Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development

Kevin Crowston

Syracuse University
School of Information Studies

4–206 Centre for Science and Technology
Syracuse, NY 13244–4100

Telephone: +1 (315) 443–1676
Fax: +1 (315) 443–5806
Email: crowston@syr.edu

Barbara Scozzi

Politecnico di Bari
Dipartimento di Ingegneria Meccanica e Gestionale

Viale Iapigia 182
Bari. Italy 70126

Telephone +39 (080) 5962725
Fax +39 (080) 5962725
Email: bscozzi@dimeg090.poliba.it

To appear in *IEE Proceedings Software*

Keywords: OSS, virtual organizations, competency rallying (CR)

Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development

Abstract

The contribution of this paper is the identification and testing of factors important for the success of Open Source Software (OSS) projects. We present an analysis of OSS communities as virtual organizations and apply Katzy and Crowston's competency rallying (CR) theory to the case of OSS development projects. CR theory suggests that project participants must develop necessary competencies, identify and understand market opportunities, marshal competencies to meet the opportunity and manage a short-term cooperative process. Using data collected from 7477 OSS projects hosted by the SourceForge system (http://sourceforge.net/), we formulate and test a set of specific hypotheses derived from CR theory. The empirical data analysis supports the majority of these hypotheses, suggesting that CR theory provides a useful lens for studying OSS projects.

Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development

1. Introduction

Open Source Software (OSS) is software whose source code is distributed without charge or limitation on possible modifications and distributions by third parties (http://www.opensource.org/). Each OSS project is the product of a community comprising a few, dozens or even hundreds of geographically distributed developers, who contribute via the Internet for the sake of peer recognition and personal satisfaction [1, 2]. Due to the success of software such as Linux, interest in this approach to software development has increased, both in the academic and commercial world [3]. This interest was further augmented when Microsoft cited Linux as one of its main competitors in trying to deny the corporation's monopoly power [4].

Many researchers have investigated the nature and the characteristics of OSS projects and their developer communities. Different interpretations have emerged. On the one hand, OSS communities have been depicted as completely open and chaotic markets. Due to their inherent instability, such markets have been considered ineffective for pursuing innovative projects in the present turbulent competitive environment [5]. On the other hand, the OSS development process has been proposed as a paradigm to solve the "software crisis" [6, 7]. Between these two extremes, some studies have examined advantages and disadvantages of the OSS development model and have proposed alternative interpretations [3, 8, 9]. The small number of organizational studies related to the OSS development process and the different interpretations show the need for further inquiries.

In this paper, we attempt to further the analysis of the OSS model. Our research question is to identify and test factors important for the success of open source projects. After

a brief discussion of the literature on OSS, we present an analysis of OSS communities as virtual organizations. By taking advantage both of the Internet and dispersed individual knowledge, these virtual organizations are able to bring together a diverse and geographically distributed set of individuals to rapidly develop reliable software. As a basis for this analysis, we present a theory of competency rallying (CR) [10], which suggests factors important for the success of projects. We then formulate and test a set of hypotheses using data on OSS projects supported by the "SourceForge" project (http://sourceforge.net/). Our analysis shows that the CR framework successfully identifies factors related to the success of OSS projects. Our results should therefore be useful in understanding which projects are likely to succeed using an OSS approach.

The paper is organized as follows. In the remainder of this section, we briefly review the literature on OSS development. In the section 2, we introduce the theory used for our analysis of OSS communities as virtual organizations. We then use this theory to develop four general propositions about factors important for OSS project success. In section 3, we discuss the available data and formulate more specific hypotheses to test the propositions. We present the results of our analysis in section 4. Finally, we conclude by discussing our results and drawing some implications for future research and practice.

1.1. Open Source Software: A brief introduction

Open Source Software (OSS) is a broad term used to embrace software that is developed and released under some sort of "open source" license that allows inspection and reuse of the software's source code. There are thousands of OSS projects spanning a range of applications [11]. The success of OSS projects has been attributed to their speed of development and the reliability, portability, and scalability of the resulting software [12-17]. In turn, the quality of the software and speed of development have been attributed to the fact

that the source code is open to inspection by and contributions from any interested individual. As Raymond [6] put it, "Given enough eyeballs, all bugs are shallow."

The literature on OSS has mostly focused on two separate questions. First, various explanations have been proposed for the interest in OSS, particularly the decision by individuals to contribute to projects, often without pay. Other authors have attempted to explain the success of OSS development, which is the focus of this paper. The most well known model is the bazaar metaphor proposed by [6], and criticized by [9, 18].

It is noteworthy that much of the literature on OSS has been written by developers and consultants directly involved in the OSS community. These contributions are significant as they point out the economic relevance of OSS as well as vulnerable aspects of the new development process. Yet many of these studies seem to be animated by partisan spirit, hype or skepticism [19]. Few studies come from organizational theorists and there are only a few well-documented case studies, most of which discuss successes rather than failures. Finally, with a few exceptions [e.g., 20, 21] the proposed models are usually descriptive and based on a small number of cases. This is both indicative of the relative novelty of the issue and the lack of a clear theoretical framework to describe and interpret the OSS phenomenon [3].

In this paper we identify factors important for OSS project success top-down by applying a theoretical perspective, thus offering an alternative to bottom-up research. We then test this framework against a large number of projects, thus addressing the limitations in prior work. In our view, an OSS community can be analyzed as a virtual organization [7, 22] and the Competency Rallying (CR) theory [10] can be used to suggest key success factors. Both the proposed perspective and the CR theory are discussed in the next section.

2. Theory: OSS community as a virtual organization

Many studies argue that as the extensive use of information and communication technology (ICT) decreases the cost of coordination, firms will increasingly turn to market-based transactions rather than in-house production [e.g., 23]. An increasing corpus of studies has also indicated the need to leverage inter-firm relations to create a competitive advantage. In a turbulent environment, strategic alliances and networking are considered the only viable ways to exploit emerging market opportunities, as the rate of change makes it impossible for single firms to develop all the necessary competencies [24]. Under these circumstances, "virtual organizations" become possible and desirable.

Different authors have proposed a range of definitions of a "virtual organization", each highlighting different characteristics [e.g., 10, 25, 26-32]. In this paper, we refer to the definition proposed by Ahuja and Carley [33], who emphasize three main aspects of virtual organizations, namely the existence of an interest or goal shared by the members of the group, geographical distribution, and the use of ICT to communicate and manage the interdependencies. These factors characterize OSS development communities, so by this definition, OSS is developed by a virtual organization. Both as developers and users, OSS community members share a common interest in software being developed (though this interest may be non-financial). As well, OSS communities are typically geographically distributed and, in these cases, the Internet is the main tool used to coordinate different actions [4, 34]. Mailing lists, bulletin boards and source code control systems (e.g., CVS) are the main instruments used to manage interdependencies.

Since an OSS community is an example of a virtual organization, studies of OSS communities can contribute to and benefit from our knowledge of the functioning of virtual organizations. For example, Markus et al. [22] investigated the success of the OSS business

model so as to transfer it to other industrial settings. In this paper, we will take the opposite approach and apply a theory proposed to explain the success of an industrial virtual organization to the case of OSS development. The theory suggests a set of factors needed for the success of projects such as OSS development. As well, the case of OSS development provides a further test of this theory, thus supporting generalization beyond the specific situation for which it was developed.

2.1 The Competency Rallying theory

In this paper, we apply Katzy and Crowston's [10] competency rallying (CR) theory to OSS projects. The CR theory was developed to explain the success of projects undertaken by the "Virtual Factory" [35], an organized network for industrial cooperation in the precision manufacturing industry in the south of Germany, northern Switzerland, Austria, and Liechtenstein. The theory integrates the resource-based theory of the firm with more recent literature on networking, virtual organizations and dynamic capabilities [36] to suggest factors important for the success of a virtual organization.

The reasons for the success (or failure) of virtual organizations have been investigated in a variety of theories adopting different perspectives. The CR theory has been used here because the adopted perspective is most suitable to the aim of the paper. The theory illuminates factors that have been pointed out as most relevant (e.g., competencies, interests, and coordination). Also, because of its currency, the theory represents the state of the art of the research related to virtual organizations. Finally, the case of OSS, different in many details from the original setting for CR theory, can provide a good empirical test for the generality of the theory.

CR theory identifies four generic capabilities that a virtual organization must possess to succeed, namely: 1) identification and development of individual competencies; 2)

identification of market opportunities; 3) marshalling of competencies; and 4) management of a short-term cooperative effort. These four capabilities are all considered necessary for a project to be successful in a virtual organization. Of course, these factors are relevant in every kind of organization and project. Yet, the possible lack of common culture and climate, the importance of novel market opportunities and novel combinations of competencies and the existence of different utility functions and possible time constraints make their adoption harder in a virtual context. Thus, the theory stresses the key role these four sets of capabilities play in a virtual organization.

For OSS development, the CR theory points out that the existence of unique competencies and shared needs make user-developers join and coordinate their skills in short-term cooperation effort producing high quality software. In the remainder of this section, findings from the OSS literature are summarized into general propositions for the success of OSS projects. Based on these propositions, in the following sections more specific hypotheses will be formulated and empirically tested.

(1) Identification and development of competencies

The first capability identified by Katzy and Crowston [10] is the ongoing development and refinement of individual participants' competencies, that is, the specific skills and know-how that firms develop over time. The resource-based view of the firm suggests that all successful firms posses a stock of such distinctive competencies. CR theory builds on this insight by noting that the members of a virtual organization must jointly possess, maintain and refine a stock of complementary competencies in order for them to be available to the virtual organization.

