Defining Open Source Software Project Success

Kevin Crowston, Hala Annabi and James Howison

School of Information Studies
Syracuse University

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

+1 (315) 443–1676
Fax: +1 (866) 265-7407
Email: crowston@syr.edu, hpannabi@syr.edu,
james@freelancepropaganda.com

Submission to the *Management of IT—Strategy, Planning, Policies* Track
of the 2003 *International Conference on Information Systems*

Draft of 5 May 2003

Defining Open Source Software Project Success

Abstract

Information systems success is one of the most widely used dependent variables in information systems research. In this paper, we identify a range of measures that can be used to assess the success of Open Source Software (OSS) projects. We identify measures based on a review of the literature, a consideration of the OSS development process and an analysis of the opinions of OSS developers. For each measure, we provide examples of how they might be used in a study of OSS development.

Submission to the *Management of IT—Strategy, Planning, Policies* Track
of the 2003 *International Conference on Information Systems*

6506 words, 21 pages (1 title/abstract + 19 text)

Keywords: Open Source Software, software development, project success

Defining Open Source Software Project Success

Information systems success is one of the most widely used dependent variables in information systems research. Not surprisingly much attention has been given to how best to measure it (e.g., DeLone & McLean, 1992, 2002; Rai, Lang, & Welker, 2002; Seddon, 1997; Seddon, Staples, Patnayakuni, & Bowtell, 1999). In this paper, we identify measures that can be applied to assess the success of Open Source Software (OSS) projects based on a brief review of the literature, a consideration of the OSS development process and an analysis of the opinions of OSS developers. Since project success is likely to be a multi-dimensional construct, especially for OSS, our goal is to present a range of measures for future researchers to consider. As Seddon (1999) says, a "diversity of IS effectiveness measures is to be encouraged".

The remainder of this paper is organized as follows. To ground our investigations, we first present the research context, namely Open Source Software development. We then briefly review the literature on IS success to see what measures might be adopted and to identify problems in applying others. To address the problems, we reexamine the underlying vision of system development and identify additional success measures that might be appropriate for OSS. For each measure, we provide examples of how they might be used in the context of a study of OSS. Finally, we compare our selection of measures to the opinions of OSS developers as expressed in an informal poll taken on SlashDot, a popular Web-based discussion board (http://slashdot.org/). The comparison suggests additional measures that might be incorporated to develop a fuller understanding of OSS project success. We conclude by making some suggestions for future research.

Research context

OSS is a broad term used to embrace software that is developed and released under some sort of "open source" license. There are many licenses with a range of different features (Gacek, Lawrie, & Arief, n.d.), but all allow inspection of the software's source code. Much (though not all) of this software is also free software, in two senses: "free as in speech", meaning that the code may be freely redistributed and reused in other OSS projects, and "free as in beer", meaning that the software is available for download without charge. There are thousands of OSS projects, spanning a wide range of applications. Due to their size, success and influence, the Linux operating system and the Apache Web Server are probably the most well known, but hundreds of

other projects are in widespread use, including projects on Internet infrastructure (e.g., sendmail, bind), user applications (e.g., the GIMP, OpenOffice), programming languages (e.g., Perl, Python, gcc) and games (e.g., Paradise). As well, many (though by no means all) programmers contribute to the project as volunteers, without working for a common organization or being paid. As we will see, these characteristics have implications for the applicability of certain measures of success.

It is important to develop measures of success for OSS projects for at least two reasons. First, having such measures should be useful for OSS project managers in assessing their projects. In some cases, OSS projects are sponsored by third parties, so measures are useful for sponsors to understand the return on their investment. Second, OSS is an increasingly visible and copied mode of systems development. Millions of users depend on OSS systems such as Linux (and the Internet, which is heavily dependent on OSS tools), but as Scacchi (2002a) notes, "little is known about how people in these communities coordinate software development across different settings, or about what software processes, work practices, and organizational contexts are necessary to their success". A recent EU/NSF workshop on priorities for OSS research identified the need both for learning "from open source modes of organization and production that could perhaps be applied to other areas" and for "a concerted effort on open source in itself, for itself" (Ghosh, 2002). But to be able to learn from teams that are working well, we need to have a definition of "working well".

