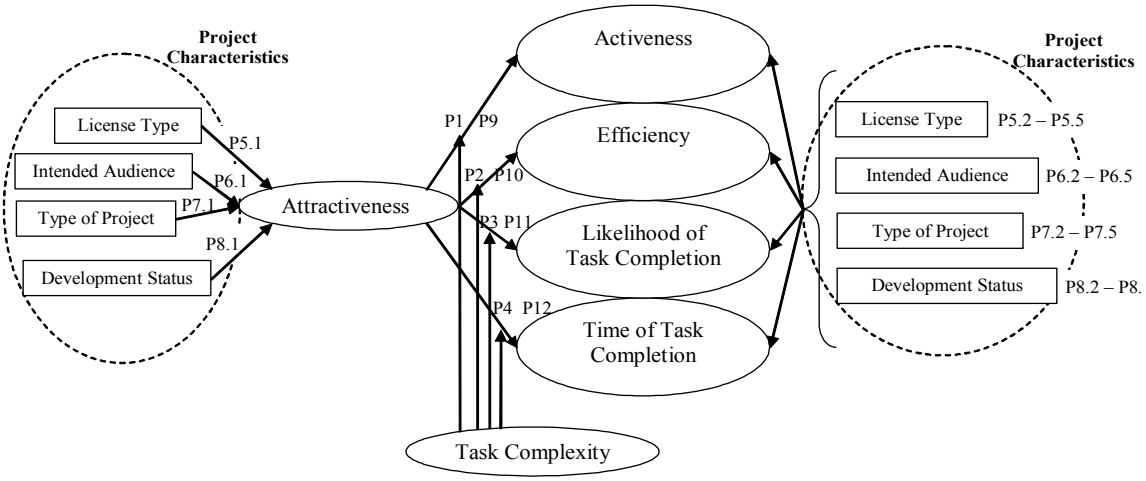


Figure 2 – Complete Theoretical Research Model

With the theoretical model developed and depicted in Figure 2, we will now turn to the discussion on how such model will be empirically tested. Chapter 3 has a detailed description on data collection procedures and statistical analysis.

CHAPTER 3 – METHODS

Description of the Website, Data Collection and Analysis Procedures

Data Source

The variety of OS projects is extremely large. On one extreme there is a project of the magnitude of Linux, and its kernel, that is supported by the Open Source Development Labs (OSDL), "a consortium formed by high-tech companies which include IBM, Hewlett-Packard, Intel, AMD, RedHat, Novell and many others."[19] On the other, there are thousands of individually-owned projects hosted and not hosted on websites like Sourceforge.net, Freshmeat.org and Tigris.net. These are "projects diverse in size, application domain and audience" (Raja & Tretter, 2006, p.4).

The theoretical model developed in this paper will be tested against the Sourceforge.net database available for academic inquiry free of charge through the University of Notre Dame. We chose SourceForge.org because it is the largest repository of OS projects (Koch, 2004). It contains a diverse population of over 170,000 projects and 1.8 million registered members. Additionally, the readily available database for queries at the University of Notre Dame website provides an ideal opportunity for data collection and analysis to test theories (Crowston & Howison, 2006). Because of that, this source of data has been used by Crowston and Scozzi (2002) when testing their theory of competency rallying,

---

[19] http://www.linux.org/info/linus.html

by Hunt and Johnson (2002) aiming to explain the number of downloads projects had, by Krishnamurthy (2002), who focused on the 100 most active projects, by Stewart and Gosain (2006), and others[20]. The data is available on a project by project, month by month basis (Fershtman & Gandal, 2007).

Sample and Time Frame

For the purposes of this research, we will filter the over 140,000 projects in the database and filter it to a more interesting sample. Accordingly, we decided to limit our analysis to the projects that conform to the two criteria that follows. First, we will exclude any project with only one member as these projects have not shown any attractiveness to anyone besides to its founder. Second, we decided to exclude any project that shows missing data on any of the variables we intend to collect. To ensure this last point, we will record only the identification numbers of those projects that have received at least one output (efficiency), implying then that they have already received at least one input (activeness).

We decided to avoid data collection from any time period before 2006 due to purges and table redefinitions that occurred in the database in that period, as reported on the website managed by the University of Notre Dame. Also, given the amount of manual work that has to be done to collect all information needed, we decided to limit our analysis to only three points in time, regardless of all others available. Furthermore, we decided to space our data collection by one

[20] See https://zerlot.cse.nd.edu/mywiki/index.php?title=Papers

year from each other to allow enough time for the website composition as well as its projects dynamics to change substantially. Accordingly, our data collection activities will gather data from January/2006, January/2007, and January/2008.

Repeated cross-sectional studies are known to give researchers higher confidence on the results because this approach works as a replication of the empirical test at the same time that it finds better estimates. In summary, with results of the same tests using data from different points in time at the same time, a researcher can evaluate how valid his proposed model is over time, reduce the likelihood of data collection errors, and increase his confidence in the estimates found (Hair, Black, Babin, Anderson, & Tatham, 2006).

Database General Characteristic

Previous studies that have utilized this database have found that a series of its variables show high skewness[21], a characteristic that goes against many statistical tools' assumption of normality. Specifically, Crowston and Scozzi (2002) found that the distributions of number of downloads, page views, developers and administrators, and count of projects using a given programming language were highly skewed. They transformed these variables into their logarithmic function so to fix this problem. We intend to follow the same steps, if non-normality appears as an issue.

---

[21] "Measure of the symmetry of a distribution." (Hair et al., 2006, p. 40)

Endogenous Variables – Measurement and Location at the Database

Attractiveness

Attractiveness is composed of three distinct empirical measures. The number of (1) *hits* (*page views*) the project has had; (2) the number of times the project has been *downloaded*; and (3) the number of *members* the project has. These three pieces of information are available in the Sourceforge.net database as shown in Table 1.

Activeness

Every time a project is created in Sourceforge.net, it automatically receives four trackers[22]. These are (1) *bug reports*, representing the number of times users submitted software bugs to be reviewed and resolved; (2) *support requests*, representing the number of times end-users have made support inquiries; (3) *feature requests*, representing the number of times users submitted requests for software enhancement; and (4) *patches submitted*, representing the number of software source code extracts submitted for review. Accordingly, this construct will be measured as the sum of (1) the number of bugs reported, (2) the number of features, (3) support requested, and (4) the number of patches submitted. Based on our decision to not include projects with missing data, we

[22] http://alexandria.wiki.sourceforge.net/Tracker+-+Bug+Reporting,+Support+Requests,+Feature+Requests,+Patches

will collect all information *available* to each project and exclude projects without data on any of these four trackers.

Sourceforge.net stores information on these four trackers under a table in the database called “artifacts”. Although it is possible to classify an artifact according to its type (e.g., bug or request), it is also very time consuming and complicated because the information is in a text field requiring a content analysis. Fortunately, this separation is not of our interest here. Thus, we will collect these data as Sourceforge.net provides it to us straightforwardly, as the sum of the number of bugs, feature and support requests, and patches submitted for each project. These four pieces of information are available in the Sourceforge.net database as described in Table 1.

Efficiency

The Sourceforge.net tracker system supports the management of software development activities of the projects. The tracker manages software issues through changes in their status. Four statuses are accepted: open (1), closed (2), pending (3), and deleted (4)[23]. For our purposes here, only those assigned closed status will be included in the measure of efficiency, given that “open”, “pending” or “deleted” tasks are included in our measures of activeness. Therefore, first bugs receive status of “open” and later, when fixed, they have their status changed to “closed”. Similarly, support requests are provided, feature

[23] Coding information extracted from the actual table. 1-Open, 2-Closed, 3-Deleted, 4-Pending.

requests are added to the software, and patches are analyzed and closed. Accordingly, efficiency of these projects will be assessed by measuring four different variables: the number of (1) *bugs closed*, (2) *features added (closed)*, (3) *patches closed*, and (4) *support provided (closed)*. Again, not every project has data on all of them, forcing us to focus on what they have in the database. As with activeness, efficiency measures will be collected as one variable resulting from the sum of the number of bugs, features, patches, and support closed. These data are available in the Sourceforge.net database as described in Table 1.

Likelihood of Task Completion and Time of Completion

The measure for likelihood of task completion is the result of the sum of all measures of efficiency divided by the sum of all measures of activeness. This measure provides the probability of a project to solve its issues such as to close a reported bug or to develop a requested new feature for the software. Similarly, every time a created task (activeness) is completed (efficiency), a certain amount of time was required to do so.

Time of completion will be calculated as the average amount of time a project takes to complete the tasks it does complete. Time stamps for tasks can be found in the Sourceforge.net database. Therefore, time intervals can be calculated by subtracting one from the other, and the average can be computed

by summing up all time intervals and dividing this sum by the number of completed tasks.

In exploring these variables and tables to ensure the possibility of successful data collection, we found that some of the artifacts are “closed” but have a negative result in the operation (close date – open date). The *artifacts* table has a default value on the close_date field of 0. Therefore, when people don’t fill it out, the information becomes inaccurate. We will exclude any negative values from our dataset. Moreover, the time format in the Sourceforge.net database is the Unix time format. It indicates how many seconds since January 1, 1970 have passed. This format should not be a problem, given that a measure in seconds to complete tasks would fit as well as any other measure such as number of days. Additionally, we can transform this time intervals as we wish. Details for these two variables are found in Table 1.

Table 1 – Endogenous Constructs: Composition and Measurement

Construct	Empirical Measures	Database Table.Field	How to Get the Data (SQL Queries and other transformations)
Attractiveness	Page view(hits)	Top_group.pageviews_proj	Select top_group.pageviews_proj , top_group.group_id
	Downloads	Stats_groupid_alltime_agg.downloads	Select stats_groupid_alltime_agg.downloads, stats_groupid_alltime_agg.group_id
	Members	User_group.count(*)	SELECT count(*), group_id FROM user_group group by group_id
Activeness	Bugs reported	Count (artifact.status_id)	SELECT artifact.group_artifact_id, artifact_group_list.group_id, artifact.status_id FROM sf0108.artifact, sf0108.artifact_group_list WHERE
	Features requested		
	Support requested		

Table 1 (continued) – Endogenous Constructs: Composition and Measurement

	Patches submitted		artifact_group_list.group_artifact_id=artifact.group_artifact_id <<Total number of Artifacts >> THEN Aggregate them by project (counting the number of tasks-artifacts). Replace missing values by 0s.
Efficiency	Bugs closed Features added Support provided Patches Analyzed	Count (artifact.status_id= 2)	SELECT artifact.group_artifact_id, artifact_group_list.group_id, artifact.status_id FROM sf0108.artifact, sf0108.artifact_group_list WHERE artifact_group_list.group_artifact_id=artifact.group_artifact_id and artifact.status_id = 2 <<Artifacts closed by project>> THEN Aggregate them by project (counting the number of tasks-artifacts). Replace missing values by 0s.
Overall Likelihood of Task Completion	(bugs closed + features added + support provided + patches analyzed) DIVIDED BY (bugs reported + features requested + support requested + patches submitted)		Efficiency DIVIDED BY Activeness

Table 1 (continued) – Endogenous Constructs: Composition and Measurement

Average Time of Task Completion	∑(date_close – date_open) :::: For each bug, features, support, and patches. DIVIDED BY Total number of completed tasks (artifacts).	SELECT artifact.group_artifact_id, artifact_group_list.group_id, artifact.close_date - artifact.open_date, artifact.status_id FROM sf0108.artifact, sf0108.artifact_group_list WHERE artifact_group_list.group_artifact_id=artifact.group_artifact_id and artifact.status_id=2 and close_date - artifact.open_date > 0 <<Each line represents a completed task with its respective time to be completed>> THEN <<Group dataset by project counting the number of tasks-artifacts and their respective time of completion>> THEN <<Divide the total time of completion by the number of tasks>>

Exogenous Variables

Project Characteristics

For each project characteristic (i.e., license type, intended audience, type of project, and development status), Sourceforge.net has assigned one independent category codes. For example, each development status has a different assigned code (e.g., production/stable is coded 11). Every project has a code as well. For instance, a project for the development of the software

Sourceforge.net (web site) is coded 1. Then, Sourceforge.net keeps a table called *trove_group_link* in its database that works as a link between projects and their categories. In looking at that table, one is able to find one record that contains both 1 for project identification code and 11 for production/stable development status, indicating that the project is under the production/stable development status. The database works similarly for the intended audience, type of project, and license type. Accordingly, to capture these effects, we will create one variable for each category of interest and populate it with a "1" value whenever the project has such characteristic represented by the variable; otherwise with a "0". Each of these project characteristics are discussed next.

License Type

For license type, Sourceforge.net allows projects to be of different kinds. We were able to identify 12 of them (see Table 2). Therefore, for license type, we will create 12 variables in our dataset where a project that is registered under GPL only, for instance, would score 1 in the variable GPL and 0 in all other eleven. Details on collection of this information are in Table 4.

Table 2 – License Types and Codes in the Database

License Type	Code
Open source initiative (osi)	14
General public license	15
Gnu library or "lesser" general public license (lgpl)	16
Artistic license	17
Mozilla public license (mpl)	304
Mozilla public license 1.1 (mpl11)	305
Apple public source license (apsl)	306
Berkeley software distribution (bsd)	187
MIT	188
Python software foundation license (psfl)	189
Q public license (qpl)	190
Ricoh source code public license (rscpl)	193

Intended Audience

For intended audience, we will have to create four variables, one for each possible audience the project intends to attract for use or development (end-user, system administrator, software developer, and other). These categories are coded 2, 4, 3, and 5, respectively in the *trove_group_link* table. Further details can be found in Table 4.

Type of Project

The immense variety of types of projects allowed by Sourceforge.net hinders statistical analysis (approximately 162 categories). Fortunately, Sourceforge.net provides these project's topics hierarchically. As Crowston and Scozzi's (2002) explained, the category "Web browsers" is under "Internet" and so forth. Accordingly, based on Crowston and Scozzi's (2002) approach to deal with this issue, we will focus our analysis on the 19 first-level categories.

Therefore, 19 variables are necessary to capture this information. These categories, the quantity of projects within them, and their respective codes are listed in Table 3 in alphabetical order. Further details on measurement and collection can be found in Table 4.

Table 3 – First-Level Types of Projects[24]

Category Description	Number of Projects	Assigned Code
Communications	24,381	20
Database	9,262	66
Desktop Environment	5,011	55
Education	7,240	71
Formats and Protocols	4,771	611
Games/Entertainment	24,051	80
Internet	36,740	87
Multimedia	21,048	99
Office/Business	14,496	129
Other/Nonlisted Topic	4,136	234
Printing	664	154
Religion and Philosophy	480	132
Scientific/Engineering	21,417	97
Security	4,367	43
Sociology	546	282
Software Development	39,920	45
System	29,802	136
Terminals	867	156
Text Editors	4,227	63
TOTAL	**253,426**[25]	**N/A**

Development Status

Our approach to deal with development status is similar to license type's, intended audience's, and type of project's ones. Development status is also a name given for a group of categories with codes from 7 to 12. Respectively,

[24] Information collected from the Website Sourceforge.net on 02/25/08.
[25] This number is higher than the total number of projects (roughly 160,000) because projects assign themselves to more than one of these categories often.

these codes refer to Planning, Pre-alpha, Alpha, Beta, Production/Stable, and Mature. Accordingly, six variables will have to be created to give us capability to separate them in groups pertaining to one or more of these statuses. Further details are given in Table 4.

Task Complexity

Sourceforge.net stores information related to every task a project decides to manage through its provided tools. By exploring the database, its tables and their relationships, we discovered that projects create “projects”, which are composed of tasks, which in turn depend on other tasks to be successfully completed. Examples of “projects” created by these communities are “Baytek Module”, “X10 Module”, “Match and cube evaluations”, “optimize”, and “To Do List”[26]. Again, these “projects” are composed of tasks and each one of their tasks depends on a certain number of other tasks, indicating their degree of complexity.

Given that a contributor’s contribution would occur ultimately in a task(s), and only indirectly to the project, it is the degree of interdependence of a project (software) tasks that we should be after to represent its complexity to the contributors (members) in general. In the *project_group_list* table one finds information to which project a specific “project” belongs. Furthermore, there is a table in the database called *project_task* that contains information on tasks such

[26] Information extracted from the table *sf0108.project_group_list* of the database by the authors.

as to which “project” this task belongs, what its completed percent is, and so on. Additionally, there is another table called *project_dependencies* that contains information on which other task(s) a specific one depends on.

With these three pieces of information, we are able to count how many tasks a project has and on how many other task(s) each one is dependent, making it possible to derive an overall measure of any project’s complexity. To calculate this derived overall index of complexity for each project, we will first take each project (software) task and count the number of other task(s) it depends on (i.e., one task dependency index). Then, we will sum up all task dependency indexes of a project and divide it by the number of tasks a project has, resulting in an overall project’s task complexity measure. By doing this, we expect to create a standardized measure of each project task complexity in general. Details on calculations and data collection are in Table 4.

One specific limitation of this approach is noteworthy. Sourceforge.net does not provide documentation for the database besides its E-R diagram (see Appendix A) to researchers or to the University of Notre Dame. Nevertheless, we could infer by analyzing the E-R diagram that what Sourceforge.net calls tasks is not the same thing as artifacts (e.g., bug reports or feature requests), which we used to build our activeness and efficiency indexes. Therefore, the degree of dependency developed here is related only indirectly to project’s artifacts, as we found out through email conversations with the IT staff at the University of Notre Dame. These dependencies are related to tasks that may contain one or many bug reports or feature requests (i.e., artifacts), for example. Nonetheless, this

derived complexity measure is related to the project's (software's) source code, which is where developers work. Additionally, we are not interested in a measure of complexity for each artifact, but to the whole project. Therefore, we argue that our task complexity measure can be used as a proxy to a project's tasks complexity in general.

As project complexity appears in the model as a moderator, to facilitate the statistical analysis, we will classify our sample projects into categories to represent their complexity, based on its overall project complexity index. Accordingly, we will group projects as "very simple" when their overall task complexity scores are 0 or 1, as "simple" when scores are 2 or 3, as "complex" when 3 or 4, and as "very complex" when 5 or more.

Although we found prior studies dealing with task complexity, we had to create our own grouping method due to the lack of previous literature aiming at defining what correspondence should be used between the number of other tasks a task is dependent on, and its level of difficulty in the case of software development, as judged by developers. Jiang and Benbasat (2007), for example, dealt with task complexity in the context of proper understanding of products by online customers. Nevertheless, we do not believe Jiang and Benbasat's (2007) measure could be directly applied to our study of software development activities, given that it aims at end-users. They dealt with people's memorization capabilities. Therefore, we had to develop our own grouping procedure.

Control Variable (Project Life-span)

As an attempt to rule out other concurrent explanations for our endogenous constructs, based on previous studies, we will control for age of project. According to Fershtman and Gandal (2007) and Crowston and Scozzi (2002), one should expect age (lifespan) of the project to be an influencer of projects' activities due to maturity.

To control for age of project, we will collect information on when the project was registered on Sourceforge.net, subtract this date value from the current date, and utilize the project age in days as a continuous variable for analysis matter (see Table 4 for details).

Table 4 – Exogenous Constructs: Composition and Measurement

Construct	Empirical Measures	Database Table.Field	How to Get the Data (SQL Queries and other transformations)
Project Characteristics	License Type	Trove_group_link.trove_cat_id=14(osi) Trove_group_link.trove_cat_id=15(gpl) Trove_group_link.trove_cat_id=16(lgpl) Trove_group_link.trove_cat_id=17(artistic) Trove_group_link.trove_cat_id=304(mpl) Trove_group_link.trove_cat_id=305(mpl11) Trove_group_link.trove_cat_id=306(apsl) Trove_group_link.trove_cat_id=187(bsd) Trove_group_link.trove_cat_id=188(mit) Trove_group_link.trove_cat_id=189(psfl) Trove_group_link.trove_cat_id=190(qpl) Trove_group_link.trove	SELECT trove_cat_id, root_parent, group_id FROM sf0108.trove_group_link <<All projects and all their categories>> <<Filter for a specific category and create a different dataset>> <<Eliminate duplicates>> <<From the just created dataset, import all variables from the main dataset>> <<The new dataset then becomes the main dataset>> <<Recode the new

Table 4 (continued) – Exogenous Constructs: Composition and Measurement

		_cat_id=193(rscpl)	variable with 0s or 1s>>
	Intended Audience	Trove_group_link.trove _cat_id=2(endusers) Trove_group_link.trove _cat_id=3(developers) Trove_group_link.trove _cat_id=4(sysadmins) Trove_group_link.trove _cat_id=5(other)	
	Type of Project	19 different codes (see Type of Project section).	
	Development Status	Trove_group_link.trove _cat_id=7(planning) Trove_group_link.trove _cat_id=8(prealpha) Trove_group_link.trove _cat_id=9(alpha) Trove_group_link.trove _cat_id=10(beta) Trove_group_link.trove _cat_id=11(production/ stable) Trove_group_link.trove _cat_id=12(mature)	

Table 4 (continued) – Exogenous Constructs: Composition and Measurement

Task Complexity	Summation of the Degree of Dependency of each Task DIVIDED BY Number of Tasks	∑(number of tasks a task depend on) ::: per task DIVIDED BY Count(Project_task.project_task_id)	SELECT project_group_list.group_id, project_dependencies.project_task_id FROM sf0108.project_task, sf0108.project_dependencies, sf0108.project_group_list WHERE sf0108.project_task.group_project_id=sf0108.project_group_list.group_project_id and sf0108.project_task.project_task_id=sf0108.project_dependencies.project_task_id <<Each line represents one specific task dependency on another. Therefore, we will have repeated lines for one task that depends on more than one other>> THEN <<Aggregate per task (counting the number of dependencies or lines per task) and saving the group_id>> THEN <<Aggregate by group_id with a SUM of the tasks' degree of dependency and counting the number of tasks, "unweighted number of cases" of task_id>> THEN <<Divide the SUM by the number of tasks. This would represent an overall standardized measure of projects' complexity>>

Table 4 (continued) – Exogenous Constructs: Composition and Measurement

Control Variable	Life Span (age)	Groups.register_time	SELECT group_id, register_time FROM sf0108.groups THEN <<Subtract today's date from the register_time>>

Statistical Analysis and Empirical Evaluation of the Propositions

The number of proposed relationships between our model's constructs led us to Structural Equation Modeling (SEM) as to avoid an enormity of individual regression equations to accomplish a similar empirical exam.