For OSS projects, the main competencies needed are the design and programming skills possessed by the individual participants [37]. Faraj and Sproull [38] talk more generally

of the importance of "expertise". Participants autonomously develop their competencies in the course of their on-going "real world" activities and sharpen them in the course of an OSS project. For example, each participant must know some particular languages, operating systems or programming environments and have developed experiences in specific software applications. The existence of a variety of competencies is one of the elements that allow user-developers to create good and successful software. In summary, we argue that:

P1. The more available the required competencies, the more successful the OSS project.

(2) Identification of market needs

The second set of capabilities in the CR theory is the ability to recognize a market need that might be met by the virtual organization. This capability extends beyond any single participant, since the virtual organization embodies the combined capabilities of all members. As a result, an important development in a virtual organization is the capability to recognize and exploit new market opportunities that the virtual organization can meet, but which are beyond the capabilities of any single participant.

The literature on OSS development makes it clear that OSS projects usually originate from a personal need [2, 39]. Such needs also attract the attention of other user-developers and inspire them to contribute to the project. This approach to software offers some real benefits in the design process. OSS is software that "make sense" to users because users are in many cases the same as developers. In this way, the ambiguity that often characterizes the identification of user needs or requests for improvement in traditional software development process is eliminated: programmers know their own needs [40]. Common language and knowledge reduce this ambiguity [16]. A main implication can be derived: the majority of OSS projects will be for products that meet developers' needs [41]. In summary, we argue that:

P2. The more readily developers can recognize the needs and problems addressed by the project, the more successful the OSS project.

(3) Marshalling of competencies

The third set of capabilities in the CR theory is the ability to identify and bring together participants with the competencies necessary to address an identified market opportunity, a process Katzy and Crowston refer to as "competency marshalling". Again, such a capability is usually a new development for members of a virtual organization, as it requires being able to match a wider range of market opportunities and competencies than are traditionally faced by any individual participant.

In the case of an OSS project, this capability implies the ability to bring together individuals with the necessary competencies to work on a given project. Faraj and Sproull [38] note the importance for software development teams of a similar process called "expertise coordination", which they define as "knowing where expertise is located, knowing where expertise is needed, and bringing needed expertise to bear" (p. 1554). Koch and Schneider [42] claim that the participation of programmers is the most important aspect in OSS development projects.

The process of competency marshalling appears to occur more or less spontaneously in OSS. The OSS community represents a nexus of exchanges in which people report bugs expecting that other members will fix them. Similarly those who fix bugs expect other developers to contribute to other parts of the project [43]. Thus, the return-on-investment of the effort and time investment is represented by the solution of the need [44, 45]. Reputation is another important aspect—the community is in fact frequently described as being based on peer recognition and in some cases on a "cult of the personality". In particular, peer

recognition is a value for the community that can sometimes lead to employment opportunities or access to venture capital [22].

This analysis can also be applied on a smaller grain to the on-going tasks that arise in a project. Take for example bug fixing, often used as an example of the responsiveness of the OSS process. The capability for competency marshalling presumes that participants in a project are aware of the experts for a particular type of bug, resulting in "a natural deferring process whereby most participants immediately realize which subgroup or person is best suited for fix a particular bug" [46]. In this way, competency marshalling is undertaken at a more tacit level as it emerges during the development of the different projects.

In summary, we argue that:

P3. The more quickly and accurately competencies can be marshalled, the more successful the OSS project.

(4) Short-term cooperation

The final set of capabilities in the CR theory is the ability to manage a short-term cooperative process. For a virtual organization to succeed, it is necessary to improve the capability of participants to work across organizational boundaries to exploit the diverse competencies necessary for a given project.

It is hard to clearly understand all the adopted coordination mechanisms without taking into account the peculiarities of OSS. Well-designed software has a modular architecture that allows users to extend the system's functionality without changing the core functionality [43, 44, 47]. In a word, the systems are extensible. This implies that the self-governance mechanisms pointed out in many studies [6, 22] are possible because most of the dependencies emerging among developers are pooled rather than sequential or reciprocal [48].

For these dependencies, standardization is a sufficient coordination mechanism (e.g., contributing to a mailing list or to the CVS tree according to certain standard).

In one of the few empirical studies on OSS characteristics, Godfrey and Tu [1] pointed out that Linux is less complex than it might appear by simply counting the lines of code (2.2 million at the time). By analyzing the different sub-systems, it emerged that the section growing most quickly was device drivers and other hardware adaptation layers. These pieces of code are indeed complicated but they are also self-contained, interacting with the rest of the system in a limited and well-defined fashion. The creation of self-contained tasks is one of the traditional ways proposed in the literature to reduce information needs and coordination effort [49]. Two levels of coordination—direct control by leaders and maintainers, self-contained tasks for developers or, as stated by Bezroukov [18], cathedral for the core of the project and bazaar for the peripheral parts—keep the community from complete chaos. In summary, we argue:

P4. *The greater the ability to manage short-term cooperation, the more successful the OSS project.*

In summary, Katzy and Crowston's [10] competency rallying theory suggests four interrelated sets of capabilities that are necessary for the success of any project, but particularly vital for a virtual organization. Applied to an open source project, the theory suggests four general propositions about factors that contribute to a project's success. The propositions suggests that the existence of shared competencies and common needs mixed with the ability of project leaders and the adoption of *ad hoc* coordination mechanisms enable user-developers to marshal needed competencies and produce, through short-term cooperation, successful software.

3. Data and hypotheses

In this section, testable hypotheses are formulated from the propositions above. However, before presenting them, we will first discuss the data we used and their source.

3.1 Source of data: The SourceForge project

Rather than collecting data ourselves, we based our analysis on data collected by the SourceForge project (http://sourceforge.net/), a free service created to support OSS projects. SourceForge provides projects with a project home page, CVS repositories, mailing lists, bug tracking service, task management software and permanent file archival. At the time of our study, it supported more than 15,000 projects and had more than 115,000 registered users[1].

For each project, SourceForge records a brief description and the registration date (and thus lifespan). Many of the projects on SourceForge are also classified on the following project dimensions, as shown in Figure 1:

- development status (planning, alpha, beta, production, etc.),
- environment (terminal, web, daemon, etc.),
- intended audience (end user, systems administrator, developer),
- license,
- natural language,
- operating system,
- programming language, and
- topic (from a hierarchical list).

[1] Data are updated to February 2, 2001.

The system reports project activity on the various services: number of page views, downloads, bug reports or support requests, etc. These data are available for various time periods, as shown in Figure 2. For this project, we used the lifetime statistics. The level of activity is summarized as a percentile and as a ranking among all projects. Finally, each project has a list of administrators and developers. Each administrator's record includes the date they joined SourceForge. As well, some of these individuals are peer rated as to their reliability and their coding, design and management skills. A few are ranked as to their overall peer rating. (Eric Raymond, the author of the "Cathedral and the Bazaar", was ranked number 1 at the time of our data collection). Figure 3 shows an example of a developer statistics page.

Insert Figure 1, Figure 2 and Figure 3 about here

SourceForge provides an interesting and potentially valuable source of information on OSS projects. However, there are limitations to the use of the data. First, SourceForge does not host some of the most well known OSS projects, such as Linux or Apache. For understanding the general dynamics of OSS development though, the omission of these two projects is insignificant given the inclusion of 15,000 others. Indeed, attention only to the largest and most successful OSS projects provides a misleading view of the general state of OSS development.

A more serious limitation is that the available data is predefined, non-modifiable and limited. Specific hypotheses must be based on this given set of variables to be testable, restricting our analysis. Unfortunately, the variables do not perfectly overlap the concepts proposed in the propositions developed above. To test our propositions, we identify available variables that provide an indication of each construct, but acknowledge that these measures

are imperfect. However, while this process adds noise, making it harder to find reliable relationships, it should not add bias, meaning that the relationships we do find are "true" and not artifacts of the testing process.

We note as well that our predicament is not at all unusual. All researchers who work with secondary data face such problems. We consider the limitations acceptable, considering the cost of developing a similar data set and the potential benefits achievable from the information available, and especially given the limited number of existing quantitative studies of OSS projects. The development of better metrics is a goal of further research. (We note that because SourceForge is itself an OSS project, it is possible to suggest that additional data be collected in future versions.)

3.2 Operationalization of constructs and hypotheses

To validate the propositions, the main concepts—namely OSS project success, user-developers' competencies and needs, the ability to marshalling competencies, and the management of short-term cooperation relationships—are operationalized in terms of available variables in the SourceForge project. In this section, we will specifically discuss the variables we chose to operationalize the constructs discussed above. A summary of variables and their definition is given in Table 1.

Insert Table 1 about here

Success

The outcome variable for CR theory is project success. In the original application of CR theory, success was assessed by the degree to which the product developed met the needs

of the customer of the project. For software, success can be interpreted in several different ways:

- in a commercial sense, success is measured by the number of users of the software (i.e., total sales).
- from the perspective of a user, we might look at frequency of use or satisfaction with the software.
- from the perspective of software developers, success might be an active project or one that successfully releases software.