The most commonly used models of IS success are based on a vision of system development in an organization. These models are appropriate for traditional IS projects, but some of the resulting measures are difficult to apply to OSS because of its unique features, such as free distribution and volunteer developers.

Literature review

OSS is a form of system development, so we begin our hunt for success measures in the Information Systems literature. Note though that we are not attempting an exhaustive review of this extensive literature, but rather are trying to use the literature to guide on consideration of possible measures. The most commonly cited model for IS success is DeLone and McLean (1992), shown in Figure 1. This model suggests 6 interrelated measures of success: system quality, information quality, use, user satisfaction, individual impact and organizational impact. Seddon (1997) proposed a related model that includes system quality, information quality,

perceived usefulness, user satisfaction, and IS use. Taken together, these models suggest a number of possible measures that could be applied to OSS.

System and information quality. Code quality has been studied extensively in software engineering. This literature provides many possible measures of the quality of software, including understandability, completeness, conciseness, portability, consistency, maintainability, testability, usability, reliability, structuredness and efficiency (Boehm, Brown, & Lipow, 1976; Gorton & Liu, 2002). Code quality measures would seem to be particular applicable for studies of OSS, since the code is publicly available. Indeed, a few studies have already examined this dimension. For example, Stamelos et al. (2002) suggested that OSS code is generally of good quality. On the other hand, not many OSS systems include information per se, so information quality seems to be not as applicable.

User satisfaction. User satisfaction is an often-used measure of system success. For example, it is common to ask stakeholders if they felt a project was a success (e.g., Guinan, Cooprider, & Faraj, 1998). There is some data available regarding user satisfaction with OSS projects. For example, Freshmeat, a Web-based system that tracks releases of OSS (http://freshmeat.net/), collects user ratings of projects. Unfortunately, these ratings are based on a non-random sample (i.e., users who take the time to volunteer a rating), making their representativeness suspect. Furthermore, we have observed that the scores seem to have low variance: in a recent sample of 59 projects, we found that scores ranged only from 7.47 to 9.07. It seems likely that users who do not like a piece of software simply do not bother to enter ratings. There do not seem to be any easily obtainable data on the related measures of perceived ease of use and usefulness (F. D. Davis, 1989). Opinions expressed on project mailing lists are a potential source of qualitative data on these facets, though again there would be questions about the representativeness of the data.

In principle, it should be possible to survey users to collect their satisfaction with or perceptions of the software. However, to do so properly poses a serious methodological problem. Because most OSS projects are freely distributed through multiple channels, the population of users is unknown, making it impossible to create a true random sample of users. In this respect, OSS differs greatly from information systems developed in an organizational setting that have a clearly defined user population. The situation is also different than for the Web, another non-traditional systems environment, because with a Web site, the users are by definition the ones

who visit the site. Though doing so might annoy some users, the best solution might be to build the survey into the software. For example, recent versions of the Mozilla Web browser include a program that offers to report crashes and collect other feedback.

Use. Although there is some debate about its appropriateness (DeLone & McLean, 2002; Seddon, 1997), many studies employ system use as an indication of information systems success. For software for which use is voluntary, as is the case for most OSS, use seems like a potentially relevant indicator of the project's success. Some interesting data are available. Avery Pennarun' Debian Popularity Contest (http://people.debian.org/~apenwarr/popcon/) collects statistics on the usage of software on Linux machines running the Debian distribution. Users install a program that collects and reports usage information daily and the resulting statistics show which packages have been installed, and which of these have been recently used. Unfortunately, these data are collected from a non-random sample of machines, running a particular Linux distribution, so the results may not be representative of use in the broader population.

Rather than measuring actual use, it may be sufficient to count the actual or potential number of users of the software, which we label "popularity". For rare projects, these numbers can be directly measured. For example, Netcraft conducts a survey of Web server deployment (http://news.netcraft.com/archives/webserver_survey.html), which estimate the market share of the Apache Web server. Other projects that require some kind of network connection could potentially be measured in the same way, but this approach does not seem to be suitable for many projects.