Our model has five categorical variables (license type, intended audience, type of project, development status, and task complexity), and it has six continuous or metric constructs (attractiveness, activeness, efficiency, time of task completion, likelihood of task completion, and project life-span). Moreover, we have categorical exogenous variables, metric endogenous variables, and one categorical moderator; everything over three distinct time periods, a situation somewhat complex (see Figure 3 for the complete measurement model). Nevertheless, according to Hair et al. (2006), SEM is capable of dealing with these conditions using multisample SEM analysis.

According to Chin (1998), SEM is a flexible statistical technique able to model relationships among unobservable constructs (latent variables) such as attractiveness, accounting for model errors in measurements for observed variables. SEM is capable of testing all relationships at once. SEM is the best option when a researcher requires assessment of a series of equations

simultaneously (Hair et al., 2006). Therefore, in the case of this research, SEM is a more efficient technique that allows the researcher to specify and test complex paths rigorously using more than one sample at once (Kelloway, 1998).

According to Hair et al. (2006), the “multigroup SEM” runs the analysis treating different samples as “groups”, providing one chi-square value so that discrepancies in model fit between single-group (one sample) and multigroup (3 samples) options can be assessed. If the difference in chi-square values (model fit) between two options is significant, then “group” membership influence is determined, otherwise, it is not (Hair et al., 2006). For instance, a model setting a particular relationship (attractiveness and activeness) to be equal across all three time periods can be tested against a model allowing the relationship to be estimated separately in the different time periods. In case significant differences between chi-square values are found, the model would be considered inconsistent over time.

A similar approach will be used to test the moderation effect of task complexity. Projects will be grouped based on their complexity and two models will be estimated. One that has the paths between constructs set to be equal across groups, and another that has the paths freely estimated for each group. If the one that estimates the paths freely across groups performs better, evidence for moderation would be found.

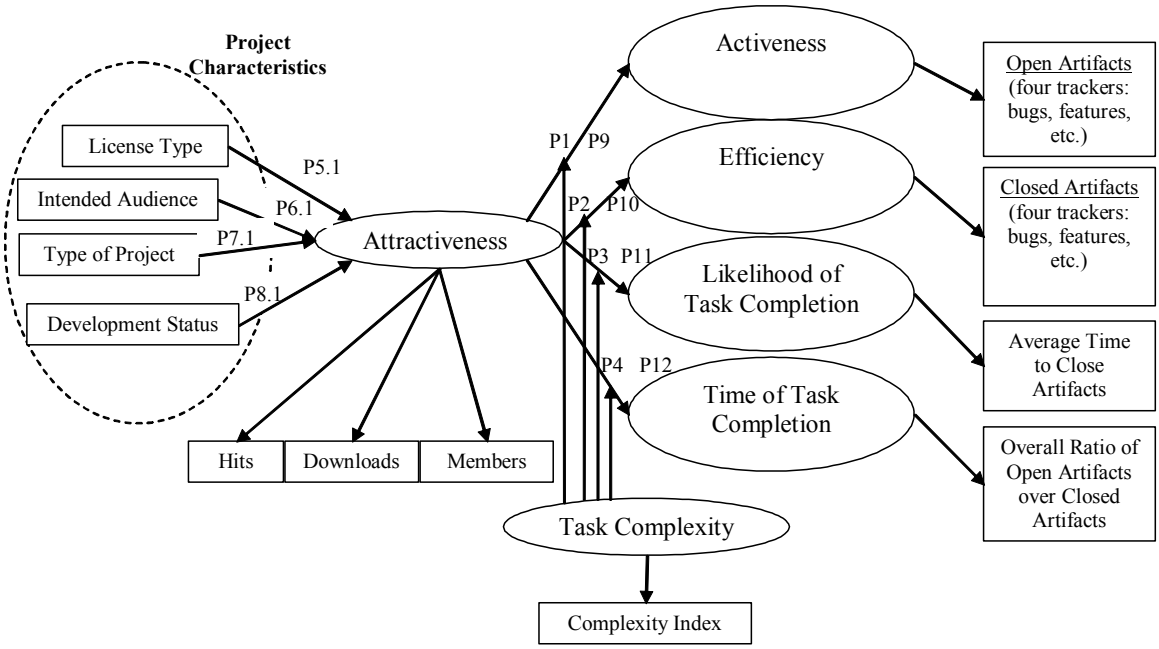


Figure 3 – Measurement Model[27]

Sample Size and Statistical Package

Due to the complexity of the analysis required to test the proposed model, Hair et al. (2006, p.873) suggests that sample sizes have to increase to more than 500 observations, otherwise "problems with model convergence and distortion of the standard errors" are likely to appear. Fortunately, that will not be a problem in the case of this research, where we anticipate about 10,000 usable observations after the data is cleaned (details in the first section of Chapter 4 – Examining the Data).

[27] Propositions 5.2-5.5, 6.2-6.5, 7.2-7.5, and 8.2-8.5 are not represented in the figure because their intent is merely to show how variables are linked with constructs, and that has already been made for propositions 5.1, 6.1, 7.1, and 8.1, which deal with their same constructs.

After cleaning the data, EQS 6.1 for Windows will be used to perform the statistical analysis due to its user-friendliness and robustness. EQS is capable of dealing with different types of input: raw data, correlation or covariance matrices. We decided to input raw data, given that this is the only option that allows statistical corrections to be performed on non-normal data for bias reduction (Bentler, 1989).

SEM does have its assumptions, requiring an evaluation of multivariate normality to be performed. Fortunately, its assessment can be readily made using EQS, Lisrel, or PLS, allowing the researcher to make an informed decision on the estimation method (fitting criteria) to be adopted (i.e., Maximum Likelihood or General Least Squares). Priority should be given to a more robust method (i.e., ML) when the assumption of multivariate normality is violated (Hoyle, 1995). EQS 6.1 provides these statistical correction features straightforwardly. To assess multivariate normality, EQS provides Mardia's (1970) index, which provides a z-statistic. When high values of z-statistics are found (i.e., greater than 1.96), data is considered not normally distributed (Byrne, 1994).

Model-To-Data Fit Method and Evaluation

The most common fitting criterion (coefficient estimation criterion) adopted is maximum likelihood (Anderson & Gerbing, 1998; Hair et al., 2006). Maximum likelihood (ML) is considered efficient with large samples and is capable of dealing with non-normal data (Bentler, 1989; Kline, 1998). Accordingly, we

anticipate using maximum likelihood in this study as we expect to deal with non-normal data.

Having decided the fitting criterion to be used, the statistical analysis can be run for an overall model fit evaluation. Several goodness-of-fit measures are available to the researcher for such evaluation. These measures can be classified in three different groups: absolute, incremental and parsimonious (Joreskog & Sorbom, 1993; Kelloway, 1998).

Absolute fit measures allow assessment of how well the proposed model reproduces the actual data or covariance matrix (Kelloway, 1998). Examples of such type of measures are the likelihood-ratio chi-square test (χ^2), the root mean squared residual (RMR), the root mean squared error of approximation (RMSEA), and the goodness-of-fit index (GFI) (Kelloway, 1998).

The chi-square test result provides a measure of the discrepancy between the sample and the fitted correlation or covariance matrices. When one finds a non-significant χ^2, no discrepancy is found and therefore good fit exists. RMR also provides a discrepancy measure (square root of the mean of the squared discrepancies) between the implied and the observed correlation or covariance matrices, assessing how well a model fits the actual data. RMR values of less than 0.05 are suggestive of good model fit (Kelloway, 1998). Likewise, RMSEA values, based on an analysis of residuals, below 0.10 or 0.05 are suggestive of good and very good model fit, respectively (Steiger, 1990). Finally, providing a comparison of the squared residuals from the predicted and the actual data, GFI values range from 0 (poor fit) to 1 (perfect fit). As GFI value increases, so does a

model goodness-of-fit (Hair et al., 2006). In our results section (Chapter 4), we will present and compare all these measures for our models.

Incremental fit measures are of interest to us due to our multisample SEM analysis strategy adopted to evaluate model consistency between project groups and over time. These measures provide comparative goodness-of-fit information between competing models (Hair et al., 2006; Kelloway, 1998). Examples of such measures are the adjusted goodness-of-fit index (AGFI), Tucker-Lewis index, the normed fit index (NFI), the relative fit index (RFI), the incremental fit index (IFI), and the comparative fit index (CFI). AGFI is an adjusted by the degrees of freedom measure of the GFI. Moreover, Tucker-Lewis, or nonnormed fit index (NNFI), provides a comparative index, as NFI, RFI, IFI, and CFI values do, ranging from 0 (no fit at all) to 1.0 (perfect fit) (Hair et al., 2006).

Parsimonious fit measures were created for comparisons between varying numbers of estimated coefficients, rather than overall model fit. Examples of such measures are the normed chi-square, the parsimonious normed fit index (PNFI), and the parsimonious goodness-of-fit index (PGFI). PNFI and PGFI account for degrees of freedom in their calculations and their results range from 0 (no fit at all) to 1 (perfect fit).

Additionally to the measures discussed above, EQS 6.1 provides corrected goodness-of-fit statistics when non-normal data is utilized. These measures are called robust statistics. When a researcher runs a robust analysis, EQS provides a robust chi-square (χ^2) called the Satorra-Bentler scaled statistic (S-Bχ^2) and robust standard errors (Byrne, 1994). Moreover, EQS would still

provide all regular indices such as Bentler-Bonett Normed Fit Index (NFI), Bentler-Bonett Nonnormed Fit Index, Comparative Fit Index (CFI), and Root Mean Square Error of Approximation (RMSEA), however adjusted for the non-normal data. We will decide on whether to utilize the robust option after we evaluate normality of the data.

Evaluation of Construct Reliability and Propositions

To analyze the measurement model results, an approach described by Hair et al (2006) will be adopted. This approach consists of evaluating indicators loadings on their proposed factor (construct), when more than one indicator for a construct exists. To accomplish such a test, construct reliability analysis through Cronbach's alpha can be made. Measures are said to be reliable as their indicators show high internal consistency (Hair et al., 2006). Usually, Cronbach's alpha values greater than 0.70 are considered acceptable (Hair et al., 2006). In the case of our model, we will assess only one of our constructs (attractiveness) reliability, given that it is the only one that has more than one indicator (see Figure 3). All other constructs have one indicator, forcing us to assume acceptable reliability based merely on the logic of the design of the study.

To test each proposition, the structural model fit will be evaluated. This evaluation will occur based on each proposed relationship's (path's) coefficient value and statistical significance (p-values smaller than 0.05). Each path coefficient will be discussed in the context of our developed propositions,

presenting support, or lack of, for each one of them. This approach, we believe, will give us grounds to develop a general theory about OS project dynamics, providing an empirical test of our model that will guide us towards the revision of our propositions to increase their generalizability (Chapter 5).

Of course, SEM is not free of limitations. As its main one, SEM by itself is not capable of guaranteeing causal relationships as suggested in a model. As with any statistical tool, SEM results may support correlations, or associations, between constructs at best. Only theory and logic can accomplish such task of avoiding omitted variables, or mistaking cause for effect, to claim a causal relationship.

Expected Contributions

First, open source software was adopted by organizations and proved to be a viable alternative for them. As these software applications became well known in the marketplace, their communities also did, becoming a source of new ideas and possibilities to organizations in their developing software endeavors.

Although attempts to interact with the open source community by corporations and to study them by academics have already been made, proper understanding of the dynamics of these relationships and their impacts on both sides involved are yet to be reached. This research is a step in that direction. We focus on software of many kinds released open to the public so to evolve with the

help of others, especially volunteers; just like Linus Torvalds once did with what is known today as the operational system Linux.

Additionally, we believe that this research is capable of shedding light onto a theoretical dilemma, where one stream claims that quantity increases diversity, and that diversity is a good thing for problem-solving ("the more eye balls, the better"); and another, that points out recent empirical evidence suggesting that the relationship between the size of a community (number of members) and its ability to "evolve" a piece of software is not a significant one.

Furthermore, this research is fully funded by a consortium that brought together two governmental agencies; one Brazilian (CAPES[28]) and one American (Fulbright[29]). As modern organizations, a rationalist one could state that they (should) act towards the accomplishment of their own goals, whatever those are. Accordingly, we look at this research as an effort towards that direction, which could benefit them, our sponsors, and, consequently, other organizations with goals similar to theirs, thereby appreciating insights on "going-open" strategies and open source software communities dynamics to use in their own future endeavors.

---

[28] www.capes.gov.br
[29] www.fulbright.org

CHAPTER 4 – DATA DESCRIPTION AND STATISTICAL RESULTS

For clarity and ease of following, we are going to present the statistical analyses in two blocks. In the first, we developed two SEM models using continuous and categorical (dummy) variables, testing propositions 1 to 8.5 on three different samples (2006, 2007 and 2008). In the second block, two other SEM models were developed, utilizing only continuous variables and testing propositions 9 to 12 in a variety of subsamples for two reasons: (a) to evaluate the model robustness over time across samples; and (b) to test the plausibility of the moderator *task complexity* to act as suggested. Chapter 4 is dedicated to the presentation of the steps taken in this theory-testing process as well as the statistical results obtained.

Next, we discuss the form all variables used in this study took, plus their descriptive statistics. That section is followed by a description of how the first two models were assembled and the results they generated, providing statistical tests for 24 propositions over three different samples. The third section describes the approach used for testing the moderation effects predicted in 4 propositions and presents the statistical results acquired.

Continuous Variables' Composition

Our data were similar to Crowston & Scozzi's (2002), requiring all continuous variables to be transformed due to high-levels of Kurtosis and Skewness (see Table 7 for descriptive statistics). These continuous variables are

attractiveness (page_views, downloads, members), *activeness*, *efficiency*, *likelihood of task completion*, *time for task completion* and *project life-span*. All of them but *likelihood of task completion* were log-transformed[30]. *Likelihood of task completion* was transformed into its inverse-sine-square-root[31], which makes more normal distributions from variables in a form of proportion[32], as in the result of efficiency divided by activeness[33] (Crowston & Scozzi, 2002).

The effects of these transformations on Skewness and Kurtosis' levels can be seen in Table 7. But for illustration, the variable *downloads (logdownloads)* had Skewness of 45.499 (0.234), 46.217 (0.264) and 42.052 (0.348), and Kurtosis of 2265.368 (0.359), 2565.711 (0.403) and 2226.074 (0.473) in 2006, 2007 and 2008, respectively. As Skewness' values move away from the range of -1 and +1, indication of deviation from the normal distribution increases. For Kurtosis, as its values depart from zero, the less normal a distribution is said to be (Hair et al., 2006). Based on that, we can see that *downloads* became a normally distributed variable as it was log-transformed, a pattern observed in all other variables.

---

[30] Natural log (Ln).

[31] $p' = \arcsin\sqrt{p}$

[32] For guidelines on data transformation, go to: http://www.zoology.ubc.ca/~whitlock/bio300/LectureNotes/Transformations/Transformations.html

[33] For emphasis: likelihood of task completion is the result of efficiency divided by activeness. More specifically, it is the number of artifacts a project was able to solve (e.g., fixing a bug) divided by the total number of artifacts a project generated.

Categorical Variables' Composition

As for the categorical variables such as *type of project* (topic), we found during data collection that a project could be associated with many different *topics*, *licenses*, *audiences* and *development statuses* at the same time. For example, the database had projects listed with as many as seven *development statuses* at once, all the way from *planning* to *inactive*. A similar pattern was observed with all other categories used in our propositions. Aside from the discussion of whether this makes logical sense from any perspective, we had to find a way to capture these effects statistically.

To do so, we transformed each categorical construct[34] (e.g., *license type*) into a set of dummy variables in a way that a project could score 1 in as many dummy variables as listed-characteristics it had. In the next section, we present how we developed dummy variables for each categorical construct and explain how one dummy variable captures the effect of its categorical construct (see Table 5 for information on this representation).

[34] A categorical construct is empirically studied through the use of many variables, each representing the construct partially. Sometimes constructs of this type are said to be measured *formatively* as opposed to *reflectively*, when measures are consequences of the theoretical constructs they represent (Coltman, Devinney, Midgley, & Venaik, 2008).

Table 5 – Categorical Constructs and their Dummy Variables

Categorical Construct	Formative Dummy Variables
License Type	No_restriction Mod_restriction Both_restrictions Dual_License
Intended Audience	End_users Developers System_administrators Others_audience Advanced_end_users
Project Type (topic)	Communications Database Desktop Education Games Internet Multimedia Office Other Printing Religion Scientific Security Sociology Software System Terminals Text_editor
Project Development Status	Planning Pre_alpha Alpha Beta Production Mature Inactive

Categorical Construct: License Type

Due to the immense number of licenses in use by OS projects, any empirical test is virtually forced to group them into more general categories to be operational. We captured the effects of license type using 4 dummy variables we created in a process described in Appendix B. (Table 6 has information on frequencies and licenses' classification).

Table 6 – Classification of Licenses.

License Name	Restrictive	H-Restrictive	Parent	Code	Sample Frequency			Dummy Coding
					2006	2007	2008	
GNU General Public License (GPL)	Y	Y	14	15	3059	3270	3565	Both_Restrictions
Sub-Total					3059	3270	3565	
Apple Public Source License	Y	N	14	306	2	6	8	Mod_Restriction
Common Public License	Y	N	14	307	63	88	103	Mod_Restriction
Eiffel Forum License	Y	N	14	319	2	3	5	Mod_Restriction
GNU Library or Lesser General Public	Y	N	14	16	747	845	938	Mod_Restriction
IBM Public License	Y	N	14	191	7	11	11	Mod_Restriction
Jabber Open Source License	Y	N	14	300	1	3	4	Mod_Restriction
Motosoto License	Y	N	14	321	0	0	0	Mod_Restriction
Mozilla Public License 1.0 (MPL)	Y	N	14	304	32	31	33	Mod_Restriction
Mozilla Public License 1.1 (MPL 1.1)	Y	N	14	305	84	91	105	Mod_Restriction
Nethack General Public License	Y	N	14	303	5	6	5	Mod_Restriction
Nokia Open Source License	Y	N	14	301	2	2	3	Mod_Restriction
Qt Public License (QPL)	Y	N	14	190	18	18	19	Mod_Restriction
Ricoh Source Code Public License	Y	N	14	193	2	2	2	Mod_Restriction
Sleepycat License	Y	N	14	302	2	3	3	Mod_Restriction
Sun Public License	Y	N	14	318	6	13	17	Mod_Restriction
Sub-Total					973	1122	1256	
Apache Software License	N	N	14	296	105	116	125	No_restriction
Artistic License	N	N	14	17	102	112	116	No_restriction
BSD License	N	N	14	187	451	532	604	No_restriction
Intel Open Source License	N	N	14	299	3	6	9	No_restriction
MIT License	N	N	14	188	128	155	196	No_restriction
Open Group Test Suite License	N	N	14	316	1	2	2	No_restriction
Public Domain	N	N	13	197	124	228	300	No_restriction
Python License (CNRI Python License)	N	N	14	194	18	19	19	No_restriction
Python Software Foundation License	N	N	14	189	12	16	20	No_restriction
Sun Industry Standards Source Licens	N	N	14	298	5	5	5	No_restriction
University of Illinois/NCSA Open Sou	N	N	14	323	6	7	6	No_restriction
Vovida Software License 1.0	N	N	14	297	1	0	0	No_restriction
W3C License	N	N	14	320	2	5	6	No_restriction
X.Net License	N	N	14	317	1	1	1	No_restriction
zlib/libpng License	N	N	14	195	21	28	33	No_restriction
Zope Public License	N	N	14	322	1	1	3	No_restriction
Sub-Total					981	1233	1445	

Table 6 (continued) – Classification of Licenses.

License Name	Restrictive	H-Restrictive	Parent	Code	Sample Frequency			Dummy Coding
					2006	2007	2008	
Academic Free License (AFL)			14	324	18	89	120	N/A
Adaptive Public License			14	628	4	16	17	N/A
Apache License V2.0			14	401	40	86	143	N/A
Attribution Assurance License			14	325	2	2	1	N/A
Common Development and Distribution			14	630	5	15	26	N/A
Common Public Attribution License 1.0 (CPAL)*			14	639	0	0	2	N/A
Computer Associates Trusted Open Sou			14	631	1	5	7	N/A
CUA Office Public License Version 1.			14	402	0	0	1	N/A
Eclipse Public License			14	406	10	31	44	N/A
Educational Community License			14	629	2	11	17	N/A
Eiffel Forum License V2.0			14	392	1	2	3	N/A
Entessa Public License			14	397	0	3	3	N/A
EU DataGrid Software License			14	403	0	1	2	N/A
Fair License			14	404	0	9	14	N/A
Frameworx Open License			14	400	0	2	2	N/A
Historical Permission Notice and Dis			14	393	1	2	2	N/A
Lucent Public License (Plan9)			14	398	0	0	1	N/A
Lucent Public License Version 1.02			14	405	0	2	3	N/A
MITRE Collaborative Virtual Workspac**	Y	Y/N	14	192	0	0	0	N/A
NASA Open Source Agreement			14	407	1	1	2	N/A
OCLC Research Public License 2.0			14	390	1	1	2	N/A
Open Software License			14	388	21	62	84	N/A
OSI-Approved Open Source			13	14	45	42	42	N/A
Other/Proprietary License	?	?	13	196	150	170	195	N/A
PHP License			14	399	16	41	59	N/A
RealNetworks Public Source License V			14	395	0	0	1	N/A
Reciprocal Public License			14	396	1	1	1	N/A
Sybase Open Watcom Public License			14	389	0	0	0	N/A
wxWindows Library Licence			14	391	8	16	18	N/A
Sub-Total					327	610	812	

OBS.: A project may be counted as many times as the number of licenses it has.