Some of variables collected on SourceForge projects can be used as proxies for the success of the software project. As none of the variables alone represents an adequate measure of the success, in this paper we will compare three different measures of success. First, we consider success as measured by the interest showed by users. According to this measure, a more successful project is one that creates software that is used. For this purpose, we use a scale comprising the number of downloads of the software and page views (USE)[2]. To form the scale, these two variables were transformed and then normalized prior to averaging to ensure they contributed equal variance to the final scale. Analysis of the reliability of the resulting USE scale shows acceptable results (Cronbach's $\alpha = 0.71$, N = 7477). Second, we consider the project's development status (e.g., preplanning, alpha, beta, production, etc.) (DEVSTATUS). According to this measure, a more successful project is one that is in a more advanced state of development. Finally, we consider the intensity of work undertaken by developers (ACTIVITY). According to this measure, a more successful project is one that has more activity. For the activity measure, we used the transformed percentile rank of the activity.

Availability of competencies

For each project we wanted a score that would reflect the availability of required competencies, that is, the skills and capabilities of the developers. In the case of OSS, developer might include knowledge of specific operating systems, programming language and programming environments, which are recorded by SourceForge. We decided to use the programming language of the project, since this factor seemed likely to be of concern primarily to the developers. We decided not to consider operating system and environment, since these factors seemed likely to directly influence the use of projects, one of our success measures—there are simply more potential users for Windows software than for Amiga software. There are of course many other kinds of competencies that a project might require, but unfortunately, we have no data regarding these.

To estimate the diffusion of the competencies in each programming language, we used the number of projects in the sample written in that language. In other words, we took the fact that there are more projects written in the C language as an indication that the competency of writing in C is more widely available. For example, there were 2441 projects written in the C programming language, so each of these projects had a score of 2441 for programming language use. On the other hand, there were only 10 projects written in Ada, and so these projects had a score of 10 for programming language use. The resulting variable is PLPOP.

We propose the following hypothesis:

H1. Projects using more common programming languages are more successful (i.e., a) more active, b) in more advanced states of development and c) more used).

[2] We assume that a proportion of software downloads are actually used, making downloads a reasonable proxy for the number of users.

Recognition of market opportunities

Successful projects address needs and the problems of the community. We used a project's stated topic (TOPIC) and intended audience (AUDIENCE) as an indication of these needs. The OSS community is mainly composed of user-developers. Thus, projects that address the needs and solve the problems of this community should be more successful. In particular, we expect that projects (software) that list developers and system administrators as their stated audience will be more common, have higher activity percentile, be in a more advanced development status and be more used than those described as intended for end-users. Also, we expect 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. Projects on more familiar topics should also show higher use.

We propose the following specific hypotheses:

H2.1 *Projects intended for developers are more successful than projects developed for system administrators, which are more successful than projects developed for end-users. As well, d) there are more projects intended for developers than for system administrators, and more for system administrators than for end-users.*

H2.2 *Projects developed on topics dealing for developers and system administrators are more successful than projects on topics for end-users. As well, d) there are more projects on topics for developers than for system administrators, and more for system administrators than for end-users.*

Competency marshalling

A project to be successful has to involve participants with required competencies. The ability of a project to attract competent individuals depends on the shared culture and climate, characteristics of the application, the abilities and charisma of the project leaders, etc. As one measure, we considered whether a project had administrators who had been peer-rated or ranked (RATED and RANKED). A well-regarded developer should be more able to attract developers, addressing competency rallying. We also measured the outcome of these factors, that is, the extent to which projects seemed to have successfully marshalled competencies. To measure this outcome, we used the number of developers and administrators taking part in the project (DEVEL and ADMIN), although this measure is noisy in that it also reflects differences in the scale of the projects.

We propose the following specific hypotheses:

H3.1 *Projects with more developers are more successful (i.e., a) more active, b) in more advanced states of development and c) more used).*

H3.2 *Projects with more highly ranked or rated project administrators are more successful (i.e., a) more active, b) in more advanced states of development and c) more used).*

Management of short-term cooperation

The lack of information on intrinsic characteristics of the software (such as its modularity) or adopted coordination mechanisms makes it hard to formulate testable hypotheses on the concepts proposed in proposition P4. For tests using development status as a measure of success, we included the variable ACTIVITY as a measure of the intensity of the cooperation. As well, the presence of rated and ranked developers as mentioned above might be considered as measures of the managerial ability of the project leader. However,

further research on the details of coordination in OSS software development is necessary to properly test the concepts proposed in the proposition P4.

We propose the following specific hypothesis:

H4. Projects with higher activity are in a more advanced state of development.

Control variables

Finally, we identified one variable that might provide alternative explanations for the success variables, namely project lifespan. Projects might be more successful (i.e., be more used, in more advanced stages of development and more active) simply because they have had more time to mature. Therefore, we included project lifespan (LIFESPAN) as a control variable.

The resulting model is shown graphically in Figure 4.

Insert Figure 4 about here

4. Methodology and results

In this section we discuss the method used to test the hypotheses discussed above and the results of our analyses.

4.1 Data collection and pre-processing

Data were collected using a Web spider to retrieve pages from SourceForge for each project and for each administrator. Pages for projects were collected on 29 November 2000 and for statistics on 18 January 2001. Pages for administrators were collected on 3 February 2001. The resulting HTML files were parsed to extract the relevant data. Data were obtained on a total of 11,959 projects and 10,323 administrators. However, only 7,477 of the projects

included all the project classification data needed for further analysis. As a result, our analysis is based only on these projects. Descriptive statistics for the variables described above are reported in Table 3.

Insert about Table 3 here

Data were first explored variable by variable. Some of the variables were highly skewed (e.g., number of downloads, page views, developers and administrators and count of projects using a given programming language) and so a logarithmic transformation was applied to these variables to correct the skew. Percentile rank data was transformed using an inverse-sine-square-root transformation, which renders more nearly normally distributed a uniformly distributed variable such as a proportion.

For the two categorical variables, TOPIC and AUDIENCE, we created dummy variables per value. For AUDIENCE, we created 4 dummy variables, one for each audience the project was intended to serve (end-user, system administrator, software developer and other). Topic required special handling because there were 162 topics, as shown in Table 4, far too many to analyze individually. To develop a manageable number of dummy variables, we merged related topics together. The list of topics in SourceForge is hierarchical, so each topic was first recoded to the appropriate top-level topic (e.g., the topic "Web browsers" became "Internet"). Second, three less common and theoretically related first-level topics were merged. These are shown in italics in Table 4. A dummy variable was created for each remaining first-level topic (shown in bold in Table 4), and the projects coded appropriately.

Insert Table 4 about here

Programming language, topic and intended audience created additional problems because a project might list up to three programming languages, intended audiences or topics.

For projects that listed several programming languages, we used the average of the scores for each programming language. Projects that listed multiple audiences and topics had non-zero values for more than one dummy variable, weighted by the total number of audience or topics addressed by the project. For example, if a project addressed both systems administrators and software developers, it was given a score of 1/2 for the two appropriate dummy audience variables. These projects experience the average effects of the different languages, topics and audiences they address.

We initially planned to calculate average peer ratings and rankings for each project's administrators. However, we found that only 646 projects had administrators that had been rated at all, and only 66 had administrators who were peer ranked. Examining the rating process, we concluded that rated and ranked developers are those who are well known and hypothesized that having such a well-known administrator should make the project more successful. Therefore, we decided to compare projects with and without rated and ranked administrators. We therefore simply created two dummy variables (DRATED and DRANKED) that indicated whether a project had a ranked or a rated administrator.

4.2 Data Analysis

Most hypotheses were tested using multiple regression. Three regressions were run, one for each of the three success variables defined above. To test the hypotheses related to the activity of the projects (*H#.a*), transformed rank activity percentile was used as the dependent variable. The resulting regression equation is shown in Equation A.

(A)
$$ACTIVITY = \beta_A + \beta_{A1}PLPOP + \sum \beta_{A2i}DAUDIENCE_i + \sum \beta_{A3i}DTOPIC_i + \beta_{A4}DEVEL + \beta_{A5}ADMIN + \beta_{A6}DRATED + \beta_{A7}DRANKED + \beta_{A8}LIFESPAN$$

To test the hypotheses related to the development status (*H#. b*), we used the regression equation shown in Equation B[3].

(B) $$STATUS = \beta_B + \beta_{B1}PLPOP + \sum \beta_{B2i}DAUDIENCE_i + \sum \beta_{B3i}DTOPIC_i + \beta_{B4}DEVEL + \beta_{B5}ADMIN + \beta_{B6}DRATED + \beta_{B7}DRANKED + \beta_{B8}LIFESPAN + \beta_{B9}ACTIVITY$$

Finally, to test the hypotheses related to the use of the software (*H#. c*) a multiple regression model was adopted using software use (i.e., downloads and page views) as the dependent variable, as shown in Equation C.

(C) $$USE = \beta_C + \beta_{C1}PLPOP + \sum \beta_{C2i}DAUDIENCE_i + \sum \beta_{C3i}DTOPIC_i + \beta_{C4}DEVEL + \beta_{C5}ADMIN + \beta_{C6}DRATED + \beta_{C7}DRANKED + \beta_{C8}LIFESPAN$$

The coefficients for the models were estimated using the stepwise linear regression in SPSS. The criterion for entering variables was $p < 0.05$, and for removing variables, $p > 0.10$. No variables were removed, and only the results of the final model are reported.