A simple measure of popularity is the number of downloads made of a project. These numbers are readily available from various sites. Of course not all downloads result in use, so variance in the conversion ratio will make downloads an unreliable indicator of use. Furthermore, because OSS can be distributed through multiple outlets on-line as well as offline (e.g., on CDs), the count from any single source is likely to be quite unreliable as a measure of users. A particularly important channel is "distributions" such as RedHat, SuSE or Debian. Distributions provide purchasers with pre-selected bundles of software packaged for easy installation and are often sold on a CD-ROM to obviate the need to download everything. Indeed, the most popular software might be downloaded only rarely because it is already installed on most users' machines and stable enough to not require the download of regular

updates. Therefore, an important measure of popularity to consider is the package's inclusion in distributions.

Other sources of data reflecting on users are available. Freshmeat provides a measure for packages it tracks that it calls popularity, though a better name might be "interest", as it is one step further removed from actual use. The measure is calculated as the geometric mean of subscriptions and two counts of page viewings of project information (for the precise details, see http://freshmeat.net/faq/view/30/). Similarly, SourceForge provides information on the number of page views of the information pages for projects it supports.

Finally, it may be informative to measure use from perspectives other than that of an end user. In particular, the openness of OSS means that other projects can build on top of it. Therefore, one measure of a project's success may be that many other projects use it. Package dependency information between projects can be obtained from that package descriptions available through the various distributions' package management systems.

Individual or organizational impacts. The final measures in DeLone and McLean's (1992) model are individual and organizational impacts for the users. Though there is considerable interest in the economic implications of OSS, these measures are hard to define for regular I/S projects and doubly hard for OSS projects, because of the problems defining the intended user base and expected outcomes. Therefore, these measures are likely to be unusable for most OSS studies.

Conclusion

To summarize, existing models of information systems success suggest a range of potential success measures for OSS projects as shown in Table 1. However, a number of the measures are inapplicable, while others are difficult to carry out in the OSS environment. Finally, the measures do not take into account some of the unique characteristics of the OSS development environment. In the next section, we examine the process of OSS development in more detail to identify additional measures of success.

The process of OSS development

In this section, we reexamine the vision of systems development underlying DeLone and McLean's success model to identify additional measures that might be used for OSS project success. DeLone and McLean's (1992) state that their model was built by considering "a process

model [that] has just three components: the creation of a system, the use of the system, and the consequences of this system use" (DeLone & McLean, 2002), which we have shown graphically in Figure 2. We note that the measures include in the model focus on the use and consequences of the system (the right side of the figure), and do not open up either box in the process. While this focus may be appropriate given the traditional concern of information systems research with the organizational implication of IS, it seems to unduly restrict the range of measures considered.

The choice of measures also seems to be influenced by the relative ease of access to the use environment compared to the development environment (especially for packaged or commercial software). In the context of OSS though, researchers are frequently faced with the opposite situation, in that the development process is publicly visible and the use environment is difficult to study or even identify. For both reasons, we believe that it will be useful to complement existing success measures with ones that take advantage of the availability of data on the development process. The following discussion examines such measures of success, some of which have been previously discussed in the information systems literature.

Measures of the output of systems development

Two of the measures in the DeLone and McLean model concern the product of the systems development process, namely systems quality and information quality. We first consider possible additional measures of the systems development output.

Project completion. First, given large number of abandoned projects (Ewusi-Mensah, 1997), just completing the project may be a sign of success. However, many OSS projects are continually in development, making it difficult to say when they are completed. Faced with this problem, Crowston & Scozzi (2002) instead measured success as the progress of a project from alpha to beta to stable status, as self-reported on SourceForge.