Nevertheless, one may read this table as how many projects have a specific license attached to it.

* License present only in the 2008 database.

Restrictive: Y implies that the source code from modifications to the program must be made available.

Highly Restrictive: Y implies that the program cannot be compiled with proprietary programs.

** Licensees can choose between the two possible options.

*** N/A: License not classified in Lerner & Tirole (2005) or with dubious classification.

Source: Adapted from Lerner & Tirole (2005).

Categorical Constructs: Intended Audience, Project Type and Development Status

The remaining categorical constructs were coded in a consistent form. As shown in Table 5, each construct was measured through a number of dummy variables, each representing one characteristic a project might possess, scoring 1, or not possess, and then scoring 0. *Intended audience* required 5 dummies, *project type* required 18[35], and *development status* 7 (totaling 30 dummy variables – plus 4 for *license type*). In the descriptive statistics section that follows, we present their frequencies (see Table 8).

Block 1: Descriptive Statistics and Statistical Results – Propositions 1 to 8.5

Sourceforge.net database contains information on 149,542 OS projects in 2006, 179,867 in 2007, and 143,591 in 2008. However, many of these projects are inactive, have only 1 (one) member, and possess other characteristics that distance them from any software project we may find in more interesting settings. Therefore, we applied a filter to select a more appropriate sample to run our tests on.

We restricted our working sample to projects that (1) had more than one member, (2) had received at least one visit, (3) at least one downloads, (4) at

---

[35] This number was previously said to be 19 (Table 3), but one of the *topics* available to open source projects to choose – *formats* – had to be excluded from our sample. That was necessary because no constant is allowed to enter an SEM-model in EQS and 41 projects had *formats* listed in 2006, but none of them had it in 2007 and 2008. Then, *formats*, a constant with only zeros in 2007 and 2008, was completely excluded from the statistical analysis.

least one artifact (e.g., a bug report – *activeness*), and (5) at least one closed artifact (e.g., a bug fix – *efficiency*). Additionally, we excluded projects with untrustworthy information such as (6) negative – or zero – *time to complete tasks*, (7) negative – or zero – registered time, (8) negative *likelihood of task completion* (i.e., *efficiency* divided by *activeness*), and (9) negative – or zero – *average time for task completion*. As a result of this filter, we ended up working with a sample of 4,769 OS projects in 2006, 4,611 in 2007 and 4,661 in 2008.

For illustration purposes, the average project in the 2008 sample received 3,123 website visits, was downloaded 224,252 times, and has 6 members. Also, this project generated over 143 artifacts (*activeness*) and closed (*efficiency*) roughly 109 of them in an *average* of approximately 130 days. Moreover, the average project is 2,109 days old (over 5.5 years).

Nevertheless, these descriptive statistics have to be interpreted with caution for at least two reasons. First, as we have mentioned, the data is highly skewed and kurtotic with many variables' standard deviations greater than their own means. For example, webpage *visits* has a standard deviation of 32,168, *downloads* has one of 2,836,568, *members* has 9, *activeness* has 1,843, *efficiency* 1,781, and *average time of task completion* of 176 days. Second, it is important to bear in mind that we have presented the descriptive statistics of the variables in their natural form and not in the form we actually used in the structural equation models, which have all continuous variables in their log-transformed form. But since to discuss the logs of webpage visits or the logs of members in a project would not make intuitive sense, we opted to present them

here as we have (see Table 7 for descriptive statistics on the continuous variables of all samples in both forms).

For the categorical constructs, we report the number of projects that scored 1 (one) in each dummy variable in Table 8. To clarify, one may think of the dummy variable as a group separator. The number of projects that score 1 (one) in a dummy variable represents the size of the group that the dummy variable represents. For example, in our 2008 sample, 1,327 OS projects had *licenses* attached to it that would not impose any *restriction* on the software use or modification; 2,721 projects targeted *end-users* as an *audience*; 228 projects were listed under the *topic database*; And 2,009 projects had their software in the *beta development stage* (for a complete list see Table 8).

Table 7 – Descriptive Statistics for Model 1 Testing (without moderator).

Descriptive Statistics												
Sample Sizes			Mean			Std. Deviation			Skewness			Kurtosis		
2006	2007	2008							Statistic			Statistic		
4769	4611	4661												
Variables			**2006**	**2007**	**2008**	**2006**	**2007**	**2008**	**2006**	**2007**	**2008**	**2006**	**2007**	**2008**
pageviews			5097.486	2858.363	3123.592	89313.100	24500.981	32168.194	54.208	25.340	29.595	3325.833	817.214	1086.181
logpageviews			5.084	4.991	4.967	2.413	2.326	2.322	0.190	0.154	0.160	0.056	0.023	0.077
downloads			167612.232	168797.169	224252.312	2921545.550	2301616.385	2836568.737	45.499	46.217	42.052	2265.368	2565.711	2226.074
logdownloads			8.876	9.102	9.324	2.198	2.129	2.108	0.234	0.264	0.348	0.359	0.403	0.473
members			6.359	6.280	6.375	9.247	9.489	9.680	12.295	14.133	15.017	326.330	409.309	471.574
logmembers			1.481	1.471	1.480	0.757	0.750	0.758	0.988	1.015	1.014	0.784	0.907	0.846
activeness			133.313	133.920	143.223	1635.576	1739.353	1843.776	61.271	60.027	62.685	4031.743	3868.334	4147.710
logactiveness			3.046	3.024	3.117	1.712	1.711	1.713	0.369	0.371	0.360	0.022	0.023	0.023
efficiency			99.308	105.882	109.350	1535.000	1681.616	1781.454	63.373	61.676	64.351	4229.496	4023.721	4300.308
logefficiency			2.424	2.447	2.477	1.795	1.806	1.824	0.592	0.582	0.577	-0.033	-0.042	-0.087
likelihood			0.619	0.643	0.612	0.267	0.270	0.268	-0.243	-0.336	-0.237	-0.930	-0.925	-0.949
loglikelihood			0.755	0.797	0.742	0.430	0.444	0.426	0.564	0.434	0.563	-0.538	-0.801	-0.494
average			8605733.143	9947864.963	11237244.165	11352682.288	13214125.768	15221059.687	3.699	3.552	3.422	23.869	20.394	17.665
logaverage			15.067	15.192	15.283	1.879	1.901	1.928	-2.274	-2.217	-2.171	8.682	8.222	7.934
life_span			1399.301	1749.564	2109.233	494.993	498.623	499.876	-0.074	-0.053	-0.055	-0.984	-0.995	-0.993
log_life			7.169	7.422	7.624	0.408	0.309	0.251	-0.770	-0.547	-0.447	-0.092	-0.550	-0.707

Table 8 – Frequency Table – Categorical (dummy) variables.

Categorical Construct		2006	2007	2008
	Sample Size	4769	4611	4661
	Variables' Names	**Frequencies**		
License Type	no_restriction	947	1143	1327
	mod_restriction	942	1077	1197
	both_restrictions	3059	3270	3565
	dual_license	620	609	889
Intended Audience	end_users	2445	2548	2721
	Developers	2933	3022	3241
	system_administrators	1133	1202	1274
	others_audience	488	568	633
	advanced_end_users	155	444	616
Project Type (Topic)	communications	773	181	220
	Database	392	198	228
	Desktop	235	126	139
	Education	171	148	180
	Games	606	219	254
	Internet	1184	241	274
	Multimedia	714	38	50
	Office	328	137	149
	Other	138	149	171
	Printing	39	43	45
	Religion	21	22	22
	Scientific	567	117	130
	Security	156	126	146
	Sociology	28	15	17
	Software	1158	572	604
	System	923	45	50
	Terminals	40	11	14
	text_editor	215	51	57
Life-cycle Stage	Planning	243	638	983
	pre_alpha	318	544	692
	Alpha	818	985	1117
	Beta	1811	1898	2009
	Production	2251	2299	2462
	Mature	208	227	243
	Inactive	76	111	154

Attractiveness' Reliability and Model Building

The SEM model assembled to test propositions 1 to 8.5 contained 42 variables, organized in 40 factors (or constructs). *Attractiveness* is the first construct, "measured" through *webpage visits*, *downloads*, and *members*. The next 5 (five) constructs represent *activeness*, *efficiency*, *likelihood of task completion, time for task completion*, and *project life-span*. Each of these 5 constructs was measured through one single indicator, as **Error! Reference ource not found.** indicates, and together with *attractiveness* completes the list of continuous variables. The categorical constructs were represented by dummy variables, each with a single indicator also. Thus, as the only construct measured through more than one indicator (observed variable)[36], *attractiveness*' measurement reliability should be assessed before proceeding into model assembling.

Attractiveness' Reliability

To assess *attractiveness'* internal consistency across all our samples, we calculated Cronbach's alpha. Combining the 3 empirical measures proposed (*pageviews*, *downloads* and *members*), *attractiveness* scored 0.705 in 2006,

[36] Due to this characteristic of the model, its resemblance to a regression-type of model is no coincidence. We actually regressed attractiveness as the average of website visits, downloads and members (dependent variable) on the dummy variables (independent variables) and observed that the coefficients virtually match between the SEM and multiple regression results.

0.712 in 2007, and 0.714 in 2008, just above the standard acceptable level of Cronbach's alpha (Hair et al., 2006; Rutner, Hardgrave, & McKnight, 2008).

These results indicate that the number of *pageviews*, *downloads* and *members* are likely to have at least one common cause among them, or are empirical expressions of a same latent construct, *attractiveness*, as we have theorized. Also, this conclusion appears to be consistent over time as Cronbach's alpha scores virtually match in all samples (Table 9).

Model Building

The model developed in this dissertation revolves around *attractiveness*, aiming at the explanation of its causes and consequences. To accomplish a test of this model, 5 (five) of the equations coded in EQS are of special interest. The first equation takes care of what was hypothesized to impact *attractiveness*. All dummy variables were theorized to influence *attractiveness* (propositions 5.1, 6.1, 7.1, and 8.1). Thus, we included all categorical constructs, each measured by one dummy variable with error variance set to zero, as predictors of *attractiveness* plus *project life-span*, which is a control variable. In regression terms, we have *attractiveness*, the dependent variable, measured through three indicators (logpageviews, logdownloads and logmembers), being predicted by 34 dummy variables plus *project life-span*, forming the independent variables. The next 4 (four) equations represent what was hypothesized to influence *activeness*,

efficiency, *likelihood of task completion*, and *time for task completion* (propositions 1, 2, 3, and 4; and equations 2, 3, 4, and 5, respectively).

Similarly to *attractiveness*, each of these constructs was predicted to be influenced by all categorical constructs (propositions 5.2, 5.3, 5.4, 5.5, 6.2, 6.3, 6.4, 6.5, 7.2, 7.3, 7.4, 7.5, 8.2, 8.3, 8.4, 8.5). Moreover, *attractiveness*, our main construct, and *project life-span* (a control variable) had their effects on these constructs theorized as well and had their effects evaluated (propositions 5.1, 6.1, 7.1, and 8.1).

To accomplish these tests, we modeled each dummy variable as one categorical construct with error variance set to 0 (zero), plus *attractiveness* with its three indicators, plus the control variable *project life-span* as predictors of *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion* (see Table 10 for details on all equations).

Additionally, three pieces of information on the SEM model assembled are relevant. First, all covariances between the independent factors were estimated. Second, as we have discussed previously, the dependent variables *activeness*, *efficiency* and *likelihood of task completion* are related to one another for (a) *efficiency* is a subgroup of *activeness* (i.e., *activeness* contains *efficiency*'s data plus the number of non-closed artifacts); and (b) *likelihood of task completion* is the result of *efficiency* divided by *activeness*. Thus, to account for those conditions, we assumed the disturbance terms (residuals) of the equations predicting *activeness*, *efficiency*, and *likelihood of task completion* to be correlated and estimated their covariances. According to Cole, Ciesla & Steiger

(2007), failure to account for these disturbance terms' covariances, when theory (logic) says they should correlate, generates misleading results[37]. Third, as discussed in the methods section (Chapter 3), we adopted the Maximum Likelihood (ML) estimation method due to its robustness with large and non-normal samples (Hair et al., 2006).

All 2006, 2007 and 2008's data were entered into EQS in the form of raw data, having their covariance matrices calculated by EQS (Appendix C) in a multi-sample fashion. This multi-sample SEM approach with ML is appropriate for at least two reasons. First, it permits model fit indices to be calculated taking into account all samples at once, increasing our confidence in the model robustness over time as it provides re-tests of the model. In a non-multi-sample fashion, one would have to run the estimation procedure as many times as samples one had for further comparison. Second, the equations coefficients may be calculated both (1) independently, sample-by-sample, or (2) forced to be equal across samples. Later, these two approaches' results can be compared for assessment of model robustness or invariance over time, defeating sampling fluctuations that may obscure effects (Maitland, Dixon, Hultsch, & Hertzog, 2001).

When one opts for the second option, taking advantage of more information available to develop the equations, one needs to impose constraints on the estimation process such that EQS is informed to find "*the best"* equations' coefficients (path coefficients) across all samples. Without the constraints, EQS

[37] To evaluate the impact of not including these disturbance terms' correlations, we omitted them and ran the analysis. The results changed unrealistically to equations' R-squared greater than 0.93, with an overall SEM model with poor model-to-data fit (e.g., RMSEA greater than 0.08).

would estimate equations' coefficients for each sample separately. Of course, the choice of having or not having the constraints affects model fit indices. For that reason, we opted to run the estimation procedure twice, with and without the path-coefficient equality constraints across samples, for comparability[38].

Results – Comparison between Models and Model-to-Data Fit Indices

As previously explained, we coded and ran two different, but nested[39], models to compare their results and choose one for further analysis. One model does not have any constraint and so estimates all path-coefficients between factors (constructs) freely and independently for each of the three samples. The other model has *each and every* path-coefficient between factors constrained to be equal across samples. For example, to constrain the path-coefficients between *attractiveness* and *activeness* across samples, we coded in EQS: "(1,F2,F1)=(2,F2,F1)=(3,F2,F1)", where 1, 2 and 3 stand for each sample (2006, 2007, and 2008), F1 for *attractiveness* and F2 for *activeness*. For emphasis, all 358 path-coefficients between constructs were set to be equal across samples in the constrained model.

According to Rigdon (1996), to compare nested models, one has to compute the chi-square difference between them and compare that value against

---

[38] The actual EQS code is not included in this document because it is 38 pages long. However, it is available upon request to the authors.

[39] Given that the Model with constraints is a special case of the model without the constraints (Rigdon, 1996).

a chi-square critical value, which can be obtained from a regular chi-square table given a level of significance (α=0.05). The null hypothesis in such test is that there is no significant difference in model-fit between the models. Put differently, there is no gain in model-fit to give up this set of constraints. Therefore, when there is no evidence for null-hypothesis rejection, the model *with* the constraints should be kept, as it has more degrees of freedom and facilitates interpretation in our case. This procedure is called *LaGrange-Multiplier Test* for dropping constraints.

The difference in chi-square between the models is 206 (df=358; p-value>0.9). Thus, as the null hypothesis is not rejected, the test result indicates that the model *with the equality constraints* produces at least as good fit as the non-constrained model. Additionally, the two models have similar model-to-data fit indices, matching up to the second decimal place (with the exception of RMSEA, which favors the constrained model by 0.01). Therefore, we opt for the constrained model for further analysis (see Table 9 for details on this test).

Fit indices check whether the pattern of variances and covariances are consistent between the specified model and the sample data (Dow, Jackson, Wong, & Leitch, 2008). A “good” fit is a necessary condition to analyze SEM models. Normally, to evaluate fit, researchers report chi-square values, which they do not expect to be significant (<0.05), root mean square error approximation (RMSEA), which they expect to be smaller than 0.05, and comparative fit index (CFI), which they expect to be greater than 0.90, among others (Dow et al., 2008; Hair et al., 2006). The constrained model has an

RMSEA of 0.016, a CFI of 0.982 and a *significant* chi-square of 2736.912 (df=592; p-value<0.01) – see Table 9.

The chi-square statistic is known for being sensitive to the sample size used and the number of parameters modeled (Cheung, Leung, & Au, 2006; R. Ping, 2008). Accordingly, it should be expected to find a significant chi-square when a model is compared against a sample size of over 4,500 observations with a large number of parameters being estimated. Because of that, researchers are less concerned with chi-square results, accepting a model as of "good" fit when at least one index suggests so (Cheung et al., 2006; R. Ping, 2008). Among them, "RMSEA is relatively most stable among the commonly used fit indices" (Yuan, 2005, p. 141); and the preferable one for theory testing (Cheung et al., 2006; R. Ping, 2008). Therefore, although our model has a significant chi-square, we concluded that it has an acceptable fit based on its CFI greater than 0.90 and RMSEA smaller than 0.05 (Table 9). In the next section, we move to a discussion of each equation, testing propositions 1 to 8.5.

Table 9 – Model-to-Data Fit Indices (without moderator).

	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

Results – Testing Propositions 1 to 8.5

The path loadings (equations' coefficients) for the constrained SEM model are shown in Table 10. Significant paths ($p<0.05$) are highlighted on their T-values. First, to present these results, the effects of the control variable – *project life-span* – on *attractiveness*, *activeness, efficiency*, *likelihood of task completion* and *time for task completion* is discussed. Next, the effects of *attractiveness* on *activeness, efficiency*, *likelihood of task completion* and *time for task completion* are presented. Finally, a section on the influences of *license*, *intended audience*, *project topic* and *development status* on *attractiveness*, *activeness, efficiency*, *likelihood of task completion* and *time for task completion* appears. Due to the large amount of variables' relationships tested in this dissertation, we restrict our analysis to the discussion of the statistically significant ones. The interested reader should refer to Table 10 for additional details.

Project Life-Span (Control variable)

Although the impact of *project life-span* was not formally stated in a propositional form, its effect on the dependent variables was predicted and therefore calculated. As it turned out, *project life-span* is a statistically significant predictor of *attractiveness*, *likelihood of task completion* and *time for task completion*. It has a positive impact on these variables so that one would expect them to increase as a project grows older.

In practical terms, according to the empirical findings, as a project grows older: (a) the more website visits, downloads, and members (higher

attractiveness) it is expected to have, (b) the more *likely* it becomes to solve its issues (e.g., fixing bugs), and (c) the longer it is expected to take to solve its issues. Therefore, the importance of including *project life-span* on an analysis of software development projects appears justified, as Crowston & Scozzi (2002) have posited.

Attractiveness' Impacts (Propositions 1, 2, 3, and 4)

Attractiveness is a statistically significant predictor of all dependent variables under investigation. It positively influences *activeness*, *efficiency*, and *time for task completion* (and negatively *likelihood of task completion*). This means that as a project's *attractiveness* increases, its level of content generated (i.e., *activeness*) as well as its number of actually solved issues (i.e., *efficiency*) increase along. Breaking *attractiveness* down, one may say that as a project receives more *visits*, *downloads* and has more *members*, more content ("problems" and "solutions") it generates, just as Raymond (1999) predicted and many others attempted to test (Raja & Tretter, 2006; Stewart & Gosain, 2006).

However, the impact of *attractiveness* on *likelihood of task completion* and on *time for task completion* is not as intuitive as on *activeness* and *efficiency*. The empirical findings suggest that as a project's number of visits, downloads, and members increase, its *likelihood to complete tasks* decreases and its *time to complete tasks* increases. Thus, it appears, as more visits and downloads occur and more members join an OS project, the less problem-solving-oriented it becomes and the slower it gets to complete its tasks. This pattern unfolds a

potentially negative side of an apparently only-positive project trait (*attractiveness*).

Nevertheless, propositions 1 to 4 do not predict the directionality (positive or negative) of *attractiveness*' impacts on the dependent variables, but simply that it would be a significant predictor of them. Therefore, we fail to reject these propositions and find them all tenable (Table 11).

License-Type's Impacts (Propositions 5.1 to 5.5)

Our analysis of project's licenses impact introduces the logical pattern we are going to use to judge whether a categorical construct, which groups many dummy variables together, can be deemed to be a significant influencer of the dependent variables. As one categorical construct (e.g., license) has many dummy variables (e.g., no_restriction), each representing a *unique* and *independent* facet of their construct, all we need is to check whether at least one of the construct's dummies is significant for finding a statistically significant categorical construct. Alternatively, when none of a construct's dummy variables are significant predictors of a dependent variable, the construct they represent isn't as well. This logic can be easily followed if one imagines an attempt to argue that a categorical construct has no impact on a dependent variable when there is one dummy variable representing it that does.

For *project's license*, 4 (four) dummies were used and one was found significant in predicting all dependent variables but *time for task completion*. On

predicting *attractiveness*, *dual_licensing* was the significant one; *mod_restriction* and *both_restrictions* were found to be positively (and *no_restriction* negatively) associated with *attractiveness*, however insignificantly.

As previously stated, to license a project's software under more than one license with different constraints, depending on the type of user, is an identified trend on software that was once proprietary and had its source code opened later on (Santos Jr., 2008; Watson, Boudreau, York, Greiner, & Wynn Jr., 2008). According to the empirical results, projects under this type of licensing scheme tend to have higher *attractiveness* to the public. In another words, in our sample, to dual-license a project impacts positively its numbers of webpage visits, downloads, and members. Nevertheless, it is noteworthy that these dual-licensed projects are likely to have members who are employees of the company that originally owned their software, providing them an advantage over the others.

As for *activeness*, *efficiency* and *likelihood of task completion*, only *both_restrictions* was found to have a statistically significant impact on them. The impact of *both_restrictions* is negative on *activeness* and *efficiency*, informing that a project tends to produce less content when a modification to the software must be made available and the program cannot be compiled with proprietary programs (GPL license). Additionally, having a project under a license that has *both_restrictions* influences *likelihood of task completion* positively. Thus, projects under the GPL license are more likely to solve its issues than the others.

Finally, none of the 4 dummy variables that captured the impact of license was found statistically significant on explaining *time for task completion*.