Hypotheses H2.1d and H2.2d were tested by examining the number of projects on each topic and for each intended audience, as shown in Figure 5 and Table 5 and Figure 6 and Table 6. Note that because projects can have multiple audiences and topics, the sum of the number of projects in each category is greater than the actual total number of projects.

Insert Table 5 and Figure 5
and Table 6 and Figure 6

[3] Because STATUS is an ordinal variable, we compared the results for model B to the results from SPSS's ordinal regression command. However, to use this feature, we had to replace the dummy variables for topic and audience with two categorical variables, coding only those cases where a project had a single topic or audience, thus limiting this analysis. Since the results from the ordinal regression are largely consistent with the results of the conventional regression, and the conventional regression is more familiar, we do not discuss the results of the ordinal regression further.

4.3 Results

The R^2 values for the various models are given in Table 7 and the estimated coefficients for the various models in Table 8 through Table 10. For ease of interpretation, the tables of regression coefficients have been arranged by hypotheses.

Insert Table 7 to Table 10

Model A is the regression for project activity. The R^2 for this regression was .476, which is satisfactory. The coefficients, shown in Table 8, provide support for hypotheses H1, H3.1, and H3.2 (as indicated in the leftmost column). This means that projects that adopt more used programming languages, in which more developers take part and for which a peer ratings for system administrators exist appear to have a higher activity as reflected in the project's rank percentile value. The standardized coefficients help compare the relative importance of the different parameters. These values indicate that project life span has the largest effect (older projects are more active). Of the hypothesized predictors, the number of developers seems to have the most impact on activity, with number of administrators and the administrator being peer rated being well behind. Programming language has only a weak effect.

Models B is the regression for project status. The R^2 values associated with these models are considerably lower, at .127, meaning that only the model explains a small amount of the overall variance. The lower R^2 likely reflects the crudeness of development status (a 6-level ordinal variable) as an outcome measure. Yet, as illustrated in Table 9, the values of the related regression coefficients are consistent with hypotheses H2.2, H3.2 and H4. Specifically, projects on the topic of business/office are more likely to be at lower stages of development whereas projects related to software development are associated with a more

mature development status. As well, projects with peer rated administrators are likely to be in a more mature state, as are more active projects. The evidence here offers mixed support for H2.1, since projects intended for system administrators are more advanced, but projects for software developers are less advanced, the later contrary to hypothesis. On the other hand, the results do not speak to H1 and contradict H3.1—a project with more developers is somewhat less likely to be in an advanced state of development. In this case, ACTIVITY has the largest standardized coefficient. The coefficient for project lifespan is also significant, meaning that older projects are more likely to be in advanced states of development.

Finally, Model C, the regression for software use, has an R^2 value of .129, which is also low—again, the model explains little of the variance in software use. However, the coefficients, shown in Table 10, support hypothesis H1: the adoption of more used programming language seem to be associated with more used software. Hypotheses H2.2 is supported—software dealing with office/business topics have lower popularity (measured in terms of use), while software for software development and systems are more used. Hypothesis H2.1 though is contradicted—end user applications are more used, not less. H3.2 is supported—software programs developed in projects with administrators who are peer rated and ranked are more likely to have higher use—while hypothesis H3.1 has mixed support—more developers make a project more used, but more administrators make projects less used. Considering the standardized coefficients, DEVELOPERS is the most dominant factor. In contrast to the other regressions, project lifespan was not entered, meaning that older projects are not more likely to be used.

We turn now to hypotheses H2.1d and H2.2d. The data in Table 5 contradict hypothesis H2.1d: there are about the same number of projects for software developers and for end users, and fewer for systems administrators. The data in Table 6 clearly support hypothesis H2.2d: the majority of projects report software development or systems related

topics, and only a small minority address office/business topics or more specific topical areas. A summary of the empirical support for the hypotheses is presented in Table 11.

Insert Table 11 about here

5. Discussion

In this section, we will discuss the hypotheses tested above. Hypothesis 1, that use of more popular languages is related to success, was supported by the data. We interpret this finding as support for proposition P1 of our theory, that the availability of competencies is a precondition for the success of projects undertaken by OSS.

Hypothesis 2.1 was not supported. Contrary to our expectation, there were almost as many projects intended for end-users as for software developers, and these projects were in more advanced states of development and were more used, not less. We believe that the lack of support for this hypothesis is because developers' use of the SourceForge classification of "end-user" projects does not imply a project intended for non-developers as we assumed in formulating our hypotheses. Rather, it seems to be used to describe a desktop application, regardless of intended use. In retrospect then, H2.1 was not a good test of proposition P2.

Hypothesis 2.2, on the contrary, was supported. There are fewer projects for business or specialized topics, and these projects tend to be in earlier stages of development and less used (though the amount of the variance explained is small). In summary then, our data offers qualified support for proposition P2 of our theory, that the ability of the project participants to identify and understand the market opportunity is important to the success of projects. This finding implies that OSS, with its reliance on self-interested developers, may be less well suited for developing applications that address problems that large numbers of developers tend not to face, such as business or scientific applications.

Hypothesis 3.1 had mixed support. The number of developers was associated with increased project activity and software use, but also with less advanced states of development. It seems reasonable that more developers would lead to more activity. An explanation for the second finding is that less advanced projects need more work, and therefore more developers. As projects move into advanced stages and the work is primarily maintenance, developers move on to new projects, thus explaining the reduced number of developers on these projects. In any event, H3.1 was only a weak test of proposition P3.

Hypothesis H3.2, on the contrary, was strongly supported. Projects that have well-known developers, that is, developers who are peer-rated or ranked, are more active, in more advanced states of development and more used. This result can be interpreted in two ways. This result may reflect the cult of personality that surrounds well-known developers, and suggests the importance of these figures in obtaining support. Contrariwise, it may be that having attracted a lot of contributors makes a project administrator well known. In summary, we believe the data is consistent with proposition P3 of our theory, that ability to marshal necessary competencies is a key for project success.

Finally, hypothesis H4 is supported in the model where it was tested. Projects with more activity tend to be in more advanced stages of development. In some ways, this result seems intuitive, as activity is needed to advance a project. However, this finding contradicts our explanation for the fewer number of developers in these projects, suggesting that a more detailed examination is needed. Hypothesis H4 was only an indirect test of proposition P4, but the evidence does seem consistent with the theory. As well, the value of well-known and presumably more experienced administrators (H3.2) also adds weight to this proposition.

Threats to validity and limitations of the study

Before we conclude, we will briefly discuss possible limitations of our study. First, many of our measures are likely quite unreliable (that is, each includes a good deal of noise in addition to the construct of interest). In particular, much of the data is self-reported, again decreasing reliability. As a result, our results above may understate the true strength of the relationships. Indeed, two of the regressions explained only a small percentage of the total variance. A related concern with negative results is that they may be due to a lack of statistical power in the tests. However, in our study, the sample size is large, which helps obviate this concern.

Second, our use of secondary data required us to rely on a given and non-modifiable set of variables to operationalize the theoretical concepts. Our reliance on this approach represents the main limitation of the study, since the data may not adequately measure the constructs of our theory, thus posing a threat to validity. This threat is particularly problematic for our outcome measure, project success. We addressed this threat in this study by considering multiple outcome measures, but we continue to search for better measures.

Part of the problem in obtaining data is that there is as yet no agreed set of measures for software development projects that seems adequate to describe these new endeavours. For example, lines-of-code is a common measure of project size or programmer productivity, but the relevance of this measure for OSS, which may incorporate contributions from diverse source, is not clear. Unlike other fields where secondary data is more common, software development lacks a set of professional data collection agencies—there is no equivalent to the Census or Bureau of Labour Statistics.

A third concern is that the sample may be biased. Since SourceForge only started on 3 November 1999, older and better established projects are less likely to be included (some

have subsequently moved to the system). For example, the main Linux and Apache developers do not use SourceForge, though some smaller supporting projects do. Some of our findings could be due to this selection bias. On the other hand, as we noted above, studying just large successful projects introduces the opposite bias. It is difficult to properly assess these concerns about the representativeness of the sample, as it impossible to determine the universe of OSS projects to which we hope to generalize. However, the popularity of SourceForge suggests that the sample is extensive, even if not complete, which would minimize this threat to validity.

Finally, our analysis considers only one level of effects. It would be interesting to consider the factors that enable competency rallying or short-term cooperation (e.g., the importance of community prestige and loyalty to a key project organizer), but unfortunately, such factors are difficult to operationalize. Future research might attempt to measure these factors and correlate them to the objective data collected by SourceForge.

6. Conclusions

In this paper, the success of OSS projects and programs has been investigated. Based on a review of the literature, we argued that OSS communities can be analyzed as virtual organizations. Accordingly, the theory of competency rallying was adopted to identify elements necessary for a project to be successful. This theory suggests that shared competencies and needs, the possibility to marshal many developers, project administrators' managerial skills and charisma, and the adoption of *ad hoc* coordination mechanisms are crucial for success.

Based on this framework, four propositions were formulated. By adopting the variables used in the SourceForge project to classify and report on projects, the main concepts proposed in the propositions were operationalized and a set of testable hypotheses formulated.

These hypotheses were tested using data regarding the OSS projects supported by SourceForge. The obtained results mostly confirm the proposed framework.