Second, another commonly used measure of success is whether the project achieved its goals. This assessment is typically made by a comparison of the project outcomes with the formal requirements specification. However, OSS projects often do not have such specifications. Scacchi (2002b) examined the process of "requirements engineering" in open source projects and provided a comparison with the traditional processes (e.g., A. M. Davis, 1990; Jackson, 1995). He argues that requirements engineering does occur for OSS projects, but in a significantly different fashion through what he terms "software informalisms", which do not result in agreed "requirements documentation" that could later be analyzed to see whether the project has met its

goals. Scacchi concludes that, in sharp contrast to traditional requirements engineering, "Developing open software requirements in a community building process that must be institutionalized both within a community and its software informalisms to flourish" (Scacchi, 2002b). He concludes that the traditional measures of the quality of requirements for software will deal badly with this dynamic and informal processes and their dual role as community building practices, valuable for their own sake. Scacchi's ethnography suggests that for OSS, goals are likely come from within, through a discursive process centered on the developers. Therefore, a key measure for OSS may be simply developer satisfaction with the project, which could be measured by surveying developers. The developer community is much more clearly delineated than users, making such a survey feasible. Indeed, there have already been several OSS developer surveys (e.g., Ghosh, 2002; Hertel, Niedner, & Herrmann, n.d.), though not on this topic specifically.

Measures of the process of systems development

In DeLone and McLean's (1992) process model, systems development is implicitly treated as a one-off event creating a software system to study. However, for OSS projects (and indeed many other types of projects) development is instead an ongoing activity, as the project continues to release "often and early" (Raymond, 1998). In other words, an OSS project is characterized by a continuing process of developers fixing bugs, adding features and releasing software. This characteristic of the OSS development process suggests a number of possible measures of success.

Number of developers. First, since OSS is dependent on volunteer developers, being able to attract developers to a project on an on-going basis is important for their success. Thus the number of developers involved in a project could be an indicator of success. The number of developers can be measured in at least two ways. OSS development systems such as SourceForge list developers who are formally associated with each project. Examination of the mailing lists and other fora associated with projects can reveal the number of individuals who actively participate.

Level of activity. More important than the number of developers is their contribution to a project. Thus the level of activity of developers in submitting code and bug reports may be useful as an indicator of project success. SourceForge computes and reports a measure of project

activity based on the activities of developers. Researchers could also examine development logs for evidence of software being written and released.

Cycle time. Another measure related to the group activity is time between releases. In OSS development, there is a strong community norm to "release early and release often", which implies that an active release cycle is a sign of a healthy development process and project. For example, FreshMeat provides a "vitality score" that assesses how recently a project has made an announcement of progress (http://freshmeat.net/faq/view/27/).

In addition, detailed examination of bug-fixing and feature-request fulfillment activities might yield useful process data indicative of the project's status. These processes involve interaction with the user community and might involve applying patches of contributed code supplier by non-core developers. Bug reports and feature-requests are typically managed through a task-management system which records the developer and community discussion, permit labeling of priority items and sometimes include informal "voting mechanisms" to allow the community to express its level of interest in a bug or new feature.

We are currently exploring an analysis of the time to close bugs (or implement requested features) as a measure of project success. To collect data on bugs, we spidered the bug report data from SourceForge and extracted the length of time taken to fix bugs. Our preliminary analysis suggests that this measure that shows interesting variance between projects as well as providing useful characteristics of each bug (priority, issue area and explicit developer assignment) in addition to the project level data.

Project effects on projects

Finally, because the projects are on-going, it seems important to consider the impact of a project on the abilities of the project team itself and its ability to continue or improve the development process.

Job opportunities. Some literature on the motivation of OSS developers suggests that developers participate to improve their employment opportunities. Thus, one can consider jobs acquired through the involvement in a particular project as a possible measure of success. Again, one might measure this indicator by surveying developers.

Individual reputation. Similarly, although perhaps less concretely, literature also suggests that developers participating in OSS projects are rewarded with reputation in the community, and that this reputation is a sufficient reward for interaction. Kelty (2001) suggests that reputation

might be measured through an analysis of credits located in source code (which he terms "greputation"). Alternative measures of OSS reputation might include the OSS communities implementation of a "Web of Trust" at the community site Advogato (http://www.advogato.org/trust-metric.html) where developer status is conferred through peer-review. Analyses of this kind of measure faces the difficulty of tying the earning of reputation to the success of a particular project.