Therefore, we have support for propositions 5.1, 5.2, 5.3, and 5.4, but not 5.5, which is then rejected (Table 11).

Intended Audience Impacts (Propositions 6.1 to 6.5)

The impacts of a project's *intended audience* on our model's dependent variables can be summarized as follows. Three of *audience*'s dummies are statistically significant influencers of *attractiveness*. Projects that target *end-users* and *developers* tend to have higher *attractiveness*, whereas projects aiming at *others* tend to have lower. Moreover, the impact of *audience* on *activeness* and *efficiency* is restricted to *end-users*, pushing their levels down. In its turn, *likelihood of task completion* is positively influenced when projects have software aimed at *advanced-end-users*. Finally, a project's average *time for task completion* is significantly affected, negatively, when at least one of a project's *audiences* is *others*; and positively, when the *audience developer* is of target. In conclusion, as at least one of *audience*'s dummy variables was significant in predicting each one of the five dependent variables, there is no empirical reason to reject propositions 6.1 to 6.5 (Table 11).

Type of Project's Impacts (Propositions 7.1 to 7.5)

According to the empirical results, a few *types of projects* stand out as far as *attractiveness* is concerned. On the positive side, increasing a project's *attractiveness*, we have *multimedia*, *printing*, *security*, and *system*; whereas on the negative side there are *database*, *education*, *other*, *scientific*, and *sociology*, hindering a project's *attractiveness*.

Moreover, *project types* influence project's *activeness* and *efficiency* levels along similar lines – aside from *desktop*, which affects negatively *activeness* but not at all *efficiency*. On the *types of project* that are common to affect both: *education*, *office*, and *sociology* influence *activeness* and *efficiency* positively; and *games*, *multimedia*, and *system* do so negatively.

As far as the *likelihood of task completion* is concerned, projects focused on the *communications*, *sociology*, *system*, and *text-editor* niches tend to complete their tasks less often; whereas those focused on *education* and *internet* tend to do so more often.

Finally, the *types of projects* that tend to affect a project's work-pace, increasing the *average amount of time to complete tasks* spent by them, are *other*, *religion*, *sociology*, and *terminals*. No *type of project* was found to influence *average time for task completion* negatively and significantly, reducing a project's average *time to complete tasks*.

Therefore, as *types of project* were found to be statistically significant on predicting all dependent variables – *activeness*, *efficiency*, *likelihood of task*

completion, and *time for task completion*, we have no ground for the rejection of propositions 7.1 to 7.5 (Table 11).

Development Status' Impacts (Propositions 8.1 to 8.5)

The *development status* of a project's software was found to significantly influence a project's *attractiveness* in all of its seven possibilities. Apparently, the initial phases of a project's life-cycle (*planning*, *pre-alpha*, and *alpha*) tend to affect a project's *attractiveness* negatively, scaring people away from the project. Towards the more advanced phases (*beta*, *production*, and *mature*), this pattern is reversed as a project's *attractiveness* tend to increase when in such *statuses*. Reversing the influence pattern again, our findings indicate that people usually is not attracted towards projects listed as *inactive*.

For a project's level of content generated, *activeness* and *efficiency* are positively influenced by projects' classified as *beta*, *production*, and *mature*. Also, the *inactive* status was found to affect a project's *activeness* positively. Moreover, when in the stage of *planning*, a project's *likelihood of task completion* is affected negatively; whereas it is influenced positively when in *pre-alpha*, *production*, and *mature*.

Finally, projects with software listed as in *pre-alpha*, *alpha* and *inactive* life-cycle stages tend to work faster towards the completion of their tasks; whereas those with applications in *production* and *mature* stages tend to be at a

slower work-pace. These results are consistent with all propositions, 8.1 to 8.5, leading us to a fail-to-reject decision on all of them (Table 11).

As we come to the end of the independent variables' influences presentation, it is important to discuss how powerful they are, when used together to predict or explain the dependent variables under investigation.

Table 10 – Equations to test propositions 1 to 8.5.

	Dependent Variables									
	F1 - Attractiveness		**F2-Activeness**		**F3-Efficiency**		**F4-Likelihood of Task Completion**		**F5-Time for Task Completion**	
	2006, 2007, and 2008		*2006, 2007, and 2008*		*2006, 2007, and 2008*		*2006, 2007, and 2008*		*2006, 2007, and 2008*	
Independent Variables										
	Coef.	*T-Statistic*	*Coef.*	*T-Statistic*	*Coef.*	*T-Statistic*	*Coef.*	*T-Statistic*	*Coef.*	*T-Statistic*
F1-Attractiveness*	--	--	0.617	74.872[A]	0.593	67.194[A]	-0.035	-15.249[A]	0.250	25.246[A]
F2-Activeness*	--	--	--	--	--	--	--	--	--	--
F3-Efficiency*	--	--	--	--	--	--	--	--	--	--
F4-Likelihood of Task Completion*	--	--	--	--	--	--	--	--	--	--
F5-Time for Task Completion*	--	--	--	--	--	--	--	--	--	--
F6-Life-Span*	1.019	20.495[A]	0.015	0.402	0.06	1.484	0.031	2.624[A]	0.735	14.941[A]
F7-License(No-Restriction)	-0.077	-1.235	0.018	0.406	0.046	0.927	0.021	1.42	0.034	0.555
F8-License(Mod-Restriction)	0.048	0.991	-0.042	-1.207	-0.024	-0.615	0.02	1.8	0.003	0.059
F9-License(Both-Restrictions)	0.084	1.605	-0.144	-3.827[A]	-0.097	-2.344[A]	0.028	2.299[A]	-0.038	-0.755
F10-License(Dual-Licensing)	0.169	2.522[A]	-0.009	-0.177	-0.05	-0.925	-0.028	-1.78	-0.057	-0.868
F11-Audience(End-Users)	0.549	15.605[A]	-0.052	-2.014[A]	-0.082	-2.903[A]	-0.009	-1.055	-0.031	-0.889
F12-Audience(Developers)	0.16	4.349[A]	0.001	0.042	0.019	0.653	0.01	1.194	0.131	3.643[A]
F13-Audience(System-Admins)	-0.044	-1.141	0.017	0.61	-0.003	-0.099	-0.015	-1.72	0.007	0.189
F14-Audience(Others)	-0.106	-2.171[A]	0.044	1.249	0.044	1.145	0.009	0.833	-0.165	-3.469[A]
F15-Audience(Advanced-End-Users)	0.034	0.583	-0.036	-0.889	0.012	0.254	0.054	4.103[A]	-0.045	-0.813
F16-Type of Project(Communications)	0.004	0.063	0.036	0.849	-0.004	-0.076	-0.037	-2.731[A]	-0.002	-0.026
F17-Type of Project(Database)	-0.333	-4.99[A]	0.079	1.626	0.064	1.198	-0.013	-0.829	0.029	0.439
F18-Type of Project(Desktop)	0.1	1.18	-0.134	-2.177[A]	-0.116	-1.711	-0.012	-0.631	-0.031	-0.369
F19-Type of Project(Education)	-0.348	-4.092[A]	0.228	3.706[A]	0.307	4.53[A]	0.045	2.269[A]	0.154	1.852
F20-Type of Project(Games)	0.041	0.682	-0.16	-3.629[A]	-0.153	-3.163[A]	0.013	0.938	-0.035	-0.592
F21-Type of Project(Internet)	0.071	1.416	-0.034	-0.92	-0.007	-0.173	0.032	2.722[A]	-0.004	-0.08
F22-Type of Project(Multimedia)	0.363	5.194[A]	-0.254	-4.87[A]	-0.298	-5.242[A]	-0.029	-1.78	0.121	1.747
F23-Type of Project(Office)	0.074	0.968	0.254	4.536[A]	0.231	3.76[A]	-0.034	-1.877	0.079	1.049
F24-Type of Project(Other)	-0.535	-6.039[A]	0.039	0.61	0.05	0.709	0.001	0.031	0.238	2.75[A]
F25-Type of Project(Printing)	0.54	3.259[A]	-0.221	-1.851	-0.177	-1.348	0.035	0.924	-0.136	-0.843
F26-Type of Project(Religion)	0.157	0.682	0.295	1.771	0.329	1.794	0.048	0.891	0.736	3.265[A]
F27-Type of Project(Scientific)	-0.227	-3.341[A]	0.046	0.912	0.095	1.732	-0.001	-0.048	0.104	1.549
F28-Type of Project(Security)	0.384	4.153[A]	-0.07	-1.038	-0.032	-0.428	0.004	0.188	0.002	0.02
F29-Type of Project(Sociology)	-0.769	-3.226[A]	0.55	3.168[A]	0.512	2.684[A]	-0.11	-1.986[A]	0.521	2.23[A]
F30-Type of Project(Software-Dev)	0.055	1.2	0.013	0.379	0.015	0.405	-0.008	-0.783	-0.042	-0.939
F31-Type of Project(System)	0.211	3.282[A]	-0.263	-5.491[A]	-0.324	-6.198[A]	-0.036	-2.342[A]	-0.006	-0.096
F32-Type of Project(Terminals)	-0.1	-0.44	-0.199	-1.196	-0.231	-1.265	-0.008	-0.151	0.495	2.215[A]
F33-Type of Project(Text-Editors)	0.174	1.686	0.053	0.692	-0.087	-1.043	-0.069	-2.83[A]	0.039	0.384
F34-Life-Cycle(Planning)	-0.21	-4.247[A]	0.026	0.749	-0.002	-0.048	-0.027	-2.389[A]	-0.008	-0.165
F35-Life-Cycle(Pre-Alpha)	-0.506	-9.41[A]	0.038	0.985	0.08	1.87	0.03	2.458[A]	-0.263	-5.023[A]
F36-Life-Cycle(Alpha)	-0.146	-3.233[A]	-0.026	-0.785	-0.017	-0.471	0.006	0.618	-0.15	-3.407[A]
F37-Life-Cycle(Beta)	0.169	4.365[A]	0.126	4.513[A]	0.163	5.297[A]	0.003	0.348	-0.034	-0.908
F38-Life-Cycle(Production)	0.948	23.811[A]	0.129	4.431[A]	0.239	7.423[A]	0.047	5.038[A]	0.183	4.632[A]
F39-Life-Cycle(Mature)	1.097	14.477[A]	0.112	2.028[A]	0.206	3.404[A]	0.067	3.783[A]	0.187	2.508[A]
F40-Life-Cycle(Inactive)	-0.397	-3.834[A]	0.17	2.286[A]	0.137	1.676	-0.019	-0.818	-0.301	-2.986[A]
R-Squared	**2006; 2007; 2008**		**2006; 2007; 2008**		**2006; 2007; 2008**		**2006; 2007; 2008**		**2006; 2007; 2008**	
	0.218; 0.174; 0.158		0.45; 0.469; 0.476		0.394; 0.406; 0.402		0.029; 0.025; 0.029		0.131; 0.112; 0.104	

*** : Variable log-transformed.**

[A] : Significant at 0.05 level; T-value > 1.96.

Independent Variables' Prediction Power

The model built to test propositions 1 to 8.5 has 5 equations that interest us the most. These equations intend to explain *attractiveness*, *efficiency*, *likelihood of task completion*, and *average time for task completion* based on projects' characteristics and *life-span*. Also, our tests successfully attempted to use *attractiveness* to predict *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*.

In explaining *attractiveness*, a project's characteristics (license type, intended audience, type of project, and life-cycle stage) plus project life-span were capable of accounting for 21.8% (2006), 17.4% (2007), and 15.8% (2008) of *attractiveness*' variance. These same independent variables plus *attractiveness* were used to predict *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*, achieving different degrees of predictability.

Activeness' explained variance was of 45% in 2006, 46.9% in 2007, and 47.6% in 2008. Also, the independent variables were capable of explaining 39.4% of *efficiency's* variance in 2006, 40.6% in 2007, and 40.2% in 2008. In its turn, *likelihood of task completion*'s equation was capable of explaining 2.9%, 2.5%, and 2.9% of its variance in 2006, 2007, and 2008, respectively. Finally, towards the prediction of *average time for task completion*, *attractiveness*, *project life-span* and *project's characteristics* were able to explain 13.1% of its variance

in 2006, 11.2% in 2007, and 10.4% in 2008 (see Table 10 for a complete list of R-squared values).

Table 11 – Decisions: Empirical Results on Propositions 1 to 8.5.

Propositions	Decision	Variable: Direction
P1: A project's attractiveness is a significant predictor of a project's *activeness*.	Not rejected	Positive
P2: A project's attractiveness is a significant predictor of a project's *efficiency*.	Not rejected	Positive
P3: A project's attractiveness is a significant predictor of a project's *overall likelihood of task completion*.	Not rejected	Negative
P4: A project's attractiveness is a significant predictor of a project's *average time for task completion*.	Not rejected	Positive
P5.1: A project's license type is a significant influencer of its attractiveness.	Not rejected	Dual_license: Positive
P5.2: A project's license type is a significant influencer of its activeness.	Not rejected	Both_restrictions: Negative
P5.3: A project's license type is a significant influencer of its effectiveness.	Not rejected	Both_restrictions: Negative
P5.4: A project's license type is a significant influencer of its overall likelihood of task completion.	Not rejected	Both_restrictions: Positive
P5.5: A project's license is a significant influencer of its average time to complete tasks.	Rejected	No effect
P6.1: A projects' intended audience significantly influences its attractiveness.	Not rejected	End-users: Positive Developers: Positive Others: Negative
P6.2: A projects' intended audience significantly influences its activeness.	Not rejected	End-users: Negative
P6.3: A projects' intended audience significantly influences its efficiency.	Not rejected	End-users: Negative
P6.4: A projects' intended audience significantly influences its overall likelihood of task completion.	Not rejected	Advanced-end-users: Positive
P6.5: A projects' intended audience significantly influences its average time for task completion.	Not rejected	Developers: Positive Others: Negative
P7.1: A project's topic is a significant influencer of a project's attractiveness.	Not rejected	Database: Negative Education: Negative Other: Negative Scientific: Negative Sociology: Negative Multimedia: Positive Printing: Positive Security: Positive System: Positive
P7.2: A project's topic is a significant influencer of a project's activeness.	Not rejected	Desktop: Negative Games: Negative Multimedia: Negative System: Negative Education: Positive Office: Positive Sociology: Positive
P7.3: A project's topic is a significant influencer of a project's efficiency.	Not rejected	Games: Negative Multimedia: Negative System: Negative Education: Positive Office: Positive Sociology: Positive

Table 11 (continued) – Decisions: Empirical Results on Propositions 1 to 8.5.

Propositions	Decision	Variable: Direction
P7.4: A project's topic is a significant influencer of a project's overall likelihood of task completion.	Not rejected	Communications: Negative Sociology: Negative System: Negative Text-Editor: Negative Education: Positive Internet: Positive
P7.5: A project's topic is a significant influencer of a project's average time for task completion.	Not rejected	Other: Positive Religion: Positive Sociology: Positive Terminals: Positive
P8.1: A project's development status is a significant influencer of a project's attractiveness.	Not rejected	Planning: Negative Pre-alpha: Negative Alpha: Negative Beta: Positive Production: Positive Mature: Positive Inactive: Negative
P8.2: A project's development status is a significant influencer of a project's activeness.	Not rejected	Beta: Positive Production: Positive Mature: Positive Inactive: Positive
P8.3: A project's development status is a significant influencer of a project's efficiency.	Not rejected	Beta: Positive Production: Positive Mature: Positive
P8.4: A project's development status is a significant influencer of a project's overall likelihood of task completion.	Not rejected	Planning: Negative Pre-alpha: Positive Production: Positive Mature: Positive
P8.5: A project's development status is a significant influencer of a project's average time for task completion.	Not rejected	Pre-alpha: Negative Alpha: Negative Production: Positive Mature: Positive Inactive: Negative

Block 2: Testing the Effect of Task Complexity as a Moderator

Projects' Classification and Working-Sample Characteristics

To perform empirical tests derived off propositions 9 to 12, we utilized a slightly different filter to select our working-sample. In addition to the previous filter (see page 71); only projects without *missing data* on the variables we used to calculate *task complexity* were included in the sample. Although some might

question this procedure of avoiding missing data in the sample on grounds of artificiality, we believe there are stronger reasons favorable to its adoption.

First, one needs to realize that we are interested in separating projects based on their "complexity" for further evaluation of whether the strength of the connection between constructs change depending on that same "projects' complexity". Therefore, if a project's complexity cannot be calculated due to missing data, one simply cannot classify it as more or less complex without relying on chance or doubtful procedures.

Unfortunately, we could not classify projects according to their complexity as we intended and described in Table 4 (Chapter 3). Our initial intentions were to classify projects' level of complexity along a continuum, from simple to very complex, with various categories within those extremes. However, the calculation of a project tasks complexity[40] index resulted in an extremely skewed distribution, with the vast majority of the projects scoring 1 (one) on that index.

To exemplify, from the 149,542 projects in the non-filtered 2006 sample, only 48 projects scored higher than or equal to 2 on *task complexity* (7 projects scored higher than or equal to 3). This scenario forced us into developing a new strategy to separate projects based on their complexity, focusing on the achievement of acceptable group sizes for SEM utilization. Accordingly, instead of working with many groups, as originally intended, we classified projects in two

[40] The number of other modules a specific module exchanges information with, creating interdependency.

groups (separated by *task complexity*'s median): (i) those that scored 1, and (ii) those that scored more than 1.

This separation method gave us 6 (six) groups to work with; as the 2006, 2007, and 2008 samples were broken down into two groups each, of "low" and "high" complexity projects, so to speak. These groups have sizes of 1,216 (2006-low complexity), 296 (2006-high complexity), 979 (2007-low complexity), 214 (2007-high complexity), 978 (2008-low complexity) and 209 (2008-high complexity). These numbers are shown in Table 12.

Model Building and EQS Coding

In assembling the model for testing the moderation effects specified in propositions 9 to 12, we focused exclusively on the constructs of interest. These propositions posit that the relationships between *attractiveness* and (1) *activeness*, (2) *efficiency*, (3) *likelihood of task completion*, and (4) *time for task completion* are moderated by *task complexity*. Accordingly, we did not include any of the dummy variables nor the control variable in this model. That left us with 7 variables (3 for *attractiveness*; 1 for *activeness*; 1 for *efficiency*; 1 for *likelihood of task completion*; 1 for *time for task completion*). The univariate statistics of these variables are in Table 12, where it is possible to see that no variable has Skewness or Kurtosis outside the range of +/- 1.

Each one of the six groups' covariance matrix was calculated by EQS based on the raw data inputted to the software. These six distinct covariance

matrices are available in Appendix D. Similarly to the first model we built to test propositions 1 to 8.5, we utilized Maximum Likelihood (ML) as the estimation method, the variables were entered in their log-transformed form, and the disturbance terms from the equations were allowed to correlate for the same reasons explained previously.

As for the equations coded for analysis in EQS, four of them are of interest for testing the moderation-propositions. In each one of them, *attractiveness* is the independent construct, predicting *activeness* in the first equation, *efficiency* in the second, *likelihood of task completion* in the third, and *time for task completion* in the fourth (dependent variables).

As explained, the effect of *task complexity* was captured by the separation of the projects in groups of low and high complexity. So, in operational terms, our task was to test whether *attractiveness*' coefficients, developed to predict each one of those four dependent variables, are statistically different from each other across all six samples (groups) created to separate low- from high-complexity projects. This approach compares to a between-group difference detection using an ANOVA-type of analysis.

Nevertheless, this is not the only approach to test for moderation using SEM. Besides the multi-sample analysis we adopted, at least one alternative approach, referred to as product-term regression, is known. To perform a test using product-term regression, one has to create an additional variable, which is the result of the independent variable times the moderator, and regress it on the dependent variable, together with the original independent variable. The

statistical significance of this calculated variable works as a test for the moderation.

Although some previous research have reported that multi-sample analysis (our approach) performs slightly worse than product-term regression (R. A. Ping, 1996) – detecting spurious interactions 8% of the time (against 3%); more recent findings in the information systems area suggest that the covariance-based SEM multi-sample analysis works just as well as any other approach when large sample sizes with normal data are used (Qureshi & Compeau, 2009). Additionally, no technique works satisfactorily under conditions of high Skewness and Kurtosis (Qureshi & Compeau, 2009). Accordingly, our chosen approach seems to be a reasonable one in face of the variables and samples in place[41].

Attractiveness' Reliability

As we worked with different samples to test for the moderation role of project's *task complexity*, it is necessary to re-evaluate *attractiveness'* internal consistency. Accordingly, we calculated *attractiveness'* Cronbach's alpha for each one of the six subsamples we ended up working with.

All subsamples reach the threshold of 0.7 normally adopted in the literature. The subsamples of projects classified as of low complexity achieved

[41] To double check our assumption, we ran the moderation tests using the regression-approach. The conclusions were the same.

0.711, 0.721, and 0.723 in 2006, 2007, and 2008, respectively. The high-complexity projects' subsamples have Cronbach's alphas of 0.69 (2006), 0.696 (2007), and 0.703 (2008). Although two of the high-complexity subsamples are slightly below the 0.7 threshold, the difference is only of 0.01 and 0.004 and may be justified by their smaller sample-sizes. Therefore, we consider *attractiveness'* internal consistency satisfactory and proceed to the models results presentation.

Table 12 – Descriptive Statistics: Variables for Moderation Testing.