Our study has some implications for those organizing OSS projects. First, the availability of competencies is a factor in the success of projects, suggesting the choice of a more widely known programming language. Second, it is important for developers to be able to recognize the needs of customers of the project. Our analysis shows that at present, projects with topics related to systems development and administration are more successful. Most analyses of OSS have stressed the contributions of programmers, but we suggest that for many projects it will be important to consider how input can be obtained from non-programmers. Finally, our analysis shows that projects with well-known administrators are more successful, which we interpret as indicating the importance of being able to attract support for a project. Less well-known developers starting a project will have to pay particular attention to how to attract contributors, a problem that has been the focus of much of the research on OSS.

We have only scratched the surface in using the self-collecting data from this system. Since these projects are mostly coordinated via the Internet, in principle all project interactions should be observable. In particular, analyses of the detailed day-to-day project statistics shown in Figure 2, bug reports, support logs and other project information should provide a fascinating window into project dynamics, particularly the question of how the contributions of participants is coordinated. The definition and collection of an adequate measure of the success and the formulation of testable hypotheses on factors such as the importance of adopted coordination mechanisms should be a goal of further research.

Despite the limitations of our study, the collected information and the related analysis provide a unique quantitative picture of OSS development. The evidence we have collected

suggests that the competency-rallying framework represents a useful guide to developers interested in the OSS model and for the topic of virtual organizations more generally.

Acknowledgements

This paper has benefited from comments from Martha Garcia-Murillo, Junseok Hwang, Ian MacInnes, Jian Qin, Dmitri Roussinov, Steve Sawyer, Zixiang (Alex) Tan and Marie Williams.

References

[1] M. W. Godfrey and Q. Tu, "Evolution in open source software: A case study," presented at The 2000 International Conference on Software Maintenance, San Jose, California, 2000.

[2] G. Moody, *Rebel code—Inside Linux and the open source movement*. Cambridge, MA: Perseus Publishing, 2001.

[3] D. Cubranic, "Open-source software development," presented at 2nd Workshop on Software Engineering over the Internet, Los Angeles, 1999.

[4] P. Wayner, *Free For All*. New York: HarperCollins, 2000.

[5] M. Sawhney and E. Prandelli, "Communities of creation: managing distributed innovation in turbulent markets," *California Management Review*, vol. 42, pp. 24–54, 2000.

[6] E. S. Raymond, "The cathedral and the bazaar," *First Monday*, vol. 3, 1998.

[7] C. Browne, "Linux and decentralized development," *First Monday*, vol. 3, 1998.

[8] C. Connell, "Open Source Projects Manage Themselves? Dream On,", vol. 2000, n.d.

[9] N. Bezroukov, "Open source software development as a special type of academic research (critique of vulgar raymondism)," *First Monday*, vol. 4, 1999.

[10] B. Katzy and K. Crowston, "A process theory of competency rallying in engineering projects," in *Submitted to IEEE Transactions on Engineering Management*, 2000.

[11] T. Bollinger and P. Beckman, "Linux on the move," *IEEE Software*, vol. 16, pp. 30–35, 1999.

[12] C. Prasad and Ganesh, "A hard look at Linux's claimed strengths…,",, 1999.

[13] V. Valloppillil, "Halloween I: Open Source Software,"., 1998.

[14] V. Valloppillil and J. Cohen, "Halloween II: Linux OS Competitive Analysis,"., 1998.

[15] J. Hallen, A. Hammarqvist, F. Juhlin, and A. Chrigstrom, "Linux in the workplace," *IEEE Software*, vol. 16, pp. 52–57, 1999.

[16] B. Pfaff, "Society and open source: Why open source software is better for society than proprietary closed source software,"., 1998.

[17] E. Leibovitch, "The business case for Linux," *IEEE Software*, vol. 16, pp. 40–44, 1999.

[18] N. Bezroukov, "A second look at the Cathedral and the Bazaar," *First Monday*, vol. 4, 1999.

[19] R. L. Glass, "Of open source, Linux, …and hype," *IEEE Software*, vol. 16, pp. 126–128, 1999.

[20] R. Young, "How Red Hat Software stumbled across a new economy model and helped improve an industry," in *Open sources: voices from the open source revolution*, C. Di Bona, S. Ockman, and M. Stone, Eds. San Francisco: O'Reilly, 1999.

[21] B. Behlendorf, "Open source as a business strategy," in *Open sources: Voices from the open source revolution*, D. B. C., O. S., and S. M., Eds. San Francisco: O'Reilly, 1999.

[22] M. L. Markus, B. Manville, and E. C. Agres, "What makes a virtual organization work?," *Sloan Management Review*, vol. 42, pp. 13–26, 2000.

[23] T. W. Malone, J. Yates, and R. I. Benjamin, "Electronic markets and electronic hierarchies," *Communications of the ACM*, vol. 30, pp. 484–497, 1987.

[24] G. Lorenzoni and A. Lipparini, "The leveraging of interfirm relationships as a distinctive organizational capability: A longitudinal study," *Strategic Management Journal*, vol. 20, pp. 317–338, 1999.

[25] H. W. Chesbrough and D. J. Teece, "When is virtual virtuous? Integrated virtual alliances organizing for innovation," *Harvard Business Review*, vol. 74, pp. 65-73, 1996.

[26] T. J. Strader, F. Lin, and M. J. Shaw, "Information infrastructure for electronic virtual organization management," *Decision Support Systems*, vol. 23, pp. 75–94, 1998.

[27] R. Kraut, C. Steinfield, A. P. Chan, B. Butler, and A. Hoag, "Coordination and virtualization: The role of electronic networks and personal relationships," *Organization Science*, vol. 10, pp. 722–740, 1999.

[28] B. M. Wiesenfeld, S. Raghuram, and R. Garud, "Communication patters as determinants of organizational identification in a virtual organization," *Organization Science*, vol. 10, pp. 777–790, 1999.

[29] R. E. Miles and C. C. Snow, "Organizations: New concepts for new forms," *California Management Review*, vol. 28, pp. 62-73, 1986.

[30] C. Handy, "Trust and virtual organization," *Harvard Business Review*, vol. 73, pp. 40-50, 1995.

[31] A. Mowshowitz, "Virtual organization," *Communications of the ACM*, vol. 40, pp. 30-37, 1997.

[32] D. M. Upton and A. McAfee, "The real virtual factory," *Harvard Business Review*, vol. 74, pp. 123-133, 1996.

[33] M. K. Ahuju and C. K., "Network structure in virtual organizations," *Journal of Computer-Mediated Communication*, vol. 3, 1998.

[34] E. S. Raymond, "Interview: Linux and open source success," *IEEE Software*, vol. 16, pp. 85–89, 1999.

[35] B. R. Katzy, G. Schuh, and K. Millarg, "Die virtuelle Fabrik - Produzieren in Netzwerken," *Technische Rundschau*, pp. 30-34, 1996.

[36] D. J. Teece, G. Pisano, and A. Shuen, "Dynamic capabilities and strategic management," *Strategic Management Journal*, vol. 20, pp. 509–533, 1997.

[37] M. S. Krishnan, "The role of team factors in software cost and quality: An empirical analysis," *Information Technology & People*, vol. 11, pp. 20–35, 1998.

[38] S. Faraj and L. Sproull, "Coordinating Expertise in Software Development Teams," *Management Science*, vol. 46, pp. 1554–1568, 2000.

[39] P. Vixie, "Software engineering," in *Open sources: Voices from the open source revolution*, C. Di Bona, S. Ockman, and M. Stone, Eds. San Francisco: O'Reilly, 1999.

[40] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Communications of the ACM*, vol. 38, pp. 69–81, 1995.

[41] J. Ousterhout, "Free Software needs profit," *Communications of the ACM*, vol. 42, pp. 44–45, 1999.

[42] S. Koch and G. Schneider, "Results from Software engineering research into Open source development projects using public data," Wirtschaftuniversitat Wien, Austria, Working Paper #22, 2000.

[43] J. Y. Moon and L. Sproull, "Essence of distributed work: The case of Linux kernel," *First Monday*, vol. 5, 2000.

[44] T. O'Reilly, "Lessons from open source software development," *Communications of the ACM*, vol. 42, pp. 33–37, 1999.

[45] E. S. Andersen and M. Valente, "The two software cultures and the evolutionary economic simulation,"., 1999.

[46] T. Bollinger, R. Nelson, K. M. Self, and S. J. Turnbull, "Open source methods: Peering through the clutter," *IEEE Software*, vol. 16, pp. 30–35, 1999.

[47] J. de Goyeneche and E. A. F. de Sousa, "Loadable kernel modules," *IEEE Software*, vol. 16, pp. 65–71, 1999.

[48] J. D. Thompson, *Organizations in Action: Social Science Bases of Administrative Theory*. New York: McGraw-Hill, 1967.

[49] J. R. Galbraith, *Designing Complex Organizations*. Reading, MA: Addison-Wesley, 1973.

Tables and Figures

SourceForge - Summary

Summary | Admin | Home Page | Forums | Bugs | Support | Patches | Lists | Tasks | Docs | Surveys | News | Files |

This is the SourceForge entry for the SourceForge site itself. We are using our own bug tracker and message forums for feedback on the site.