Descriptives Statistics							
		2006-LC	**2006-HC**	**2007-LC**	**2007-HC**	**2008-LC**	**2008-HC**
	Sample Size	1216	294	979	214	978	209
logpageviews	Mean	5.003	5.246	4.889	5.081	4.870	4.994
	Std. Deviation	2.521	2.718	2.404	2.593	2.398	2.577
	Skewness	0.276	0.134	0.281	0.216	0.264	0.252
	Kurtosis	0.072	-0.467	0.116	-0.171	0.100	-0.095
logdownloads	Mean	8.968	8.960	9.106	9.107	9.309	9.183
	Std. Deviation	2.261	2.483	2.172	2.519	2.138	2.506
	Skewness	0.192	0.155	0.219	0.324	0.303	0.416
	Kurtosis	0.275	-0.209	0.417	0.183	0.487	0.253
logmembers	Mean	1.671	1.873	1.650	1.828	1.646	1.810
	Std. Deviation	0.793	0.857	0.791	0.829	0.801	0.830
	Skewness	0.822	0.491	0.780	0.506	0.757	0.631
	Kurtosis	0.827	-0.273	0.475	-0.281	0.263	-0.058
logactiveness	Mean	3.361	3.729	3.208	3.687	3.306	3.712
	Std. Deviation	1.807	1.820	1.813	1.855	1.834	1.798
	Skewness	0.322	0.150	0.413	0.185	0.372	0.210
	Kurtosis	0.008	-0.488	0.175	-0.455	0.119	-0.505
logefficiency	Mean	2.786	3.221	2.699	3.207	2.734	3.152
	Std. Deviation	1.928	1.947	1.916	2.004	1.942	1.967
	Skewness	0.449	0.202	0.568	0.231	0.550	0.240
	Kurtosis	-0.193	-0.576	0.099	-0.608	0.010	-0.614
loglikelihood	Mean	0.771	0.788	0.833	0.838	0.772	0.760
	Std. Deviation	0.409	0.365	0.428	0.408	0.411	0.386
	Skewness	0.499	0.347	0.348	0.314	0.531	0.452
	Kurtosis	-0.437	-0.298	-0.819	-0.715	-0.468	-0.406
logaverage	Mean	15.075	14.967	15.122	15.023	15.194	15.069
	Std. Deviation	1.866	1.798	1.926	1.752	1.926	1.805
	Skewness	-2.326	-2.521	-2.075	-1.860	-2.067	-1.763
	Kurtosis	9.109	10.571	6.998	6.416	7.173	5.835
OBS.: LC=Low Complexity; HC=High Complexity.							

Model Comparison Procedure – An Overall Test of Task Complexity Moderation

In a fashion similar to the one done to test propositions 1 to 8.5, we developed two SEM models, with and without constraints that forced EQS to find equal *attractiveness*' coefficients in predicting each dependent variable across all

six subsamples[42]. As Table 13 shows, the constrained-model has a chi-square of 285 (d.f.=68; p-value <0.01), whereas the unconstrained-model has a chi-square of 279 (d.f.=48; p-value<0.01). Both have good model-to-data fit indices with a CFI of 0.99 for the constrained- and 0.989 for the unconstrained-model.

A comparison between the models leads to the conclusion that the constrained model is preferable to the unconstrained model for at least two reasons. First, the chi-square difference between them (5.778) is smaller than the critical chi-square value of 31.41 (d.f.=20; α=0.05), suggesting that no significant difference exists in terms of model-to-data on estimating the coefficients independently for each subsample. Second, the most reliable fit index, RMSEA, indicates that the constrained-model is a better fit, scoring 0.029 (against 0.035 of the unconstrained one).

Although this model comparison would be sufficient evidence that *task complexity* does not work as a moderator of the relationships between *attractiveness* and *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*, we decided to test each proposition separately to increase our confidence in the results. The next section presents a statistical test for each plus a description of the equations resulting from the preferred constrained-model.

[42] The actual constraints and their relationship to each proposition can be seen in Table 15.

Equations' Results and Decisions on Propositions 9 to 12

The results presented in Table 14 are consistent with the ones we achieved on predicting *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion* using *attractiveness* on the larger samples (Table 10). *Attractiveness* is a statistically significant predictor of these dependent variables influencing: (a) *activeness*, positively and being capable of explaining from 49.3% to 55.4% of its variance; (b) *efficiency*, positively and explaining from 44.2% to 49.4% of its variance; (c) *likelihood of task completion*, negatively and explaining from 0.9% to 1.1% of its variance; and (d) *time for task completion*, positively and explaining from 11% to 13% of its variance (Table 14).

To test propositions 9 to 12 and the moderation roles of *task complexity*, we imposed a total of 20 constraints to this SEM model. These 20 constraints may be separated in 4 groups, each representing one proposition, of 5 constraints. For example, to evaluate whether the relationship between *attractiveness* and *activeness* was moderated by *task complexity*, we coded in EQS: "**1**: (1,F2,F1)-(2,F2,F1)=0; **2**:(1,F2,F1)-(3,F2,F1)=0; **3**:(1,F2,F1)-(4,F2,F1)=0; **4**:(1,F2,F1)-(5,F2,F1)=0; **5**:(1,F2,F1)-(6,F2,F1)=0". The first constraint establishes that the coefficient (path) associated with *attractiveness* to predict *activeness* in group 1 (2006: low-complexity) and group 2 (2006: high-complexity) are equal. The second one specifies that the path between *attractiveness* and *activeness* are also equal across samples 1 (2006: low-complexity) and 3 (2007: low-complexity). The list of constraints goes on to cover

each and every path between factors equal across groups (see Table 15 for a complete list).

With all constraints set, we ran the Lagrange-Multiplier test to evaluate whether the model would gain in model-fit (decrease in chi-square) if each one of those 20 constraints were dropped, one-by-one. In other words, this test informs whether each path-equality constraint harms the estimation method, which would be then better off estimating *attractiveness*' coefficients independently for each subsample (Dow et al., 2008) – providing support for moderation.

The hypothesis-testing procedure has a general form. Each constraint represents a null-hypothesis, which states that two paths are equal. If the Lagrange-Multiplier test returns a p-value smaller than 0.05, we would have evidence to reject the null-hypothesis, thus providing support for the moderation. However, we were unable to reject any of the null-hypothesis created by the constraints, leading us to the same conclusion that *task complexity*, as collected and calculated, does not moderate the relationships between *attractiveness* and (i) *activeness*, (ii) *efficiency*, (iii) *likelihood of task completion* and (iv) *time for task completion*. Therefore, we reject propositions 9, 10, 11 and 12 (Table 15)[43].

[43] Besides the univariate tests presented, we also ran the multivariate version of the Lagrange-Multiplier test to evaluate whether there was any specific path – not necessarily in the order we entered – that could be moderated by *task complexity*, increasing even more our confidence in the results. The multivariate results are consistent with the univariate ones as there is no evidence for rejection of any of the null-hypothesis as well (see Appendix E for these results).

Table 13 – Model Comparison for Task Complexity Moderation Testing.

	Model with Equality Constraints						Model without Equality Constraints						Comparison
	2006-Low	2006-High	2007-Low	2007-High	2008-Low	2008-High	2006-Low	2006-High	2007-Low	2007-High	2008-Low	2008-High	
Sample Sizes	1216	294	979	214	978	209	1216	294	979	214	978	209	
Attractiveness Cronbach's Alpha	0.711	0.690	0.721	0.696	0.723	0.703	0.711	0.69	0.721	0.696	0.723	0.703	
Chi-Square	285.586 (68 Degrees of Freedom)						279.808 (48 Degrees of Freedom)						5.778 (20 d.f.)
P-Value for Chi-Square	<0.01						<0.01						Same
Model Fit (CFI)	0.990						0.989						0.001
B-B Normed Fit Index	0.987						0.987						0.000
Root Mean Square Residual (RMR)	0.042						0.074						-0.032
RMSEA	0.029						0.035						-0.006

Chi-Square Critical Value (0.05; 20 d.f.):	31.410
Decision (given 5.778 < 31.410):	Favor Model with Constraints

Table 14 – Equations: *Attractiveness* to predict the Dependent Variables.

	Dependent Variables							
	F2-Activeness*		F3-Efficiency*		F4-Likelihood of Task Completion*		F5-Time for Task Completion*	
Samples:	*2006-L; 2006-H; 2007-L; 2007-H; 2008-L; 2008-H*		*2006-L; 2006-H; 2007-L; 2007-H; 2008-L; 2008-H*		*2006-L; 2006-H; 2007-L; 2007-H; 2008-L; 2008-H*		*2006-L; 2006-H; 2007-L; 2007-H; 2008-L; 2008-H*	
Independent Variable	*Coef.*	*T-Statistic*	*Coef.*	*T-Statistic*	*Coef.*	*T-Statistic*	*Coef.*	*T-Statistic*
F1-Attractiveness*	0.681	46.195[A]	0.684	43.201[A]	-0.02	-5.49[A]	0.326	19.869[A]
R-Squared	0.554; 0.523; 0.543; 0.493; 0.537; 0.528		0.494; 0.473; 0.489; 0.429; 0.481; 0.442		0.009; 0.011; 0.008; 0.009; 0.009; 0.01		0.119; 0.124; 0.11; 0.13; 0.11; 0.125	

[A] : Signicant at α=0.05; T-value>1.96.

* : Variable Log-Transformed.

In summary, 23 out of 28 propositions developed in this dissertation were not rejected, giving the model a success rate of roughly 82%. Additionally, the SEM models developed obtained good model-to-data fit, indicating that the theory developed around *attractiveness* with the indicators we have chosen and collected might be a fruitful one. Next, we present our final chapter, discussing in-depth the reasons we believe our results turned out as they did and how this theory and its empirical results might inform the academic and the managerial communities.

Table 15 – Decisions on Moderation-Propositions 9 to 12.

Propositions	Imposed Constraints (#: Hypothesis)	Lagrange-Multiplier For Releasing Constraints (P-Values)	Decision
P9: The overall complexity of a project's tasks moderates the relationship between attractiveness and activeness.	1: (1,F2,F1)-(2,F2,F1)=0;	0.78	Reject P9
	2: (1,F2,F1)-(3,F2,F1)=0;	0.853	
	3: (1,F2,F1)-(4,F2,F1)=0;	0.801	
	4: (1,F2,F1)-(5,F2,F1)=0;	0.733	
	5: (1,F2,F1)-(6,F2,F1)=0;	0.513	
P10: The overall complexity of a project's tasks moderates the relationship between attractiveness and efficiency.	6: (1,F3,F1)-(2,F3,F1)=0;	0.755	Reject P10
	7: (1,F3,F1)-(3,F3,F1)=0;	0.848	
	8: (1,F3,F1)-(4,F3,F1)=0;	0.791	
	9: (1,F3,F1)-(5,F3,F1)=0;	0.647	
	10: (1,F3,F1)-(6,F3,F1)=0;	0.485	
P11: The overall complexity of a project's tasks moderates the relationship between attractiveness and overall likelihood of task completion.	11: (1,F4,F1)-(2,F4,F1)=0;	0.275	Reject P11
	12: (1,F4,F1)-(3,F4,F1)=0;	0.977	
	13: (1,F4,F1)-(4,F4,F1)=0;	0.703	
	14: (1,F4,F1)-(5,F4,F1)=0;	0.457	
	15: (1,F4,F1)-(6,F4,F1)=0;	0.621	
P12: The overall complexity of a project's tasks moderates the relationship between attractiveness and average time of task completion.	16: (1,F5,F1)-(2,F5,F1)=0;	0.944	Reject P12
	17: (1,F5,F1)-(3,F5,F1)=0;	0.932	
	18: (1,F5,F1)-(4,F5,F1)=0;	0.512	
	19: (1,F5,F1)-(5,F5,F1)=0;	0.988	
	20: (1,F5,F1)-(6,F5,F1)=0;	0.504	

CHAPTER 5 – CONCLUSIONS, DISCUSSIONS AND FINAL REMARKS

After an exercise of theory development, this dissertation empirically analyzed open source software projects “from the outside” in an attempt to discover patterns in their internal dynamics. The model and its results, if put in a timeline and summarized, inform us about OS projects on a variety of areas. In the model background, there is a pool of OS projects created as a result of going-open initiatives from organizations and individuals, and there is a population interested in using and contributing to these projects’ software. Thus, our first concern was to explore what drives people to specific projects; or, what types of projects are more attractive to people.

To address this concern, we focused on *characteristics* that are capable of “positioning” projects better (worse) to the eyes of the population browsing for software to fulfill their needs as well as intrinsic and extrinsic motivations, thereby influencing projects’ *attractiveness*. Operationally, we defined *attractiveness* as the numbers of *website visits*, *downloads* and *members* a project has. Additionally, we stated that to be attractive represents only partially OS projects’ goals, as ultimately they pursue not just to be adopted and used by the people, but also improved by them. Therefore, the question of whether *attractiveness* leads to improvements in the software appears automatically. We studied these improvements in the form of *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*, which are all, at least, necessary conditions for improvements in the project (*software quality*) to occur.

We believe our endeavor and its results are timely and useful for public and private organizations as well as individuals as many have increased their involvement in the open source movement over the last few years. Additionally, we believe that to create awareness on how and why these projects operate the way they do helps to perpetuate their creation and long-term existence through organizational intervention, leading to more knowledge available to the population in the form of software source code.

Our study aimed at explaining 5 constructs related to open source projects: *attractiveness*, *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*. For that, 35 variables plus a moderator were utilized, accounting for a variety of projects' characteristics. And besides these 36 variables, attractiveness was also used in the attempt to explain *activeness*, *efficiency*, *likelihood of task completion* and *time for task completion*. Accordingly, our study's findings can be described based on 1) the selection of a repeated cross-sectional SEM model over a simple cross-sectional one and the results of 5 equations, one for each endogenous construct; and 2) a comparison between two SEM models to test the moderation.

The first model comparison focused on empirically evaluating the overall performance of the theoretical model in two distinct situations. The first involved an estimation procedure that was performed in each one of the samples independently. The second dealt with an estimation took all three samples into consideration at once to calculate same set of estimates (betas) to all three samples. This comparison provided a clue on the theoretical model's robustness

as the procedure functions as a re-test of the model across samples at same time that it reduces the chances of sampling errors and fluctuations. If the model performance (fit) had deteriorated over time, an indication against its validity would have been produced. However, the results indicated that the model not only fits the data well, but that it also performs better when more data from various samples spread over time is inputted to develop estimates, reducing the residual values (unexplained part of variables' variance).

Having established the overall model performance as satisfactory, an analysis of propositions 1 to 8.5 followed. Five equations were developed for this analysis. One of them has *attractiveness* as the dependent variable, being explained by project's *license type*, *intended audience*, *type of project*, *life-cycle stage* and *life-span*. All predictors were found to be significantly associated with *attractiveness*, explaining roughly 22% of its variance in 2006. Nevertheless, revisiting some of the results is worthwhile.

For example, projects with software under *licenses* with distinct requirements, successfully accounting for some contingency of their audience, tend to be more *attractive* than others. Also, the results suggest that: 1) projects should avoid having the audience *others* attached to their project, as this may hinder their *attractiveness*; 2) the influence of *type of project* is most negative on their *attractiveness* when they are in the topics of *sociology* and *other*, and positive when listed in *printing* and *security* categories; 3) a pattern exists on the influence of *life-cycle* stage on *attractiveness*. The pattern indicates that a project's 3-first *life-cycle* stages harm *attractiveness*, especially when in pre-

alpha. However, if the project makes it to the fourth stage, this influence is reversed. And of course, when in the *inactive* stage, *attractiveness* is influenced negatively.

The second and third equations tackle the explanation of *activeness* and *efficiency*, which happen to have a very similar pattern when it comes to the explanatory variables found significant (with the exceptions of *desktop* and *inactive*). Among the most interesting results, we found that projects *licensed* under GPL, the most common and restrictive open source license, tend to be less *active* as well as less *efficient* than projects that do not have GPL attached to them. This finding is consistent with previous studies that pointed out that organizations and individuals see GPL requirements negatively, decreasing their intention to contribute to the project (Fershtman & Gandal, 2007; Lerner & 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 unrestrictive licenses, would keep people from contributing (Sauer, 2007). Furthermore, three other comments on the results are worth-making: 1) to target *end-users* affects *activeness* and *efficiency* negatively, but projects listed under topics such as *education*, *sociology* and *office*, which are supposedly aimed at *end-users*, tend to score higher on *activeness* and *efficiency* thus creating a challenge for the interpretation; 2) projects listed as *inactive* have higher scores of *activeness*. This is surprising and not an easy finding to justify, but perhaps it simply means that members-users of the software keep reporting bugs and features after a project is considered *inactive*. Moreover, it is noteworthy that

these bugs and features reported tend to not be closed, as the effect of *inactive* on *efficiency* is not significant, showing that *inactive* software tend to be stagnated as the category name implies; 3) *life-span* is not a significant influencer of *activeness* and *efficiency*, showing that the number bugs reported and features requested do not increase as projects get older and suggesting that software's quality might be indeed increasing with time as less problems are found in the long-term.

The fourth equation was capable of explaining at best 3% of *likelihood of task completion*'s variance. Surprisingly, the results suggest that more *attractive* projects have smaller *likelihood to complete their tasks*, indicating that an overload of tasks might occur as more tasks are requested (i.e., *activeness*) 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* in the sense of features requests and bug reports. Nevertheless, the explained 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.

The fifth and last equation focuses on the explanation of *time for task completion*, accounting for 13% of its variance in 2006. More *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. This pattern of

influences, from *attractiveness* to *activeness* to *time for task completion*, is also supported by the associations of *life-span* with *attractiveness* and *time for task completion*, as they are all positive. Furthermore, projects' *life-cycle* stage 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*, supporting the positive association between *life-span* and *time for task completion*. Moreover, projects seem to rush for task closure when at *inactive*.

The second comparison between two SEM models provided a test for the proposed moderated-by-*task complexity* effects between *attractiveness* and 1) *activeness*, 2) *efficiency*, 3) *likelihood of task completion*, and 4) *time for task completion*. Statistically, no indication that *task complexity* works as a moderator in these relationships was produced. This finding is consonant with the results of an experiment that investigated the moderating effect of task complexity on the relationship between number of people working on the coding and software quality (Balijepally, Mahapatra, Nerur, & Price, 2009). Thus, further evidence that the connection between number of people and software quality (or efficiency) is direct, and not moderated by *task complexity*, was found.

Commonly, members of open source projects are developers *and* users of the community software, which broadens their perspectives on software quality to contain technical, functional and social issues. Moreover, these OS members are in an environment where many are likely 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 independent entities. Accordingly, more members should indeed lead to higher software quality, linearly. No need to worry about attracting free-riders, reducers of the probability of success (Bessen, 2005), as, apparently, the phenomenon of free-riding is not significant in OS projects. Johnson's (2002, p. 644) prediction that "when more individuals are present, the incentive to free-ride is raised", so that contributions become less likely to occur , has no support from our analysis. Thus, the theme of highest value to organizations interested in "going-open" is how to set up and run a project so 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.

Implications for Theory and Practice

Although previous studies have suggested the importance of *attractiveness* to open source software projects (Agerfalk & Fitzgerald, 2008; Stewart & Gosain, 2006), this is the first study to directly model it as a multifaceted latent construct and investigate theoretically and empirically its causes and consequences. And for that uniqueness, we believe this research contributes to theory and practice in many ways.

For theory, the results of this research suggest that variables such as the numbers of page views, downloads and members should not be treated as causes of each other nor as final dependent variables on studies of open source projects. That is because these variables are significantly correlated with each not because they cause each other, as one could interpret, but due to the

existence of a common cause to all of them, *attractiveness*. Moreover, these variables alone, or in combination, cannot improve software or service of open source communities. They are necessary conditions for improvements to occur, but none of them is sufficient. Therefore, to study what the consequences of these variables are, treating them as independent variables (or mediators), is vital to capture a more accurate picture of what success represents to OS projects, their sponsors and members.

Furthermore, this research sheds new light on a dilemma in the Information Systems literature, where one stream claims that quantity of people increases diversity, and that diversity is a good thing for problem-solving ("the more eye balls, the less bugs") and for innovation generation (Raymond, 1999; West & O'Mahony, 2005); and another, that points out recent empirical evidence suggesting that the relationship between the number of members of a software project and its ability to "evolve" a piece of software is not a significant one, not affecting *effectiveness*, *efficiency*, nor quality of these teams and therefore contradicting the other stream (Balijepally et al., 2009).

Without a claim to definitively close the debate, but perhaps conciliating the two sides, this research presents effects and side-effects to open source software projects with increasing *attractiveness* (e.g., number of members). On the one hand, these projects tend *to locate and fix more bugs*, *request and develop more features*, and *to ask and answer more questions* in general, creating an environment where innovations are more likely to occur and that in long-term generates higher-quality software. Nevertheless, these same projects

with increasing *attractiveness* tend to reduce their *likelihood to solve* their increasing *number of bugs reported* and *features requested*. Additionally, projects with higher *attractiveness* generating more *contributions* from members on bugs and features and from users on support requests tend to reduce their *work-pace*, taking more time to solve these issues. Therefore, the answer to the question of whether *number of members* affects *efficiency*, *effectiveness* and software quality in general depends on the definitions of *efficiency*, *effectiveness* and software quality one decides to adopt.

For practice, Agerfalk & Fitzgerald (2008, p. 394) have already established the need for organizations involved in open source software projects “to market the *attractiveness* of the project and improve its visibility”. But left to the organizations to deal with is the question of how they can attempt to maximize their projects’ attractiveness outside of the advertisement realm.

By focusing on the impact of a project’s *characteristics* on *attractiveness*, our study can guide organizations on many practical matters that have to be faced when transforming a closed and proprietary software into an open source project. First, a glimpse on how *attractive* a piece of software would be when released open source could be generated by looking at equation 1, which has *attractiveness* as the dependent variable of a variety software *characteristics*. These characteristics could work as a check-list to organizations and their software.

Moreover, in a more prescriptive form, organizations may use the results of equation 1 to identify among their software the ones more likely to succeed if

“open sourced”, helping to direct and prioritize resources, especially of the less experienced organization, more effectively. Finally, our study may aid organizations on the crucial decision of choosing one or more licenses, and of what kind (more or less restrictive) to register the software. It is known that such choice affects directly people’s decisions to participate in and contribute to a project and of companies to adopt or not adopt an application (Agerfalk & Fitzgerald, 2008). Our results indicate that a contingency approach is preferable when such decision is faced for projects’ attractiveness tends to be higher when they have more than one license attached, thereby fulfilling the diverse motivations of their members and intended audiences more effectively.