- Development Status: 5 - Production/Stable
- Environment: Web Environment
- Intended Audience: Developers, End Users/Desktop
- License: GNU General Public License (GPL)
- Operating System: OS Independent
- Programming Language: PHP
- Topic: Dynamic Content

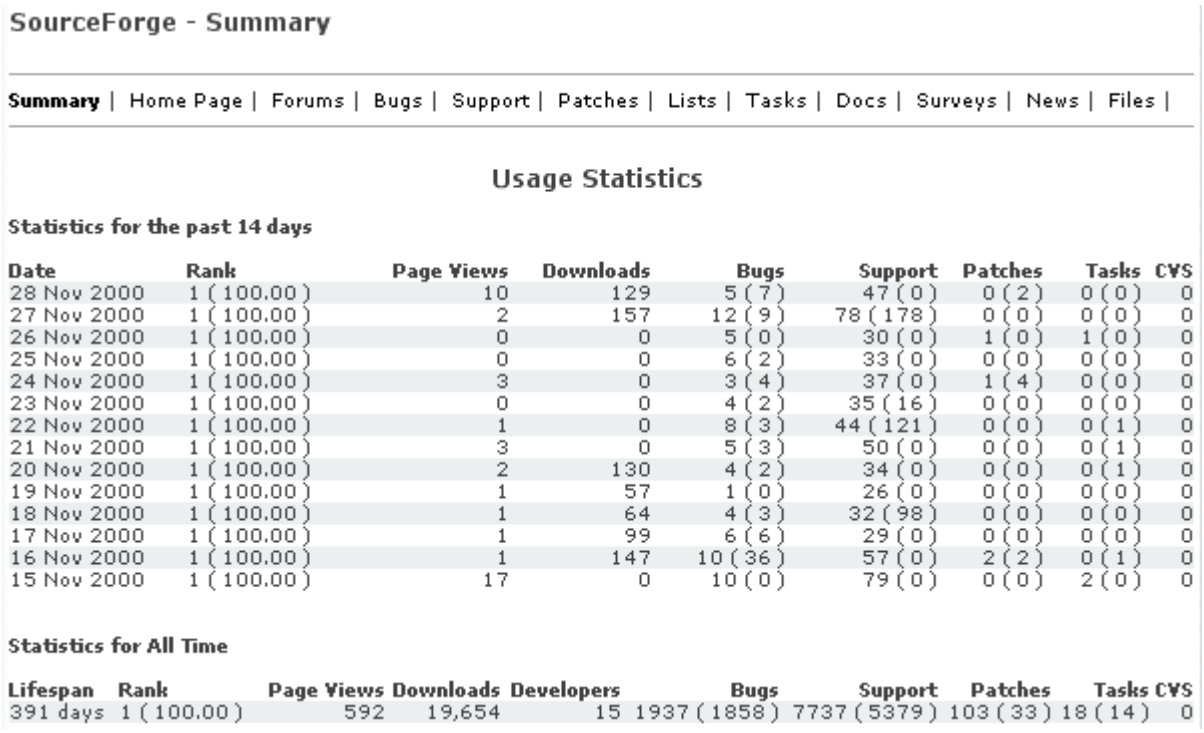
Registered: 1999-Nov-03 16:00
Activity Percentile: 95.8247%
View project activity statistics

Developer Info

Project Admins:
noworrys
mrzenn
pfalcon
blkadder
fearless
shadocat
fusion94
precision
baddog
bigdisk
dbressler
lissy
dtype
pat
nightcrew

Developers:
21 [View Members]

Figure 1. Example of a SourceForge (http://sourceforge.net/) project summary.



SourceForge - Summary

Summary | Home Page | Forums | Bugs | Support | Patches | Lists | Tasks | Docs | Surveys | News | Files |

Usage Statistics

Statistics for the past 14 days

Date	Rank	Page Views	Downloads	Bugs	Support	Patches	Tasks	CVS
28 Nov 2000	1 (100.00)	10	129	5 (7)	47 (0)	0 (2)	0 (0)	0
27 Nov 2000	1 (100.00)	2	157	12 (9)	78 (178)	0 (0)	0 (0)	0
26 Nov 2000	1 (100.00)	0	0	5 (0)	30 (0)	1 (0)	1 (0)	0
25 Nov 2000	1 (100.00)	0	0	6 (2)	33 (0)	0 (0)	0 (0)	0
24 Nov 2000	1 (100.00)	3	0	3 (4)	37 (0)	1 (4)	0 (0)	0
23 Nov 2000	1 (100.00)	0	0	4 (2)	35 (16)	0 (0)	0 (0)	0
22 Nov 2000	1 (100.00)	1	0	8 (3)	44 (121)	0 (0)	0 (1)	0
21 Nov 2000	1 (100.00)	3	0	5 (3)	50 (0)	0 (0)	0 (1)	0
20 Nov 2000	1 (100.00)	2	130	4 (2)	34 (0)	0 (0)	0 (1)	0
19 Nov 2000	1 (100.00)	1	57	1 (0)	26 (0)	0 (0)	0 (0)	0
18 Nov 2000	1 (100.00)	1	64	4 (3)	32 (98)	0 (0)	0 (0)	0
17 Nov 2000	1 (100.00)	1	99	6 (6)	29 (0)	0 (0)	0 (0)	0
16 Nov 2000	1 (100.00)	1	147	10 (36)	57 (0)	2 (2)	0 (1)	0
15 Nov 2000	1 (100.00)	17	0	10 (0)	79 (0)	0 (0)	2 (0)	0

Statistics for All Time

Lifespan	Rank	Page Views	Downloads	Developers	Bugs	Support	Patches	Tasks	CVS
391 days	1 (100.00)	592	19,654	15	1937 (1858)	7737 (5379)	103 (33)	18 (14)	0

Figure 2. Example of SourceForge (http://sourceforge.net/) project use statistics.

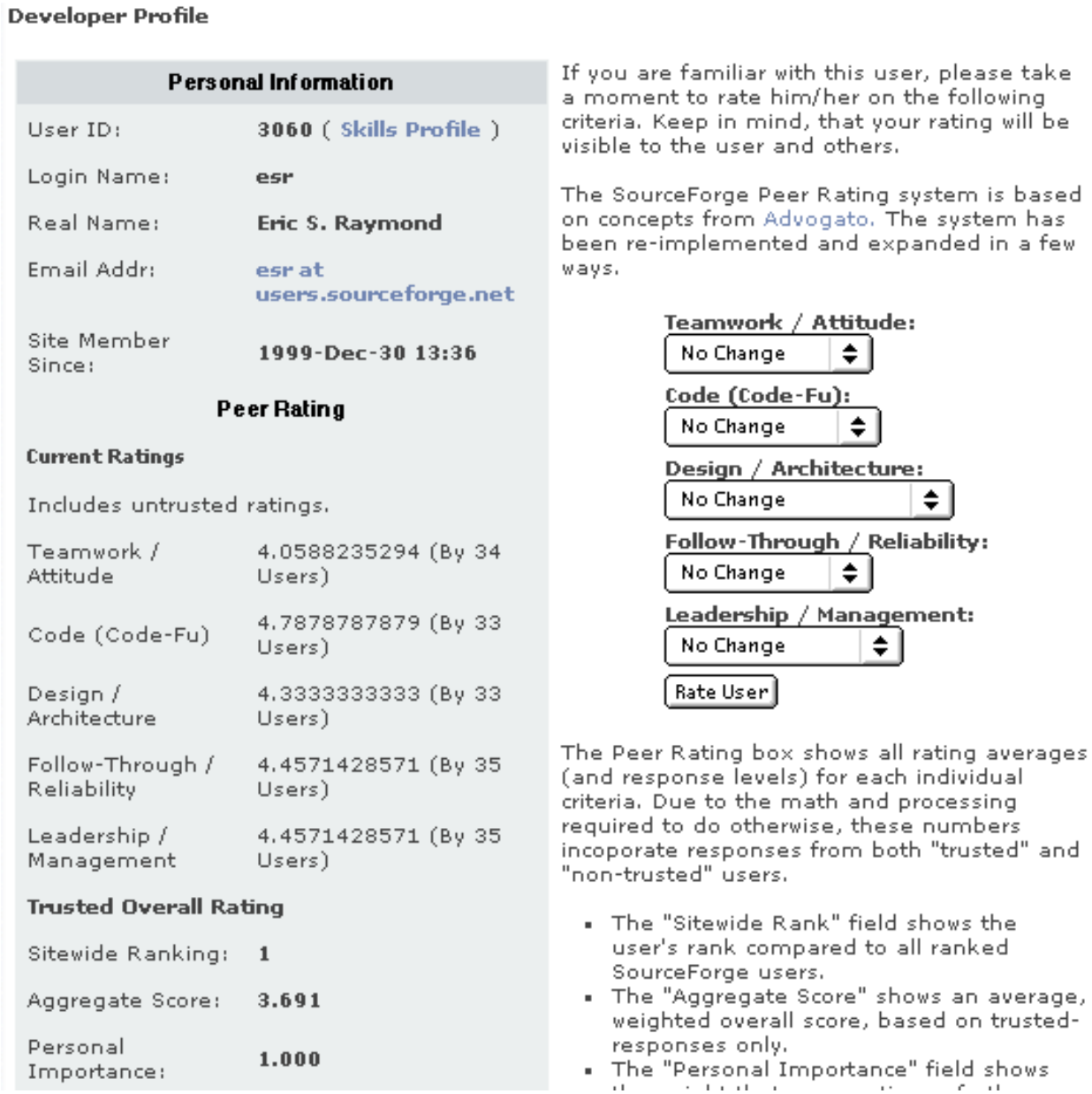


Figure 3. Example of a SourceForge (http://sourceforge.net/) developer profile, showing Peer Ratings and Overall Ratings, with ranking.