Limitations

Consistent with any research, this one does not lack limitations. Research limitations may be divided in two kinds, internal and external. The external limitations relate to the things that could have been included in the study but were not; and the internal ones relate to how the things that were done in the study could have been done differently. Fortunately, cases from both categories can be seen as opportunities for future research as well.

Among the internal limitations of our study, we were able to identify the following. First, on capturing the effects of *license type* on the endogenous variables, we did not classify each and every *license* available to OS projects according to their restrictiveness. We ignored the *licenses* not classified by Lerner & Tirole (2005) and therefore were not able to study their effects on

projects' *attractiveness* and *dynamics*. Ideally, we should have read the content of these *licenses* and classified them, but we considered that outside of the intended scope of this research. Second, although we collected data over different time periods, we analyzed it using the traditional cross-sectional approach, not the longitudinal one. In doing so, we were not able to account for some things known to be beneficial to studies such as controlling for auto-correlations and constructs' change over time.

Amid the external limitations of our study, we thought of a variety of variables that were omitted from our analysis but could have an impact on the endogenous variables investigated, and perhaps even interact with the exogenous variables, thereby changing their relationships with the endogenous ones. This limitation can never go away from any study and represents the main reason why researchers can never, strictly speaking, claim that a causal relationship between two variables *in fact* exists for it is impossible to be sure that nothing else related to the variables is missing.

Specifically, the things we could name that are likely to matter to future studies are: 1) the level of technical knowledge of the community members, as that can directly affect projects' activities and software characteristics; 2) the level of trust between members, possibly affecting their likelihood to share information with each other and therefore to contribute more or less; and 3) the amount of sponsored members in the projects, as the presence of members that are not volunteers and then dedicate their time to the project for monetary motivations may complicate a comparison with completely non-sponsored communities.

Future Research

The best way to address the limitations we identified in this research is with follow-up studies. A wide variety of these studies can be derived from our results and conclusions, covering topics related to both content and method. On the content side, a more complete study of the influence of licenses is encouraged for at least two reasons. First, as we have mentioned, it was not possible to consider all licenses available to OS projects. Second, the phenomenon of dual-licensing, which is expressed by companies such as Trolltech and MySQL offering both commercial and open source options according to user preference, was just recently identified (Watson et al., 2008). And to the best of our knowledge, this is the first study to empirically assess the impacts of this choice. Therefore, replications should be highly beneficial to the field of open source.

Moreover, studies aimed at understanding what exactly the categories utilized in this research (e.g., education and end-users) mean to members, users and the population in general – impacting their choices of which OS software to adopt and likelihood to participate and contribute to the projects – should be encouraged. This stream of research may clarify apparent contradictory findings such as the negative effect of a project being listed as *end-user*, and a positive one when listed as *office*, on its activeness. At first, it seems that software of the *office* kind aims at end-users as well. So, why aren't the effects the same? Are there sub-categories within the end-user category? Or perhaps *office*-software is

not really intended at end-users and has a distinct population of members. Future research can explain these issues based on members' perceptions.

Furthermore, we do not consider the issue of task complexity solved. Although we have not found it to moderate the relationships proposed, that does not mean that it does not have a direct effect on our endogenous variables. As a matter of fact, a direct impact of task complexity on software quality has been proposed and found significant in a recent study (Balijepally et al., 2009). A similar test could be made to assess the direct influence of task complexity on activeness and efficiency, for example. Additionally, we can not rule out the possibility that the projects we analyzed are not complex enough to make the effect identifiable statistically. Much work remains to be done in this area.

As a last topic on the content side, Agerfalk & Fitzgerald (2008) pointed out that the recruitment of members of an open source community by sponsors could erode the "unknown" aspect of the project, which in turn may affect trust levels and innovation rates. This proposition is closely related to the limitation of not accounting for the number of sponsored members in a project we pointed out, and thus reinforces the need to add this variable into future studies.

On the method side, studies adopting a longitudinal approach would add to the robustness of the statistical analysis utilized in this study, presenting a more realistic test of the propositions and demonstrating how the constructs evolve in different types of projects over time. A longitudinal approach could unfold a relationship between activeness and software quality, for example; where activeness peaks when software is at low-quality and decreases as

software quality increases. Additionally, although we could test variables' effects and the direction of them on the endogenous variables, we did not calculate their effect-sizes, taking into account coefficients and equations' R-squares. Accordingly, we do not know how relevant the effect of a specific project trait is on its activities. Future studies may address this topic. Finally, the SEM model developed in this research could be utilized as a baseline model for future studies. As such, tests of competing models against this one might be performed for improvements in how we see the relationships between variables and the achievement of more parsimonious models. All these possibilities remain open.

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.

Baldwin, Y., C., & Clark, K. B. (2003). Does Code Architecture Mitigate Free Riding in the Open Source Development Model?

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*(1), 91-118.

Baum, J. A. C., & Oliver, C. (1991). Institutional Linkages and Organizational Mortality. *Administrative Science Quarterly, 36*(2), 187-218.

Bentler, P. M. (1989). *EQS structural equations program manual*. Los Angeles, CA: BMDP Statistical Software Inc.

Bessen, J. E. (2005). *Open Source Software: Free Provision Of Complex Public Goods*: SSRN: http://ssrn.com/abstract=588763 or DOI: 10.2139/ssrn.588763.

Byrne, B. M. (1994). *Structural equation modeling with EQS and EQS/Window: Basic concepts, applications, and programming*: Sage Publications.

Cheung, M., Leung, K., & 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).

Cole, D. A., Ciesla, J. A., & Steiger, J. H. (2007). The Insidious Effects of Failing to Include Design-Driven Correlated Residuals in Latent-Variable Covariance Structure Analysis. *Psychological methods. 12, no. 4*.

Coltman, T., Devinney, T. M., Midgley, D. F., & Venaik, S. (2008). Formative versus reflective measurement models: Two applications of formative measurement. *Journal of Business Research, 61*(12), 1250-1262.

Counsell, S., & Swift, S. (2006). The interpretation and utility of three cohesion metrics for object-oriented design. *ACM Transactions on Software Engineering and Methodology, 15*(2), 123-149.

Crowston, K., Annabi, H., Howison, J., & Masango, C. (2004). *Effective work practices for software engineering: Free/libre open source software development*. Paper presented at the WISER Workshop on Interdisciplinary Software Engineering Research, SIGSOFT.

Crowston, K., Annabi, H., Howison, J., & Masango, C. (2005). *Towards a portfolio of FLOSS project success measures.* Paper presented at the 26th International Conference on Software Engineering, Edinburgh, UK.

Crowston, K., & Howison, J. (2006). Hierarchy and Centralization in Free and Open Source Software Team Communications. *Knowledge, Technology, and Policy, 18*(4), 65-85.

Crowston, K., & Scozzi, B. (2002). Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development. *IEEE Proceedings — Software Engineering, 149*(1), 3-17.

DeSouza, C. R. B., Redmiles, D., Cheng, L., Millen, D., & Patterson, J. (2004a). *Sometimes You Need to See Through Walls — A Field Study of Application Programming Interfaces*. Paper presented at the CSCW'04, November 6–10.

DeSouza, C. R. B., Redmiles, D., Cheng, L., Millen, D., & Patterson, J. (2004b). *How a Good Software Practice Thwarts Collaboration – The multiple roles of APIs in Software Development*. Paper presented at the SIGSOFT'04/FSE-12, Oct. 31–Nov. 6.

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*(4), 106-114.

Farhoomand, A. (2007). *Opening up of the Software Industry: The Case of SAP.* Paper presented at the Management of eBusiness, 2007. WCMeB 2007. Eighth World Congress on the.

Fershtman, C., & Gandal, N. (2007). Open source software: Motivation and restrictive licensing. *IEEP, 4*, 209–225.

Fitzgerald, B. (2006). The Transformation of Open Source Software. *MIS Quarterly, 30*(3), 587-598.

Fitzgerald, B., & Feller, J. (2002). A further investigation of open source software: community, co-ordination, code quality and security issues. *Information Systems Journal, 12*(1), 3-5.

Hair, J. F., Black, W. C., Babin, B. J., Anderson, R. E., & Tatham, R. L. (2006). *Multivariate Data Analysis* (6th ed. ed.). Upper Saddle River, N.J.: Pearson Education Inc.

Herbsleb, J., & Mockus, A. (2003). An Empirical Study of Speed and Communication in Globally Distributed Software Development. *IEEE Transactions on Software Engineering, 29*(6), 481-494.

Hoyle, R. H. e. (1995). *Structural Equation Modeling, Concepts, Issues, and Applications*. Thousand Oaks, CA.: Sage Publications.

Hunt, F., & Johnson, P. (2002). *On the Pareto Distribution of Sourceforge Projects.* Paper presented at the Proceedings of the Open Source Software Development Workshop Newcastle, UK.

Jiang, Z., & Benbasat, I. (2007). The Effects of Presentation Formats and Task Complexity on Online Consumers' Product Understanding. *MIS Quarterly, 31*(3), 475-500.

Johnson, J. P. (2002). Open Source Software: Private Provision of a Public Good. *Journal of Economics & Management Strategy, 11*(4), 637-662.

Joreskog, K. G., & Sorbom, D. (1993). *Lisrel 8: Structured equation modeling with the Simplis command language*. IL: Scientific Software International, Inc.

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

Kline, R. B. (1998). *Principles and practice of structural equation modeling*. New York: The Guilford Press.

Koch, S. (2004). Profiling an Open Source Project Ecology and Its Programmers. *Electronics Markets. Special Section: Open Source Software, 14*(2).

Koch, S., & Schneider, G. (2002). Effort, co-operation and co-ordination in an open source software project: GNOME. *Information Systems Journal, 12*(1), 27-42.

Krishnamurthy, S. (2002). Cave or Community? An Empirical Examination of 100 Mature Open Source Projects [Electronic Version]. *First Monday (http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/960/881), 7*. Retrieved 02/18/08.

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*.

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. *J Gerontol B Psychol Sci Soc Sci, 56*(2), P69-77.

Mardia, K. V. (1970). Measures of multivariate skewness and kurtosis with applications. *Biometrika, 57*, 519-530.

Mockus, A., Fielding, R. T., & Herbsleb, J. (2000). *A case study of open source software development: the Apache server*. Paper presented at the Proceedings of the 22nd International Conference on Software Engineering.

Mockus, A., & Herbsleb, J. D. (2002). *Why not improve coordination in distributed software development by stealing good ideas from open source?* Paper presented at the ICSE '02 Workshop on Open Source 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*, 139-150.

Ping, R. (2008). Personal Communication by Email. In C. Santos (Ed.).

Ping, R. A. (1996). Improving the Detection of Interactions in Selling and Sales Management Research. *The Journal of personal selling & sales management. 16, no. 1*.

Qureshi, I., & Compeau, D. (2009). Assessing Between-Group Differences in Information Systems Research: A Comparison of Covariance- and Component-Based SEM. *MIS Quarterly, 33*(1), 197-214.

Raja, U., & 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.

Rigdon, E. (1996). Model Comparison in SEM [Electronic Version]. http://www2.gsu.edu/~mkteer/nested.html.

Rutner, P. S., Hardgrave, B. C., & McKnight, D. H. (2008). Emotional Dissonance and the Information Technology Professional. *MIS Quarterly, 32*(3), 635-652.

Santos Jr., C. (2008). Understanding Partnerships between Corporations and the Open Source Community: A Research Gap. *IEEE Software, 25*(6).

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), 605-619.

Shaikh, M., & Cornford, T. (2003). Version management tools: CVS to BK in the Linux kernel [Electronic Version]. *Working Paper*. Retrieved May 6, 2006.

Sharma, S., Sugumaran, V., & Rajagopalan, B. (2002). A framework for creating hybrid-open source software communities. *Information Systems Journal, 12*(1), 7-25.

Stamelos, I., Angelis, L., Oikonomou, A., & Bleris, G. L. (2002). Code quality analysis in open source software development. *Information Systems Journal, 12*(1), 43-60.

Steiger, J. H. (1990). Structural model evaluation and modification: An interval estimation approach. *Multivariate Behavioral Research, 25*, 173-180.

Stewart, K., Ammeter, A., & Maruping, L. (2005). *A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects*. Paper presented at the Proceedings of the 38 Hawaii International Conference on System Sciences.

Stewart, K., & Gosain, S. (2006). The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly, 30*(2), 291-314.

Thomas, D., & Hunt, A. (2004). Open source ecosystems. *IEEE Software, 32*(1), 89-91.

von Hippel, E. (2005). *Democratizing innovation.* Boston, MA.: MIT Press.

von Hippel, E., & von Krogh, G. (2003). open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science, 14*(2).

von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research policy, 32*(7), 26.

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.

Weiss, D., & Poland, P. (2006). Measuring success of open source projects using web search engines.

West, J., & O'Mahony, S. (2005). *Contrasting Community Building in Sponsored and Community Founded Open Source Projects*. Paper presented at the Proceedings of the 38th Annual Hawai International Conference on System Sciences.

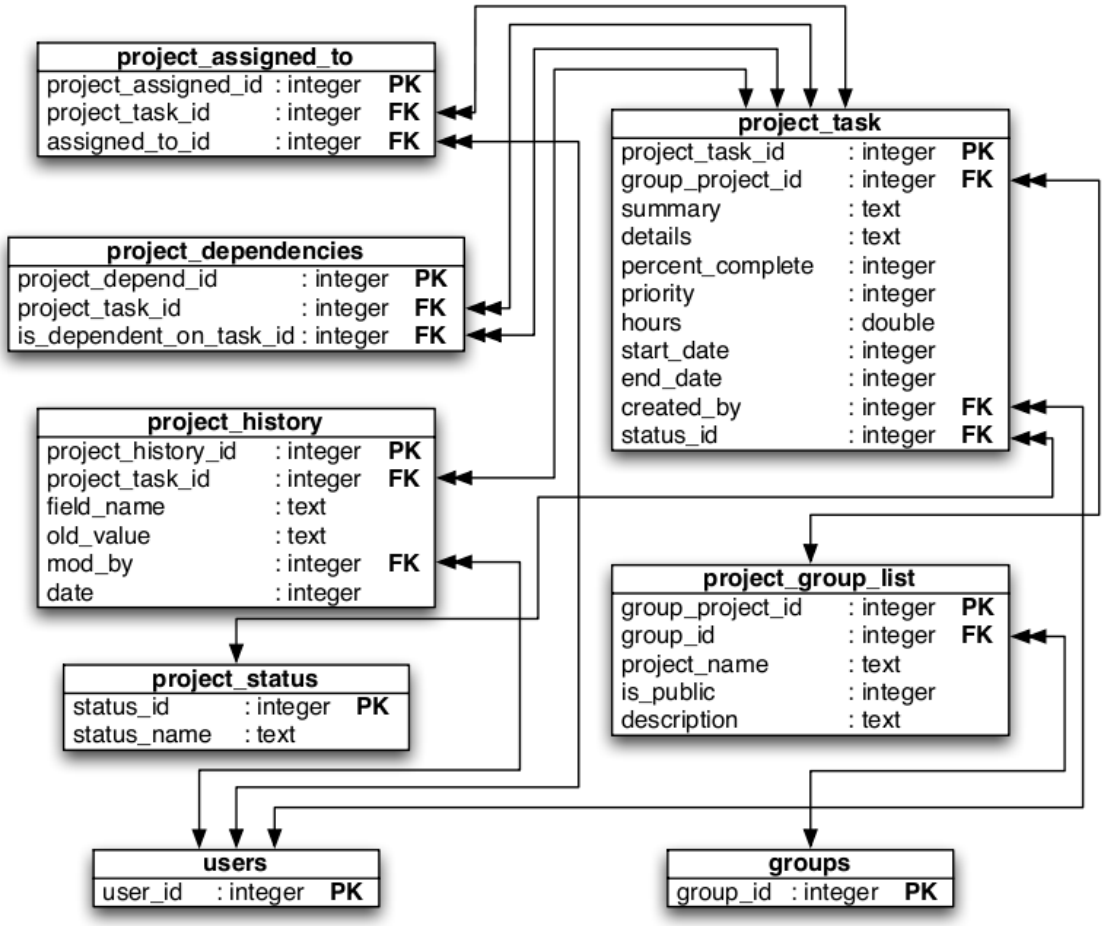
Xu, B., Qian, J., Zhang, X., Wu, Z., & Chen, L. (2005). A brief survey of program slicing. *SIGSOFT Softw. Eng. Notes, 30*(2), 1-36.

Yuan, K.-H. (2005). Fit Indices Versus Test Statistics. *Multivariate Behavioral Research, 40*(1), 115-148.

APPENDICES

APPENDIX A – SOURCEFORGE.NET PARTIAL E-R DIAGRAM

Entity Relationship (ER) Diagram – Sourceforge.net Database at U. Notre Dame



Source: https://zerlot.cse.nd.edu/mywiki/index.php?title=ER_diagrams

APPENDIX B – CAPTURING THE EFFECTS OF LICENSE TYPE

To capture the effects license type through dummy variables, we first classified each license according to three dimensions: highly restrictive, restrictive, and unrestrictive; just as previous studies have done (Fershtman & Gandal, 2007; Lerner & Tirole, 2005).

A *highly restrictive* license implies that the program cannot be compiled with proprietary programs, and a *restrictive* one implies that the source code from modifications to the program *must* be made available. When a license does not restrict the software in any of those two forms, it is said to be *unrestrictive* (Fershtman & Gandal, 2007; Lerner & Tirole, 2005). Additionally, it possible that one license has both restrictions. GPL is *restrictive* and *highly restrictive* in that software must make modifications available and not be compiled into proprietary programs. Finally, it is important to emphasize that there is no license classified *solely* as *highly restrictive*.

Having classified the licenses, we coded each project according to the characteristic(s) of the license(s) attached to it. For that, we created 4 dummy variables, *no_restriction*, *mod_restriction*, *both_restrictions*, and *dual_license*. If at least one of the license(s) attached to a project imposed no restrictions whatsoever, that project scored one in *no_restriction* (and zero, otherwise). If at least one of a project's license(s) was *restrictive*, then such a project scored one in *mod_restriction* (short for restriction of modification). If at least one of a project's license(s) was *restrictive* and *highly restrictive* (GPL), such a project scored one in *both_restrictions*. Furthermore, if a project had scored one in *no_restriction* as well as in *mod_restriction* (or one in *no_restriction* as well as in *both_restrictions*), then such project scored one in *dual_license*[44]. Finally, when a license type was not found in the cases we used, it could not be classified and therefore was left out of our statistical analysis (see Table 6 for information on licenses' classifications).

---

[44] To license software under two distinct licenses is becoming a common practice in going-open strategies. Such practice allows open source projects and software sales to coexist, creating a business based in open source initiatives (Watson, Boudreau, York, Greiner, & Wynn Jr, 2008).

APPENDIX C – COVARIANCE MATRICES (WITHOUT MODERATOR)

Covariance Matrix (without moderator) – 2006 Sample (42 VARIABLES)

	LOGPAGEV V1	LOGDOWNL V2	LOGMEMBE V3	LOGACTIV V4	LOGEFFIC V5
LOGPAGEV V1	5.824				
LOGDOWNL V2	3.831	4.831			
LOGMEMBE V3	.576	.572	.573		
LOGACTIV V4	2.107	2.199	.611	2.929	
LOGEFFIC V5	2.060	2.136	.617	2.887	3.222
LOGLIKEL V6	-.110	-.116	-.017	-.163	.060
LOGAVERA V7	.976	1.082	.207	.930	.955
LOG_LIFE V8	.089	.262	.053	.120	.122
NO_RESTR V9	.000	.012	.020	.022	.022
MOD_REST V10	.001	.021	.017	.010	.012
BOTH_RES V11	.061	.076	-.018	-.002	.000
DUAL_LIC V12	.055	.085	.027	.035	.036
END_USER V13	.119	.193	.012	.073	.062
DEVELOPE V14	.031	.056	.040	.040	.048
SYSTEM_A V15	.037	.047	-.003	.025	.025
OTHERS_A V16	.002	.002	.010	.012	.013
ADVANCED V17	.007	.010	.004	.002	.005
COMMUNIC V18	-.010	.034	.000	.025	.020
DATABASE V19	-.018	-.016	-.002	.006	.009
DESKTOP V20	.018	.035	.005	.009	.010
EDUCATIO V21	-.013	-.023	.002	-.002	.002
GAMES V22	-.011	.000	.013	-.011	-.007
INTERNET V23	.031	.003	-.004	.017	.022
MULTIMED V24	.072	.082	.009	.008	.002
OFFICE V25	.012	.012	.005	.026	.026
OTHER V26	-.011	-.014	-.001	-.005	-.004
PRINTING V27	.008	.008	-.001	.003	.003
RELIGION V28	-.002	-.001	.000	.000	.000
SCIENTIF V29	-.028	-.045	.013	-.016	-.007
SECURITY V30	.013	.015	.000	.000	.000
SOCIOLOG V31	-.004	-.005	.001	.003	.003
SOFTWARE V32	.000	.006	.012	.028	.035
SYSTEM V33	.073	.059	.008	-.002	-.011
TERMINAL V34	.001	.003	-.001	.000	-.001
TEXT_EDI V35	.008	.013	-.002	.011	.006
PLANNING V36	-.027	-.026	.005	-.010	-.012
PRE_ALPH V37	-.066	-.070	.003	-.045	-.044
ALPHA V38	-.090	-.097	-.008	-.072	-.079
BETA V39	-.012	.015	-.004	.015	.014
PRODUCTI V40	.268	.292	.038	.180	.198
MATURE V41	.044	.050	.010	.031	.032
INACTIVE V42	-.018	-.012	-.002	-.005	-.004

	LOGLIKEL V6	LOGAVERA V7	LOG_LIFE V8	NO_RESTR V9	MOD_REST V10
LOGLIKEL V6	.185				
LOGAVERA V7	-.063	3.532			
LOG_LIFE V8	-.004	.167	.167		
NO_RESTR V9	.001	.020	.013	.159	

MOD_REST	V10	.001	.019	.009	-.025	.159
BOTH_RES	V11	.000	-.002	.008	-.088	-.079
DUAL_LIC	V12	.000	.038	.031	.031	.039
END_USER	V13	-.010	.034	.020	-.020	-.034
DEVELOPE	V14	.005	.061	.032	.031	.047
SYSTEM_A	V15	-.001	.015	.015	-.001	-.014
OTHERS_A	V16	.001	-.003	.007	-.002	-.002
ADVANCED	V17	.003	.002	.003	-.001	.001
COMMUNIC	V18	-.006	.005	.004	-.003	-.008
DATABASE	V19	.002	.004	.002	.003	.004
DESKTOP	V20	.000	.009	.007	-.002	.002
EDUCATIO	V21	.002	-.001	-.002	-.003	-.002
GAMES	V22	.004	-.019	.004	-.007	-.008
INTERNET	V23	.006	.002	.001	.006	-.004
MULTIMED	V24	-.005	.036	.009	-.005	.006
OFFICE	V25	-.001	.009	.000	-.002	-.001
OTHER	V26	.001	.004	.001	-.001	-.002
PRINTING	V27	.000	.001	.000	.000	.000
RELIGION	V28	.000	.003	-.001	.000	-.001
SCIENTIF	V29	.004	.004	-.001	.002	.011
SECURITY	V30	-.001	.002	-.002	-.002	.000
SOCIOLOG	V31	.000	.000	.000	.001	-.001
SOFTWARE	V32	.002	.014	.008	.026	.035
SYSTEM	V33	-.007	.019	.010	.002	-.003
TERMINAL	V34	.000	.002	.000	.000	.000
TEXT_EDI	V35	-.003	.009	.001	.002	.002
PLANNING	V36	.000	-.003	.009	.003	.001
PRE_ALPH	V37	.003	-.036	.006	-.001	.003
ALPHA	V38	.000	-.051	-.001	.003	.000
BETA	V39	-.004	-.011	.001	-.005	.004
PRODUCTI	V40	.003	.126	.020	.006	.005
MATURE	V41	.000	.027	.007	.003	.001
INACTIVE	V42	.000	-.009	.000	.000	.000

		BOTH_RES V11	DUAL_LIC V12	END_USER V13	DEVELOPE V14	SYSTEM_A V15
BOTH_RES	V11	.230				
DUAL_LIC	V12	.022	.113			
END_USER	V13	.075	.005	.250		
DEVELOPE	V14	-.057	.022	-.062	.237	
SYSTEM_A	V15	.024	.005	-.010	-.005	.181
OTHERS_A	V16	.011	.006	.004	.002	.005
ADVANCED	V17	.004	.006	.002	.004	.005
COMMUNIC	V18	.016	.002	.022	-.013	.020
DATABASE	V19	-.004	.001	-.008	.011	.009
DESKTOP	V20	.004	.003	.015	-.004	-.005
EDUCATIO	V21	.006	.000	.002	-.005	-.005
GAMES	V22	.016	.001	.031	-.013	-.017
INTERNET	V23	.001	.003	-.010	.008	.041
MULTIMED	V24	.011	.009	.027	.000	-.023
OFFICE	V25	.006	.001	.013	-.007	-.001
OTHER	V26	.003	.001	.006	-.001	-.001
PRINTING	V27	.000	.000	.001	-.001	.000
RELIGION	V28	.001	.000	.000	-.001	.000
SCIENTIF	V29	-.007	.003	-.008	.007	-.021
SECURITY	V30	.002	.000	.001	-.004	.014
SOCIOLOG	V31	.000	.000	.002	.000	-.001

SOFTWARE	V32	-.053	.014	-.071	.087	-.020
SYSTEM	V33	.016	.010	-.006	.000	.057
TERMINAL	V34	.000	.000	.001	-.001	.002
TEXT_EDI	V35	-.003	.001	.001	.006	-.004
PLANNING	V36	.005	.013	.008	.005	.000
PRE_ALPH	V37	.007	.010	.009	.006	-.001
ALPHA	V38	.007	.010	.006	.007	-.008
BETA	V39	.019	.007	.015	.001	.004
PRODUCTI	V40	.000	.006	.000	.011	.022
MATURE	V41	-.002	.004	.000	.005	.003
INACTIVE	V42	.000	-.001	.000	.000	-.001

		OTHERS_A V16	ADVANCED V17	COMMUNIC V18	DATABASE V19	DESKTOP V20
OTHERS_A	V16	.092				
ADVANCED	V17	.000	.031			
COMMUNIC	V18	.004	.003	.136		
DATABASE	V19	.000	.001	-.006	.075	
DESKTOP	V20	-.001	.000	-.004	.000	.047
EDUCATIO	V21	.004	.000	-.003	.000	.000
GAMES	V22	.006	.002	-.012	-.008	-.004
INTERNET	V23	.010	.003	.007	.008	-.006
MULTIMED	V24	.000	.002	-.011	-.006	.001
OFFICE	V25	.004	.000	.000	.004	-.002
OTHER	V26	.004	.000	-.003	.000	.000
PRINTING	V27	.001	.000	-.001	.000	.001
RELIGION	V28	.001	.000	.000	.000	.000
SCIENTIF	V29	.002	.002	-.013	.000	-.001
SECURITY	V30	.002	.001	.002	-.001	-.001
SOCIOLOG	V31	.000	.000	.000	.000	.000
SOFTWARE	V32	-.008	.000	-.025	.004	-.002
SYSTEM	V33	-.003	.003	-.006	-.004	.001
TERMINAL	V34	-.001	.000	.001	.000	.000
TEXT_EDI	V35	.001	.000	-.004	-.002	.001
PLANNING	V36	.005	.004	.002	.000	.001
PRE_ALPH	V37	.003	.003	.002	.001	.001
ALPHA	V38	.002	.002	.000	.001	.001
BETA	V39	.006	.001	.008	-.002	.000
PRODUCTI	V40	.003	.001	.002	.007	.001
MATURE	V41	.003	.002	.000	.000	.000
INACTIVE	V42	-.001	-.001	.001	.000	.000

		EDUCATIO V21	GAMES V22	INTERNET V23	MULTIMED V24	OFFICE V25
EDUCATIO	V21	.035				
GAMES	V22	.001	.111			
INTERNET	V23	-.003	-.018	.187		
MULTIMED	V24	.000	.000	-.013	.127	
OFFICE	V25	.001	-.007	.003	-.006	.064
OTHER	V26	.002	-.001	-.001	-.001	.001
PRINTING	V27	.000	-.001	-.001	.001	.002
RELIGION	V28	.000	-.001	.000	.001	.000
SCIENTIF	V29	.008	-.003	-.016	.006	-.001
SECURITY	V30	-.001	-.004	.002	-.005	-.001
SOCIOLOG	V31	.001	.000	-.001	-.001	.000
SOFTWARE	V32	-.003	-.019	-.012	-.020	-.004
SYSTEM	V33	-.003	-.013	-.006	-.011	-.007

TERMINAL	V34	.000	-.001	.000	-.001	.000
TEXT_EDI	V35	.002	-.004	.000	-.003	.001
PLANNING	V36	.002	.006	.001	.003	.001
PRE_ALPH	V37	.000	.012	.003	.001	.000
ALPHA	V38	.000	.013	-.007	.003	-.001
BETA	V39	.003	-.002	-.001	.004	.000
PRODUCTI	V40	-.002	-.011	.016	-.003	.006
MATURE	V41	.001	.000	.002	.001	-.001
INACTIVE	V42	.000	.000	.000	.000	.000

		OTHER V26	PRINTING V27	RELIGION V28	SCIENTIF V29	SECURITY V30
OTHER	V26	.028				
PRINTING	V27	.000	.008			
RELIGION	V28	.001	.000	.004		
SCIENTIF	V29	-.001	-.001	.000	.105	
SECURITY	V30	.000	.000	.000	-.002	.032
SOCIOLOG	V31	.000	.000	.001	.001	.000
SOFTWARE	V32	-.003	-.001	-.001	-.005	-.005
SYSTEM	V33	-.001	-.001	.000	-.008	.012
TERMINAL	V34	.000	.000	.000	-.001	.001
TEXT_EDI	V35	.001	.001	.000	-.001	-.001
PLANNING	V36	.001	.000	.000	.001	.000
PRE_ALPH	V37	-.001	.000	.000	-.002	.000
ALPHA	V38	.003	.000	.000	.003	-.001
BETA	V39	-.001	.001	.001	.005	.004
PRODUCTI	V40	.001	.000	.000	-.002	.000
MATURE	V41	.000	.000	.000	.002	.000
INACTIVE	V42	.000	.000	.000	.000	.000

		SOCIOLOG V31	SOFTWARE V32	SYSTEM V33	TERMINAL V34	TEXT_EDI V35
SOCIOLOG	V31	.006				
SOFTWARE	V32	-.001	.184			
SYSTEM	V33	-.001	-.008	.156		
TERMINAL	V34	.000	-.001	.001	.008	
TEXT_EDI	V35	.001	.008	-.005	.000	.043
PLANNING	V36	.000	.003	.001	.000	.000
PRE_ALPH	V37	.000	.000	.001	.000	.001
ALPHA	V38	.000	.000	.001	-.001	.001
BETA	V39	-.001	-.007	.006	.000	.000
PRODUCTI	V40	.001	.011	.006	.001	.001
MATURE	V41	.000	.005	.001	.000	.001
INACTIVE	V42	.000	.000	-.001	.000	.000

		PLANNING V36	PRE_ALPH V37	ALPHA V38	BETA V39	PRODUCTI V40
PLANNING	V36	.048				
PRE_ALPH	V37	.008	.062			
ALPHA	V38	.003	-.001	.142		
BETA	V39	-.002	-.017	-.037	.236	
PRODUCTI	V40	-.010	-.019	-.062	-.090	.249
MATURE	V41	.000	-.001	-.006	-.010	-.006
INACTIVE	V42	.000	.000	-.001	-.003	-.005

		MATURE V41	INACTIVE V42

MATURE	V41	.042	
INACTIVE	V42	-.001	.016

Covariance Matrix (without moderator) – 2007 Sample (42 VARIABLES)

		LOGPAGEV V1	LOGDOWNL V2	LOGMEMBE V3	LOGACTIV V4	LOGEFFIC V5
LOGPAGEV	V1	5.411				
LOGDOWNL	V2	3.656	4.534			
LOGMEMBE	V3	.549	.538	.563		
LOGACTIV	V4	2.120	2.189	.605	2.927	
LOGEFFIC	V5	2.079	2.130	.616	2.912	3.262
LOGLIKEL	V6	-.107	-.113	-.011	-.146	.080
LOGAVERA	V7	1.059	1.104	.211	.974	1.007
LOG_LIFE	V8	.072	.151	.035	.081	.084
NO_RESTR	V9	-.003	.005	.019	.024	.023
MOD_REST	V10	-.001	.015	.012	.014	.015
BOTH_RES	V11	.064	.076	-.002	.014	.019
DUAL_LIC	V12	.026	.033	.018	.026	.024
END_USER	V13	.123	.173	.012	.070	.063
DEVELOPE	V14	.032	.046	.033	.040	.047
SYSTEM_A	V15	.032	.035	-.007	.018	.017
OTHERS_A	V16	.000	-.006	.009	.004	.006
ADVANCED	V17	.007	.018	.007	.006	.011
COMMUNIC	V18	-.004	.004	.005	-.001	-.002
DATABASE	V19	-.022	-.019	-.005	-.008	-.009
DESKTOP	V20	-.001	.006	.002	-.002	-.002
EDUCATIO	V21	-.012	-.013	.002	.001	.004
GAMES	V22	-.007	.000	.004	-.015	-.015
INTERNET	V23	.008	.019	.003	.003	.004
MULTIMED	V24	-.004	.001	.003	-.001	-.001
OFFICE	V25	.000	.004	.003	.011	.009
OTHER	V26	-.011	-.011	-.002	-.007	-.007
PRINTING	V27	.008	.006	.000	.002	.002
RELIGION	V28	-.001	.002	.001	.003	.003
SCIENTIF	V29	-.003	-.007	.002	.000	.001
SECURITY	V30	.009	.006	-.002	.003	.004
SOCIOLOG	V31	-.005	-.005	.000	-.001	-.002
SOFTWARE	V32	.002	.002	.000	.003	.002
SYSTEM	V33	.001	.000	.000	-.003	-.004
TERMINAL	V34	-.001	-.001	-.001	-.002	-.002
TEXT_EDI	V35	.005	.007	.000	.005	.004
PLANNING	V36	-.013	-.001	.000	-.007	-.010
PRE_ALPH	V37	-.043	-.052	.004	-.034	-.031
ALPHA	V38	-.078	-.077	-.001	-.068	-.073
BETA	V39	-.030	-.011	-.002	.006	.006
PRODUCTI	V40	.251	.270	.034	.183	.200
MATURE	V41	.051	.060	.012	.039	.042
INACTIVE	V42	-.020	-.014	-.003	-.007	-.009

		LOGLIKEL V6	LOGAVERA V7	LOG_LIFE V8	NO_RESTR V9	MOD_REST V10
LOGLIKEL	V6	.197				
LOGAVERA	V7	-.053	3.613			
LOG_LIFE	V8	.000	.106	.096		
NO_RESTR	V9	-.001	.023	.016	.186	

MOD_REST	V10	.001	.015	.010	-.021	.179
BOTH_RES	V11	.002	.001	.016	-.069	-.064
DUAL_LIC	V12	-.001	.013	.020	.099	.006
END_USER	V13	-.003	.033	.022	-.013	-.027
DEVELOPE	V14	.001	.056	.029	.030	.046
SYSTEM_A	V15	-.002	.020	.015	.003	-.013
OTHERS_A	V16	.003	-.013	.008	.004	-.001
ADVANCED	V17	.005	.001	.009	.002	.007
COMMUNIC	V18	-.001	-.002	.001	.003	.000
DATABASE	V19	-.001	-.005	.001	.002	.003
DESKTOP	V20	-.001	.001	.002	.002	.003
EDUCATIO	V21	.002	.005	.000	-.001	-.002
GAMES	V22	.000	-.004	.002	-.001	.001
INTERNET	V23	.001	.006	.001	.002	.000
MULTIMED	V24	.000	.001	.000	.000	.001
OFFICE	V25	-.002	-.001	-.001	.001	.000
OTHER	V26	.000	.006	.002	-.001	-.002
PRINTING	V27	.000	.000	.000	.000	.001
RELIGION	V28	.000	.004	.000	-.001	-.001
SCIENTIF	V29	.000	.004	.002	.001	.001
SECURITY	V30	.000	.000	-.001	-.001	.000
SOCIOLOG	V31	.000	.002	.000	.001	.000
SOFTWARE	V32	-.001	.005	.003	.013	.018
SYSTEM	V33	.000	.001	.001	.000	.000
TERMINAL	V34	.000	.002	.000	.000	.000
TEXT_EDI	V35	-.001	.002	.001	.001	-.001
PLANNING	V36	-.002	.014	.020	.013	.007
PRE_ALPH	V37	.005	-.034	.012	.004	.004
ALPHA	V38	.000	-.053	.005	.007	.002
BETA	V39	-.002	-.018	.006	.000	.008
PRODUCTI	V40	.003	.143	.016	.007	.006
MATURE	V41	.003	.027	.006	.004	.002
INACTIVE	V42	-.001	-.011	.000	.000	-.001

		BOTH_RES V11	DUAL_LIC V12	END_USER V13	DEVELOPE V14	SYSTEM_A V15
BOTH_RES	V11	.206				
DUAL_LIC	V12	.013	.115			
END_USER	V13	.062	.006	.247		
DEVELOPE	V14	-.034	.019	-.047	.226	
SYSTEM_A	V15	.024	.008	-.009	-.003	.193
OTHERS_A	V16	.012	.008	.004	.004	.004
ADVANCED	V17	.009	.006	.007	.012	.010
COMMUNIC	V18	-.002	.002	.000	.004	.000
DATABASE	V19	-.001	.002	-.003	.005	.001
DESKTOP	V20	.000	.002	.008	.002	.000
EDUCATIO	V21	.004	.001	.001	-.005	-.003
GAMES	V22	.002	.000	.006	-.001	-.005
INTERNET	V23	.000	.000	.003	.001	.007
MULTIMED	V24	.000	.000	.001	.002	.000
OFFICE	V25	.000	.001	.005	-.003	.000
OTHER	V26	.004	.001	.007	-.001	-.001
PRINTING	V27	.001	.000	.001	.000	.000
RELIGION	V28	.001	.000	.000	-.001	-.001
SCIENTIF	V29	.001	.001	-.001	.001	-.004
SECURITY	V30	.001	-.001	-.002	-.003	.013
SOCIOLOG	V31	-.001	.000	.000	.000	.000

SOFTWARE	V32	-.025	.002	-.030	.040	-.012
SYSTEM	V33	.000	.000	.001	.000	.001
TERMINAL	V34	.000	.000	.000	.000	.000
TEXT_EDI	V35	.000	.001	.001	.002	.000
PLANNING	V36	.012	.019	.016	.015	.001
PRE_ALPH	V37	.009	.008	.010	.011	.006
ALPHA	V38	.009	.011	.008	.011	-.001
BETA	V39	.017	.010	.017	.003	.006
PRODUCTI	V40	.003	.004	.003	.014	.018
MATURE	V41	-.001	.003	-.001	.005	.002
INACTIVE	V42	.001	.001	.001	-.001	.000

		OTHERS_A V16	ADVANCED V17	COMMUNIC V18	DATABASE V19	DESKTOP V20
OTHERS_A	V16	.108				
ADVANCED	V17	.001	.087			
COMMUNIC	V18	.002	.002	.038		
DATABASE	V19	.000	.002	.000	.041	
DESKTOP	V20	.000	.001	.002	.001	.027
EDUCATIO	V21	.004	.000	.000	.000	.000
GAMES	V22	.002	.000	-.001	.000	.000
INTERNET	V23	.000	.002	.007	.001	.002
MULTIMED	V24	.001	.000	.001	.000	.001
OFFICE	V25	.002	.002	.001	.002	.001
OTHER	V26	.005	.001	.000	.000	.001
PRINTING	V27	.001	.000	.000	.000	.000
RELIGION	V28	.000	.000	.000	.000	.000
SCIENTIF	V29	.001	.001	.001	.001	.000
SECURITY	V30	.001	.001	.000	.000	.000
SOCIOLOG	V31	.001	.000	.000	.000	.000
SOFTWARE	V32	-.004	.001	.001	.001	.002
SYSTEM	V33	.000	.001	.001	.000	.001
TERMINAL	V34	.000	.000	.001	.000	.000
TEXT_EDI	V35	.000	.000	.000	.000	.000
PLANNING	V36	.011	.011	.002	.003	.002
PRE_ALPH	V37	.004	.005	-.001	.001	.002
ALPHA	V38	.003	.008	.002	.001	.003
BETA	V39	.005	.005	.000	.002	.000
PRODUCTI	V40	.003	.004	.001	.002	.000
MATURE	V41	.003	.001	.000	-.001	.000
INACTIVE	V42	.000	.000	.000	.000	.000

		EDUCATIO V21	GAMES V22	INTERNET V23	MULTIMED V24	OFFICE V25
EDUCATIO	V21	.031				
GAMES	V22	.001	.045			
INTERNET	V23	-.001	-.001	.050		
MULTIMED	V24	.001	.001	.001	.008	
OFFICE	V25	.001	-.001	.001	.001	.029
OTHER	V26	.001	.000	-.001	.000	.001
PRINTING	V27	.000	.000	.000	.000	.001
RELIGION	V28	.000	.000	.000	.000	.000
SCIENTIF	V29	.003	.001	-.001	.000	.000
SECURITY	V30	-.001	-.001	.002	.000	.000
SOCIOLOG	V31	.001	.000	.000	.000	.000
SOFTWARE	V32	-.001	.000	.001	.000	.000
SYSTEM	V33	.000	.000	.000	.000	.000

TERMINAL	V34	.000	.000	.001	.000	.000
TEXT_EDI	V35	.000	.000	.000	.000	.001
PLANNING	V36	.001	.003	.002	.000	.001
PRE_ALPH	V37	-.001	.006	.000	.001	.000
ALPHA	V38	.000	.003	.000	.000	.001
BETA	V39	.002	-.001	.000	.000	.000
PRODUCTI	V40	.000	-.002	.002	.000	.001
MATURE	V41	.000	.000	.000	.000	-.001
INACTIVE	V42	.000	.001	.000	.000	.000

		OTHER V26	PRINTING V27	RELIGION V28	SCIENTIF V29	SECURITY V30
OTHER	V26	.031				
PRINTING	V27	.000	.009			
RELIGION	V28	.000	.000	.005		
SCIENTIF	V29	.001	.000	.000	.025	
SECURITY	V30	.000	.000	.000	.000	.027
SOCIOLOG	V31	.000	.000	.000	.000	.000
SOFTWARE	V32	-.001	.000	.000	.000	-.002
SYSTEM	V33	.001	.000	.000	.000	.001
TERMINAL	V34	.000	.000	.000	.000	.000
TEXT_EDI	V35	.000	.000	.000	.000	.000
PLANNING	V36	.004	.001	.000	.000	-.001
PRE_ALPH	V37	.000	.000	.000	.001	.001
ALPHA	V38	.002	.001	.000	.002	.000
BETA	V39	-.001	.001	.000	.000	.003
PRODUCTI	V40	.001	.000	.000	.000	.000
MATURE	V41	.000	.001	.000	.000	.000
INACTIVE	V42	.000	.000	.000	.000	-.001

		SOCIOLOG V31	SOFTWARE V32	SYSTEM V33	TERMINAL V34	TEXT_EDI V35
SOCIOLOG	V31	.003				
SOFTWARE	V32	.000	.109			
SYSTEM	V33	.000	.002	.010		
TERMINAL	V34	.000	.000	.000	.002	
TEXT_EDI	V35	.000	.002	.000	.000	.011
PLANNING	V36	.000	.001	.000	.000	.000
PRE_ALPH	V37	.000	-.002	.000	.000	.000
ALPHA	V38	.000	.002	.000	.000	.001
BETA	V39	.000	-.001	.001	.000	-.001
PRODUCTI	V40	.000	.005	.001	.000	.001
MATURE	V41	.000	.001	.000	.000	.001
INACTIVE	V42	.000	.000	.000	.000	.000