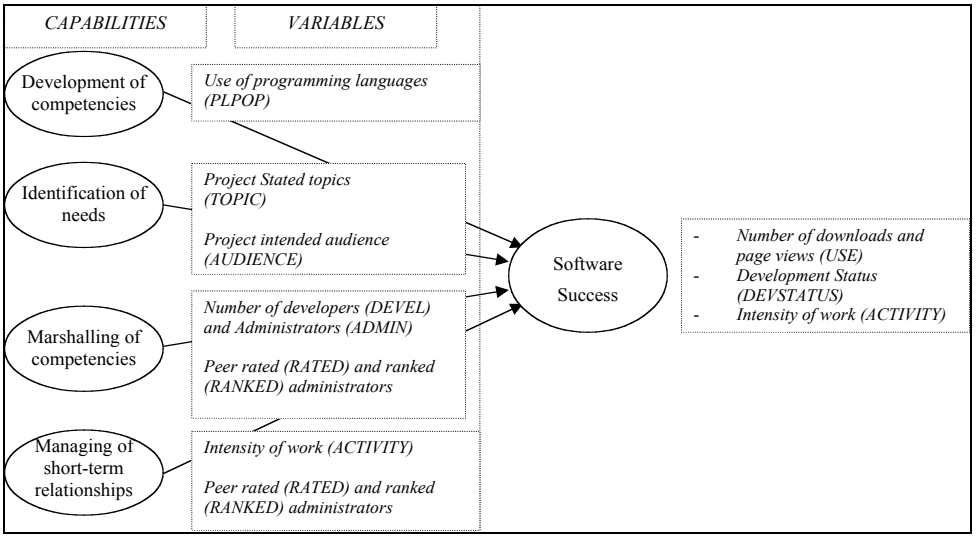


Figure 4. The operationalization of the CR framework as applied to OSS projects: Capabilities and adopted variables.

Table 1. Description of variables used in study.

Variable	Definition	Notes
ACTIVITY	Level of project activity	$\sin^{-1}$ percentile rank of activity
DEVSTATUS	Project development status	6 level ordinal variables
USE	Degree of end-user interest in project software	Scale composed of ln downloads and ln pageviews
PLPOP	Measure of use of programming language used	Average of ln count of projects using that language
DAUDIENCEi	Dummies for intended audience for project	4 variables
DTOPICi	Dummies for topic of project	9 variables
DEVEL	Number of developers on project	ln transform
ADMIN	Number of administrators on project	ln transform
DRATED	Dummy for project has a peer rated administrator	
DRANKED	Dummy for project has a peer ranked administrator	
LIFESPAN	Current age of the project, in days	

Table 2. CR Framework, Propositions and Hypotheses.

CR theory steps	Propositions and Hypotheses
Competencies development and maintenance	P1 The more available the required competencies, the more successful the OSS project.
	H1 Projects using more common programming languages are more successful (i.e. a) more active, b) in more advanced states of development and c) more used).
Recognition of market opportunity	P2 The more readily developers can recognize the needs and problems addressed by the project, the more successful the OSS project.
	H2.1 Projects intended for developers are successful than projects developed for system administrators, which are successful than projects developed for end-users. As well, d) there are more projects intended for developers than for system administrators, and more for system administrators than for end-users.
	H2.2 Projects developed on topics dealing for developers and system administrators are successful than projects on topics for end-users. As well, d) there are more projects on topics for developers than for system administrators, and more for system administrators than for end-users.
Marshalling competencies	P3 The more quickly and accurately competencies can be marshalled, the more successful the OSS project.
	H3.1 Projects with more developers are more successful (i.e. a) more active, b) in a more advanced state of development, and c) more used).
	H3.2 Projects with more highly ranked or rated project administrators are more successful (i.e. a) more active, b) in a more advanced state of development, and c) more used).
Short term cooperation	P4 The greater the ability to manage short-term cooperation, the more successful the OSS project.
	H4 Projects with higher activity are in a more advanced state of development

Table 3. Descriptive statistics for variables.

	N	Minimum	Maximum	Mean	Std. Deviation
ACTIVITY	7477	0	1.57	.8279	.3061
Ln Page Views	7477	0	13.79	5.0756	2.8164
Ln Downloads	7477	0	13.23	2.4527	3.0642
USE	7474	–1.30	3.28	0	.8800
PLPOP	7097	0	7.89	7.0102	.9689
DEVEL	7477	0	4.43	.9061	.4143
ADMIN	7477	0	3.04	.8151	.2588
LIFESPAN	7477	2	391	153.63	98.29

Table 4. Hierarchy of topics represented by SourceForge projects.

- **Communications**
  - BBS
  - Chat
    - ICQ
    - Internet Relay Chat
    - Unix Talk
    - AOL Instant Messenger
  - Conferencing
  - Email
    - Email Clients (MUA)
    - Filters
    - Mailing List Servers
    - Mail Transport Agents
    - Post-Office
    - POP3
    - IMAP
  - Fax
  - FIDO
  - File Sharing
    - Napster
  - Ham Radio
  - Internet Phone
  - Telephony
  - Usenet News
- **Games/Entertainment**
  - First Person Shooters
  - Multi-User Dungeons (MUD)
  - Puzzle Games
  - Real Time Strategy
  - Role-Playing
  - Simulation
  - Turn Based Strategy
- **Internet**
  - File Transfer Protocol (FTP)
  - Finger
  - Log Analysis
  - Name Service (DNS)
  - WAP
  - WWW/HTTP
    - Browsers
    - Dynamic Content
    - CGI Tools/Libraries
    - Message Boards
    - Page Counters
    - HTTP Servers
    - Indexing/Search
    - Site Management
    - Link Checking
- **Office/Business**
  - Financial
    - Accounting
    - Investment
    - Point-Of-Sale
    - Spreadsheet
  - Office Suites
  - Scheduling
- **Topical**
  - *Scientific/Engineering*
    - Artificial Intelligence
    - Astronomy
    - Bio-Informatics
    - Electronic Design Automation (EDA)
    - Human Machine Interfaces
    - Mathematics
    - Medical Science Apps.
    - Visualization
  - *Education*
    - Computer Aided Instruction (CAI)
    - Testing
  - *Religion*
- **Multimedia**
  - Graphics
    - 3D Modeling
    - 3D Rendering
    - Capture
      - Scanners
      - Digital Camera
      - Screen Capture
    - Editors
      - Graphics Conversion
      - Vector-Based
      - Raster-Based
      - Presentation
      - Viewers
  - Sound/Audio
    - Analysis
    - CD Audio
      - CD Playing
      - CD Ripping
    - Capture/Recording
    - Conversion
    - Editors
    - MIDI
    - Mixers
    - Players
      - MP3
    - Sound Synthesis
    - Speech
  - Video
    - Capture
    - Conversion
    - Display
    - Non-Linear Editor

Software Development
- Build Tools
- Code Generators
- Compilers
- Debuggers
- Interpreters
- Object Brokering
  - CORBA
- Version Control
  - CVS
  - RCS
  - SCCS
- Database
  - Database Engines/Servers
  - Front-Ends
- Text Editors
  - Documentation
  - Emacs
  - Integrated Development Environments (IDE)
  - Word Processors

System
- Archiving
  - Backup
  - Packaging
  - Compression
- Benchmark
- Boot
  - Init
- Clustering/Distributed Networks
- Emulators
- Filesystems
- Hardware
- Installation/Setup
- Logging
- Networking
  - Firewalls
  - Monitoring
- Operating System Kernels
  - Linux
  - BSD
  - GNU Hurd
- Power (UPS)
  - Hardware Watchdog
- Software Distribution
- Systems Administration
- Desktop Environment
  - Gnome
  - K Desktop Environment (KDE)
  - Themes
  - Screen Savers
  - Window Managers
  - Enlightenment
- Printing
- Security
  - Cryptography
- Terminals
  - Serial
  - Terminal Emulators/X Terminals
  - Telnet

Other/Nonlisted Topic

Note: For analysis, topics were recorded to the appropriate "top-level" topic, shown in the table in bold. "Topical" was formed by merging three topics that were top-level in SourceForge, shown here in italics: Scientific/Engineering, Education and Religion.

Table 5. Frequency of projects by intended audience.

Audience	Count	
End user	4146	35%
Developer	4596	38%
Systems administrators	1929	16%
Other	1057	9%
Missing	216	2%
Total	7477	

Note: Counts sum to more than total number of projects because projects may have multiple intended audiences. Percentages are of sum of counts.