		PLANNING V36	PRE_ALPH V37	ALPHA V38	BETA V39	PRODUCTI V40
PLANNING	V36	.119				
PRE_ALPH	V37	.001	.104			
ALPHA	V38	-.002	-.006	.168		
BETA	V39	-.008	-.016	-.036	.242	
PRODUCTI	V40	-.007	-.017	-.059	-.079	.250
MATURE	V41	-.002	-.002	-.006	-.011	-.004
INACTIVE	V42	.000	-.001	-.001	-.004	-.007

		MATURE V41	INACTIVE V42

MATURE	V41	.047	
INACTIVE	V42	-.001	.023

Covariance Matrix (without moderator) – 2008 Sample (42 VARIABLES)

		LOGPAGEV V1	LOGDOWNL V2	LOGMEMBE V3	LOGACTIV V4	LOGEFFIC V5
LOGPAGEV	V1	5.393				
LOGDOWNL	V2	3.622	4.444			
LOGMEMBE	V3	.562	.545	.575		
LOGACTIV	V4	2.114	2.200	.617	2.934	
LOGEFFIC	V5	2.057	2.125	.632	2.929	3.326
LOGLIKEL	V6	-.105	-.112	-.010	-.130	.096
LOGAVERA	V7	1.080	1.118	.220	.988	1.056
LOG_LIFE	V8	.065	.104	.026	.063	.067
NO_RESTR	V9	-.003	-.006	.013	.016	.015
MOD_REST	V10	.001	.012	.012	.011	.016
BOTH_RES	V11	.036	.026	-.011	-.008	-.004
DUAL_LIC	V12	.018	.007	.007	.011	.009
END_USER	V13	.106	.138	.008	.060	.052
DEVELOPE	V14	.013	.029	.031	.027	.031
SYSTEM_A	V15	.028	.023	-.008	.014	.012
OTHERS_A	V16	.004	-.010	.004	.001	.002
ADVANCED	V17	.013	.015	.008	.007	.010
COMMUNIC	V18	-.011	-.004	.002	-.003	-.004
DATABASE	V19	-.021	-.020	-.005	-.010	-.011
DESKTOP	V20	-.001	.003	.000	-.003	-.003
EDUCATIO	V21	-.013	-.016	.002	-.001	.002
GAMES	V22	-.008	-.002	.003	-.017	-.018
INTERNET	V23	.011	.019	.004	.008	.009
MULTIMED	V24	-.003	.000	.003	-.002	-.002
OFFICE	V25	-.001	.001	.001	.008	.007
OTHER	V26	-.016	-.018	-.003	-.009	-.008
PRINTING	V27	.006	.003	.000	.002	.002
RELIGION	V28	-.001	.002	.002	.003	.003
SCIENTIF	V29	-.004	-.007	.003	.000	.001
SECURITY	V30	.011	.008	-.001	.006	.007
SOCIOLOG	V31	-.004	-.005	.000	-.001	-.002
SOFTWARE	V32	-.004	-.003	.000	.004	.005
SYSTEM	V33	.002	.000	.000	-.004	-.005
TERMINAL	V34	-.001	-.001	-.001	-.002	-.002
TEXT_EDI	V35	.003	.005	.000	.005	.003
PLANNING	V36	-.021	-.024	-.010	-.021	-.028
PRE_ALPH	V37	-.040	-.058	.000	-.033	-.027
ALPHA	V38	-.080	-.092	-.006	-.073	-.078
BETA	V39	-.038	-.034	-.004	-.006	-.008
PRODUCTI	V40	.233	.252	.032	.178	.195
MATURE	V41	.053	.060	.012	.039	.042
INACTIVE	V42	-.024	-.022	-.005	-.014	-.016

		LOGLIKEL V6	LOGAVERA V7	LOG_LIFE V8	NO_RESTR V9	MOD_REST V10
LOGLIKEL	V6	.181				
LOGAVERA	V7	-.032	3.717			
LOG_LIFE	V8	.001	.079	.063		

NO_RESTR	V9	-.002	.005	.003	.204	
MOD_REST	V10	.003	.010	.002	-.023	.191
BOTH_RES	V11	.000	-.006	-.001	-.061	-.053
DUAL_LIC	V12	-.003	-.003	.002	.136	.001
END_USER	V13	-.006	.029	.011	-.014	-.026
DEVELOPE	V14	.001	.048	.012	.023	.041
SYSTEM_A	V15	-.003	.016	.009	-.001	-.014
OTHERS_A	V16	.001	-.019	.003	.004	-.003
ADVANCED	V17	.004	.002	-.001	.002	.006
COMMUNIC	V18	-.001	-.002	-.001	.004	-.001
DATABASE	V19	-.001	-.003	-.001	.001	.003
DESKTOP	V20	-.001	.000	.001	.002	.003
EDUCATIO	V21	.002	-.002	-.002	-.002	-.002
GAMES	V22	-.001	-.002	.000	.000	.001
INTERNET	V23	.001	.005	.000	.002	-.001
MULTIMED	V24	.000	.001	.000	.000	.000
OFFICE	V25	-.001	.001	-.002	-.001	.000
OTHER	V26	.000	.002	.000	-.002	-.002
PRINTING	V27	.000	.001	.000	.000	.000
RELIGION	V28	.000	.003	.000	.000	-.001
SCIENTIF	V29	.000	.001	.001	-.001	.001
SECURITY	V30	.000	.000	-.002	-.001	.000
SOCIOLOG	V31	-.001	.001	.000	.001	.000
SOFTWARE	V32	.000	.005	.001	.010	.018
SYSTEM	V33	.000	.001	.000	.000	-.001
TERMINAL	V34	.000	.003	.000	.000	.000
TEXT_EDI	V35	-.001	.000	.001	.001	.000
PLANNING	V36	-.005	-.016	-.002	.012	.006
PRE_ALPH	V37	.005	-.039	.002	.003	.005
ALPHA	V38	.002	-.058	-.004	.006	.000
BETA	V39	-.003	-.024	.000	.000	.010
PRODUCTI	V40	.001	.148	.006	.006	.005
MATURE	V41	.001	.028	.004	.003	.001
INACTIVE	V42	.000	-.018	-.001	.001	-.001

		BOTH_RES V11	DUAL_LIC V12	END_USER V13	DEVELOPE V14	SYSTEM_A V15
BOTH_RES	V11	.180				
DUAL_LIC	V12	.011	.154			
END_USER	V13	.049	.002	.243		
DEVELOPE	V14	-.027	.015	-.045	.212	
SYSTEM_A	V15	.019	.003	-.010	-.006	.199
OTHERS_A	V16	.008	.007	.001	-.001	.002
ADVANCED	V17	.011	.006	.004	.011	.011
COMMUNIC	V18	-.001	.003	.000	.003	.000
DATABASE	V19	.001	.002	-.003	.005	.002
DESKTOP	V20	.000	.002	.007	.001	.000
EDUCATIO	V21	.005	-.001	.000	-.004	-.003
GAMES	V22	.003	.001	.007	-.001	-.005
INTERNET	V23	.000	.000	.002	.000	.006
MULTIMED	V24	.000	.001	.001	.002	-.001
OFFICE	V25	.001	.001	.004	-.002	.000
OTHER	V26	.003	-.001	.008	-.001	-.001
PRINTING	V27	.001	.001	.000	.000	.000
RELIGION	V28	.001	.000	.000	-.001	-.001
SCIENTIF	V29	.001	.000	.000	.001	-.004
SECURITY	V30	.002	.000	-.001	-.003	.014

SOCIOLOG	V31	.000	.000	.000	.000	.000
SOFTWARE	V32	-.018	.004	-.028	.036	-.012
SYSTEM	V33	.000	.000	.001	.000	.001
TERMINAL	V34	.000	.000	.000	.000	.000
TEXT_EDI	V35	.000	.000	.001	.002	.000
PLANNING	V36	.012	.019	.016	.009	-.006
PRE_ALPH	V37	.009	.008	.008	.014	.005
ALPHA	V38	.009	.010	.007	.010	.000
BETA	V39	.014	.010	.018	-.002	.006
PRODUCTI	V40	.004	.005	-.004	.010	.019
MATURE	V41	-.001	.001	.000	.005	.002
INACTIVE	V42	.001	.002	.000	-.001	.001

		OTHERS_A V16	ADVANCED V17	COMMUNIC V18	DATABASE V19	DESKTOP V20
OTHERS_A	V16	.117				
ADVANCED	V17	-.001	.115			
COMMUNIC	V18	.001	.002	.045		
DATABASE	V19	.000	.001	.000	.047	
DESKTOP	V20	.001	.002	.002	.001	.029
EDUCATIO	V21	.003	.000	-.001	.000	.000
GAMES	V22	.002	-.001	.000	-.001	-.001
INTERNET	V23	.001	.003	.008	.001	.002
MULTIMED	V24	.001	.000	.001	.000	.001
OFFICE	V25	.002	.003	.002	.002	.001
OTHER	V26	.005	.000	.000	.001	.001
PRINTING	V27	.001	.000	.000	.000	.000
RELIGION	V28	.000	.000	.000	.000	.000
SCIENTIF	V29	.000	.001	.000	.001	.000
SECURITY	V30	.002	.001	.000	.000	-.001
SOCIOLOG	V31	.001	.000	.000	.000	.000
SOFTWARE	V32	-.003	.002	.001	.002	.002
SYSTEM	V33	.000	.001	.001	.000	.001
TERMINAL	V34	.000	.000	.001	.000	.001
TEXT_EDI	V35	.000	.001	.000	.000	.000
PLANNING	V36	.010	.008	.004	.003	.001
PRE_ALPH	V37	.002	.004	.000	.001	.002
ALPHA	V38	.002	.009	.003	.002	.003
BETA	V39	.005	.007	.000	.003	-.001
PRODUCTI	V40	.002	.003	.000	.001	.000
MATURE	V41	.002	.002	.000	-.001	-.001
INACTIVE	V42	-.001	.001	.000	.000	.000

		EDUCATIO V21	GAMES V22	INTERNET V23	MULTIMED V24	OFFICE V25
EDUCATIO	V21	.037				
GAMES	V22	.001	.052			
INTERNET	V23	-.001	-.001	.055		
MULTIMED	V24	.000	.001	.002	.011	
OFFICE	V25	.001	-.002	.001	.001	.031
OTHER	V26	.001	.000	-.001	.000	.001
PRINTING	V27	.000	-.001	.000	.000	.001
RELIGION	V28	.000	.000	.000	.000	.000
SCIENTIF	V29	.003	.001	-.001	.000	.000
SECURITY	V30	-.001	-.001	.003	.000	.001
SOCIOLOG	V31	.001	.000	.000	.000	.000
SOFTWARE	V32	-.002	.000	.001	.000	.001

SYSTEM	V33	.000	.000	.000	.000	.000
TERMINAL	V34	.000	.000	.001	.000	.000
TEXT_EDI	V35	.000	.000	.000	.000	.001
PLANNING	V36	.003	.004	.004	.001	.003
PRE_ALPH	V37	.000	.006	.000	.001	-.001
ALPHA	V38	-.001	.004	.000	.000	.001
BETA	V39	.001	-.001	.000	.000	.000
PRODUCTI	V40	.001	-.004	.002	.000	.001
MATURE	V41	-.001	-.001	.000	.000	-.001
INACTIVE	V42	.000	.001	.000	.000	.000

		OTHER V26	PRINTING V27	RELIGION V28	SCIENTIF V29	SECURITY V30
OTHER	V26	.035				
PRINTING	V27	.000	.010			
RELIGION	V28	.000	.000	.005		
SCIENTIF	V29	.001	.000	.000	.027	
SECURITY	V30	.001	.000	.000	-.001	.030
SOCIOLOG	V31	.000	.000	.000	.000	.000
SOFTWARE	V32	-.001	.000	.000	.000	-.002
SYSTEM	V33	.001	.000	.000	.000	.001
TERMINAL	V34	.000	.000	.000	.000	.000
TEXT_EDI	V35	.000	.000	.000	.000	.000
PLANNING	V36	.004	.001	.000	.000	-.001
PRE_ALPH	V37	-.001	.000	.000	.001	.002
ALPHA	V38	.003	.000	.000	.002	-.001
BETA	V39	.000	.000	.000	.000	.002
PRODUCTI	V40	.001	.000	.001	.000	.001
MATURE	V41	.000	.001	.000	.000	.000
INACTIVE	V42	-.001	.000	.000	.000	-.001

		SOCIOLOG V31	SOFTWARE V32	SYSTEM V33	TERMINAL V34	TEXT_EDI V35
SOCIOLOG	V31	.004				
SOFTWARE	V32	.000	.113			
SYSTEM	V33	.000	.001	.011		
TERMINAL	V34	.000	.000	.000	.003	
TEXT_EDI	V35	.000	.001	.001	.000	.012
PLANNING	V36	.001	.002	.000	.000	.001
PRE_ALPH	V37	.000	-.001	.000	.000	.000
ALPHA	V38	.000	.002	.000	.000	.000
BETA	V39	.000	-.001	.000	.000	.000
PRODUCTI	V40	.000	.007	.001	.000	.001
MATURE	V41	.000	.001	.000	.000	.001
INACTIVE	V42	.000	.000	.000	.000	.000

		PLANNING V36	PRE_ALPH V37	ALPHA V38	BETA V39	PRODUCTI V40
PLANNING	V36	.166				
PRE_ALPH	V37	-.004	.126			
ALPHA	V38	-.010	-.011	.182		
BETA	V39	-.017	-.018	-.033	.245	
PRODUCTI	V40	-.015	-.017	-.056	-.072	.249
MATURE	V41	-.003	-.002	-.007	-.010	-.004
INACTIVE	V42	-.001	-.001	-.002	-.006	-.009

		MATURE V41	INACTIVE V42
MATURE	V41	.049	
INACTIVE	V42	-.001	.032

APPENDIX D – COVARIANCE MATRICES (MODERATION TEST)

Covariance Matrix (moderator) – 2006 Sample Low-Complexity

	LOGPAGEV V 1	LOGDOWNL V 2	LOGMEMBE V 3	LOGACTIV V 4	LOGEFFIC V 5
LOGPAGEV V 1	6.356				
LOGDOWNL V 2	4.024	5.112			
LOGMEMBE V 3	.731	.691	.628		
LOGACTIV V 4	2.560	2.586	.713	3.266	
LOGEFFIC V 5	2.575	2.610	.727	3.321	3.716
LOGLIKEL V 6	-.081	-.073	-.014	-.105	.094
LOGAVERA V 7	1.139	1.255	.247	1.169	1.213

	LOGLIKEL V 6	LOGAVERA V 7
LOGLIKEL V 6	.167	
LOGAVERA V 7	-.036	3.483

Covariance Matrix (moderator) – 2006 Sample High-Complexity

	LOGPAGEV V 1	LOGDOWNL V 2	LOGMEMBE V 3	LOGACTIV V 4	LOGEFFIC V 5
LOGPAGEV V 1	7.388				
LOGDOWNL V 2	4.532	6.165			
LOGMEMBE V 3	.695	.851	.735		
LOGACTIV V 4	2.462	2.965	.614	3.312	
LOGEFFIC V 5	2.459	3.082	.640	3.422	3.790
LOGLIKEL V 6	-.037	-.001	.005	-.013	.144
LOGAVERA V 7	1.360	1.406	.360	1.050	1.140

	LOGLIKEL V 6	LOGAVERA V 7
LOGLIKEL V 6	.133	
LOGAVERA V 7	.027	3.232

Covariance Matrix (moderator) – 2007 Sample Low-Complexity

	LOGPAGEV V 1	LOGDOWNL V 2	LOGMEMBE V 3	LOGACTIV V 4	LOGEFFIC V 5
LOGPAGEV V 1	5.781				
LOGDOWNL V 2	3.779	4.717			
LOGMEMBE V 3	.711	.665	.626		
LOGACTIV V 4	2.528	2.480	.704	3.286	
LOGEFFIC V 5	2.510	2.480	.724	3.334	3.672
LOGLIKEL V 6	-.095	-.082	-.008	-.107	.090
LOGAVERA V 7	1.154	1.245	.255	1.218	1.275

	LOGLIKEL V 6	LOGAVERA V 7
LOGLIKEL V 6	.184	
LOGAVERA V 7	-.032	3.711

Covariance Matrix (moderator) – 2007 Sample High-Complexity

		LOGPAGEV V 1	LOGDOWNL V 2	LOGMEMBE V 3	LOGACTIV V 4	LOGEFFIC V 5
LOGPAGEV	V 1	6.722				
LOGDOWNL	V 2	4.507	6.344			
LOGMEMBE	V 3	.612	.832	.687		
LOGACTIV	V 4	2.289	3.064	.631	3.442	
LOGEFFIC	V 5	2.264	3.100	.687	3.608	4.017
LOGLIKEL	V 6	-.082	-.068	.038	.010	.185
LOGAVERA	V 7	1.430	1.586	.374	.995	1.139

		LOGLIKEL V 6	LOGAVERA V 7
LOGLIKEL	V 6	.167	
LOGAVERA	V 7	.050	3.071

Covariance Matrix (moderator) – 2008 Sample Low-Complexity

		LOGPAGEV V 1	LOGDOWNL V 2	LOGMEMBE V 3	LOGACTIV V 4	LOGEFFIC V 5
LOGPAGEV	V 1	5.750				
LOGDOWNL	V 2	3.701	4.571			
LOGMEMBE	V 3	.732	.665	.642		
LOGACTIV	V 4	2.514	2.509	.732	3.362	
LOGEFFIC	V 5	2.504	2.506	.760	3.408	3.770
LOGLIKEL	V 6	-.102	-.091	-.005	-.113	.083
LOGAVERA	V 7	1.161	1.242	.280	1.256	1.317

		LOGLIKEL V 6	LOGAVERA V 7
LOGLIKEL	V 6	.169	
LOGAVERA	V 7	-.031	3.710

Covariance Matrix (moderator) – 2008 Sample High-Complexity

		LOGPAGEV V 1	LOGDOWNL V 2	LOGMEMBE V 3	LOGACTIV V 4	LOGEFFIC V 5
LOGPAGEV	V 1	6.643				
LOGDOWNL	V 2	4.551	6.280			
LOGMEMBE	V 3	.630	.819	.688		
LOGACTIV	V 4	2.317	3.028	.619	3.233	
LOGEFFIC	V 5	2.225	3.006	.661	3.417	3.869
LOGLIKEL	V 6	-.143	-.099	.021	.017	.193
LOGAVERA	V 7	1.450	1.561	.368	1.035	1.208

		LOGLIKEL V 6	LOGAVERA V 7
LOGLIKEL	V 6	.149	
LOGAVERA	V 7	.069	3.260

APPENDIX E – MODERATION CONSTRAINTS (LAGRANGE TEST)

UNIVARIATE TEST STATISTICS:

NO	CONSTRAINT	CHI-SQUARE	PROBABILITY
1	CONSTR: 1	.078	.780
2	CONSTR: 2	.034	.853
3	CONSTR: 3	.064	.801
4	CONSTR: 4	.116	.733
5	CONSTR: 5	.429	.513
6	CONSTR: 6	.098	.755
7	CONSTR: 7	.037	.848
8	CONSTR: 8	.070	.791
9	CONSTR: 9	.210	.647
10	CONSTR: 10	.489	.485
11	CONSTR: 11	1.191	.275
12	CONSTR: 12	.001	.977
13	CONSTR: 13	.146	.703
14	CONSTR: 14	.552	.457
15	CONSTR: 15	.244	.621
16	CONSTR: 16	.005	.944
17	CONSTR: 17	.007	.932
18	CONSTR: 18	.430	.512
19	CONSTR: 19	.000	.988
20	CONSTR: 20	.446	.504

	CUMULATIVE MULTIVARIATE STATISTICS				**UNIVARIATE INCREMENT**	
STEP	PARAMETER	CHI-SQUARE	D.F.	PROBABILITY	CHI-SQUARE	PROBABILITY
1	CONSTR: 11	1.191	1	.275	1.191	.275
2	CONSTR: 1	2.815	2	.245	1.624	.203
3	CONSTR: 10	3.351	3	.341	.536	.464
4	CONSTR: 18	3.789	4	.435	.438	.508
5	CONSTR: 20	4.185	5	.523	.396	.529
6	CONSTR: 13	4.392	6	.624	.207	.649
7	CONSTR: 7	4.647	7	.703	.255	.613
8	CONSTR: 8	4.706	8	.788	.059	.808
9	CONSTR: 19	4.754	9	.855	.048	.826
10	CONSTR: 17	4.796	10	.904	.041	.839
11	CONSTR: 16	4.851	11	.938	.055	.815
12	CONSTR: 14	4.889	12	.962	.038	.845
13	CONSTR: 4	5.186	13	.971	.297	.586
14	CONSTR: 9	5.511	14	.977	.325	.569
15	CONSTR: 12	5.576	15	.986	.066	.798
16	CONSTR: 15	5.635	16	.992	.059	.808
17	CONSTR: 2	5.683	17	.995	.047	.828
18	CONSTR: 3	5.727	18	.997	.045	.833
19	CONSTR: 6	5.743	19	.998	.015	.902
20	CONSTR: 5	5.743	20	.999	.000	.989

VITA

Graduate School
Southern Illinois University

Carlos D. Santos Jr. Date of Birth: May 22, 1981

1925 Evergreen Terrace Dr. E – Ap. 8, Carbondale, Illinois 62901

carlosdenner@gmail.com

State University of Minas Gerais at Montes Claros (UNIMONTES)
Bachelor of Science, Business and Administration, Dec 2002

Federal University of Minas Gerais (UFMG)
Master of Science in Business Administration, Strategic Management, Feb 2005

Dissertation Title:

A Total Cost of Ownership (TCO) Comparative Analysis between Linux and Windows. [available in Portuguese]

Major Professor: Márcio A. Gonçalves, Ph.D.