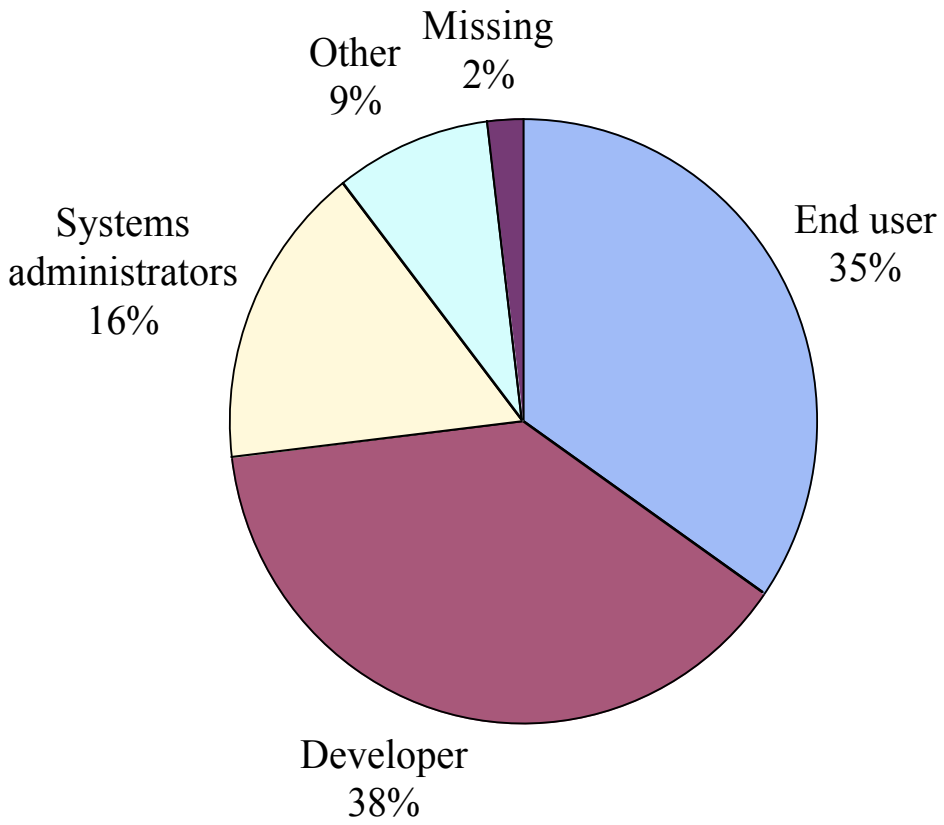


Figure 5. Pie chart of distribution of projects by intended audience.

Table 6. Frequency of project with listed topics.

Topic	Count	
Topical	749	8%
Games	1070	11%
Software development	1728	17%
Other	258	3%
Systems	1788	18%
Multimedia	984	10%
Communications	1181	12%
Office/business	365	4%
Internet	1603	16%
Missing	207	2%
Total	**7477**	

Note: Counts sum to more than total number of projects because projects may have multiple topics. Percentages are of sum of counts.

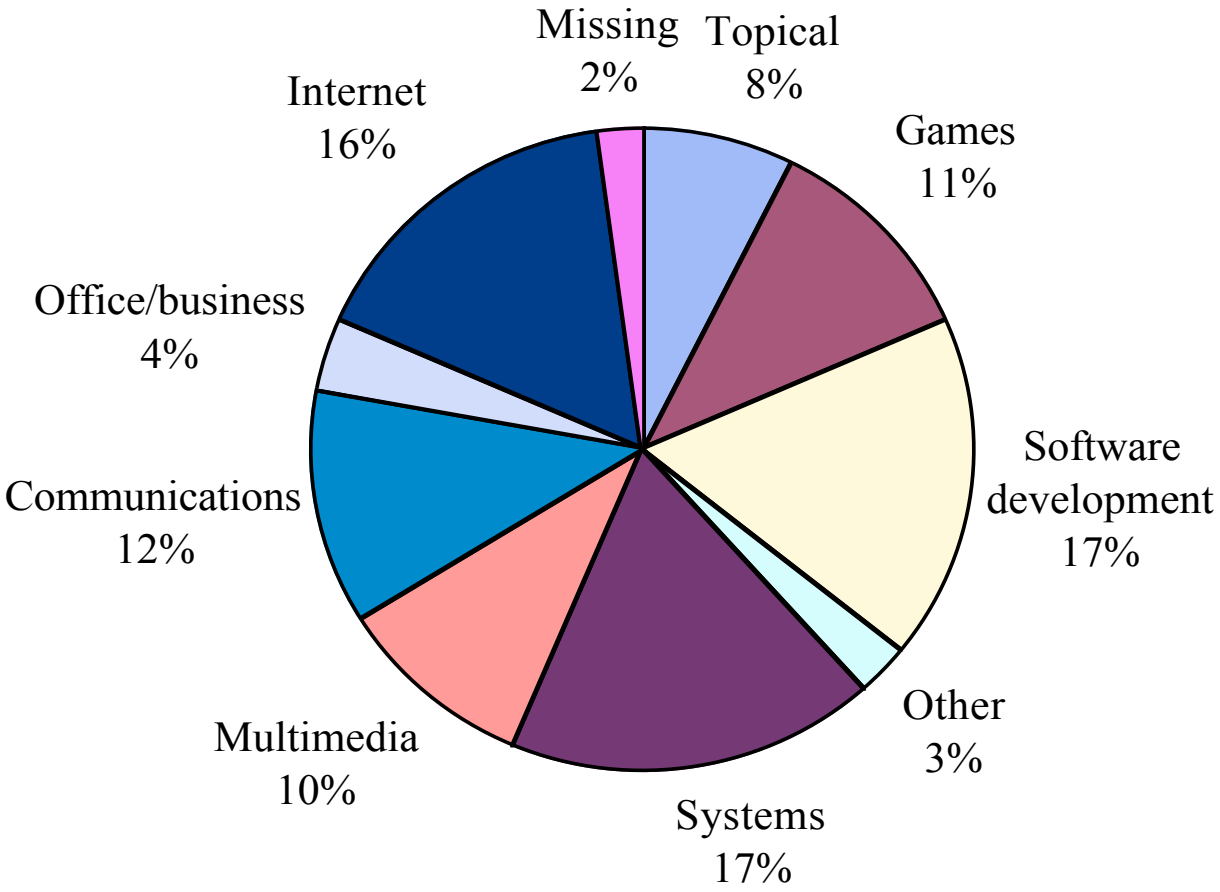


Figure 6. Pie chart of distribution of projects by topic.

Table 7. R^2 values for regression models.

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
A	.690	.477	.476	.2240
B	.358	.128	.127	1.40
C	.359	.129	.127	.8294

Table 8. Model A: Regression coefficients for activity (transformed rank percentile).

Hyp		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	*Std. Error*	*Beta*		
	(Constant)	.254	.022		11.602	.000
H1	Avg ln use of prog lang	.007101	.003	.022	2.502	.012
	Dummy for multimedia topic	.02007	.010	.018	2.032	.042
H3.1	Ln Developers	.251	.008	.337	30.259	.000
H3.1	Ln Administrators	.109	.013	.091	8.630	.000
H3.2	Dummy for administrator is peer rated	.08472	.010	.078	8.452	.000
	Project lifespan	.001315	.000	.420	43.696	.000

Note: Hyp. column indicates hypothesis tested by the given coefficient.

Table 9. Model B: Regression coefficients for development status.

Hyp		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	*Std. Error*	*Beta*		
	(Constant)	8.006	.060		133.520	.000
~~H2.1~~	Dummy for developer intended audience	–.218	.053	–.056	–4.124	.000
H2.1	Dummy for systems administrator intended audience	.145	.072	.025	2.019	.044
	Dummy for other intended audience	–.419	.094	–.053	–4.454	.000
H2.2	Dummy for office/business topic	–.630	.103	–.070	–6.094	.000
H2.2	Dummy for software development topic	.159	.055	.037	2.896	.004
	Dummy for games topic	–.617	.058	–.128	–10.654	.000
~~H3.1~~	Ln Developers	–.542	.050	–.150	–10.751	.000
H3.2	Dummy for administrator is peer rated	.461	.062	.088	7.433	.000
H4	Rank percentile (transformed)	1.450	.076	.300	19.145	.000
	Project lifetime	.001124	.000	.074	5.322	.000

Note: Hyp. column indicates hypothesis tested by the given coefficient. Struck-out hypotheses are contradicted by the data.

Table 10. Model C: Regression coefficients for use (downloads and page views).

Hyp		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	*Std. Error*	*Beta*		
	(Constant)	–1.020	.083		–12.309	.000
H1	Avg ln use of prog lang	.06640	.011	.072	6.226	.000
~~H2.1~~	Dummy for end user intended audience	.161	.028	.069	5.684	.000
H2.2	Dummy for office/business topic	–.161	.062	–.030	–2.590	.010
H2.2	Dummy for software development topic	.101	.033	.040	3.069	.002
H2.2	Dummy for system topic	.07893	.032	.031	2.461	.014
	Dummy for multimedia topic	.189	.039	.059	4.809	.000
	Dummy for games topic	–.161	.036	–.056	–4.450	.000
H3.1	Ln Developers	.690	.029	.322	24.170	.000
~~H3.1~~	Ln Administrators	–.198	.046	–.058	–4.263	.000
H3.2	Dummy for administrator is peer rated	.305	.039	.098	7.886	.000
H3.2	Dummy for administrator is peer ranked	.289	.112	.031	2.573	.010

Note: Hyp. column indicates hypothesis tested by the given coefficient. Struck-out hypotheses are contradicted by the data.

Table 11. Summary of hypotheses supported and rejected.

Hyp.	**A**	**B**	**C**	**D**
Dependent variable:	Activity (rank percentile)	Development status	Use	Project Counts
H1	✓ Supported	n.s.	✓ Supported	
H2.1	n.s.	Partially supported	× Rejected	Partially supported
H2.2	n.s	✓ Supported	✓ Supported	✓ Supported
H3.1	✓ Supported	× Rejected	Partially supported	
H3.2	✓ Supported	✓ Supported	✓ Supported	
H4		✓ Supported		