Knowledge creation. Projects can also lead to creation of new knowledge on the group level as well as for individuals (Arent & Nørbjerg, 2000). A project can integrate members' knowledge into group procedures, rules, norms and product. This learning can be measured by observing changes in the rules and procedures over time and may be reflected and transferred through the development of systems for OSS project support, such as Sourceforge and Savannah. Analysis of the development of support systems closely linked to a project might give some insight into this aspect of project success.

In summary, consideration of the process of developing OSS suggests a number of additional measures indicative of success for these projects. These measures are summarized in Table 2. We note that as the measures move further back in the process model, they become increasingly removed from the user. As such, there may be a concern about their validity as measures of success: is it a success if a project attracts developers but not users? We have two replies to this concern. First, the apparent disconnect may be an accurate representation of the reality of OSS projects, in which the developers are the users. Second, the measures developed in this section should be viewed as complements to rather than replacements for the more conventional measures of success. Using a variety of measures will provide a richer picture of the status of a project.

Open source developer opinions

In the previous two sections, we developed a list of possible success measures for OSS projects based on a review of the literature and consideration of a simple model of OSS development. To determine whether these measures had content validity for OSS project success and to identify additional possible measures, we sought input from OSS developers. In this section, we discuss our data elicitation and analysis techniques and results from the analysis.

Methods and data

To solicit input, we posted a question soliciting feedback on Slashdot (http://slashdot.org/), a Web-based discussion group that attracts considerable interest and participation from OSS developers and users. This data elicitation technique was more like an on-line focus group than a survey, as respondents were a non-random sample, and could see and respond to earlier postings. This approach was chosen to match our goal of generating ideas about success measures, rather than testing a theory or developing generalizable data. The following question was posted on Slashdot on 22 April 2003 (http://slashdot.org/article.pl?sid=03/04/21/239212):

> There have been a number of discussions on Slashdot and elsewhere about how good projects work (e.g., Talk To a Successful Free Software Project Leader), but less about how to tell if things are going well in the first place. While this may seem obvious, most traditional definitions of software project success seem inapplicable (e.g., profit) or nearly impossible to measure for most projects (e.g., market share, user satisfaction, organizational impact). In an organizational setting, developers can get feedback from their customers, the marketplace, managers, etc.; if you're Apache, you can look at Netcraft's survey of server usage; but what can the rest do? Is it enough that you're happy with the code? I suspect that the release-early-and-often philosophy plays an important role here. I'm asking not to pick winners and losers (i.e., NOT a ranking of projects), but to understand what developers look at to know when things are going well and when they're not.

The question received 201 responses within a few days, though many of these responses did not in fact address the question. Many of the individuals posting answers to our question identified themselves as developers or contributors to OSS projects.

A transcript of responses was downloaded on 26 April and content analyzed by two coders. Thematic units in messages were identified and coded into a theoretical category using Atlas-ti. A total of 170 thematic units were identified. The initial content analytic scheme was based on the literature review described above. During the process of content analysis, additional themes emerged from the data. Saturation was reached through the content scheme presented in Appendix I. The two raters agreed on the codes for 78% of the units. We felt that this level of agreement was sufficient for the purposes of the analysis (identification of possible measures), so we did not go on to refine the definitions of codes or retrain the coders to increase agreement.

Results

The outcome of the content analysis is summarized in Table 3. The codes were organized into a two-level hierarchy for presentation, with detailed codes (level 2 in the table) clustered into meta-categories (level 1).

32% of the responses included elements from the developers meta-category, indicating that the respondents felt that a project is successful if the developers are involved, satisfied, enjoy the process and that there is a variety of them. The Users meta-category also had a large number of responses. 23% of responses indicated that the poster felt a project was successful if it satisfies users (other than developers) and that the users are involved in discussions and bug reports. Involvement of both users and developers was frequently mentioned, accounting for 31% of the responses. Project recognition codes were found in 11% of units, exceeding the number of responses indicating use as a measure of success, 5% of instances. Finally, the product's quality (13%) and process (13%) were suggested to be measures of success by developers as well.

Discussion

The response of the developers posting on SlashDot were generally in agreement with the list of success measures we developed from the literature and our reexamination of the process. The analysis indicates that developers' found their personal involvement, satisfaction and enjoyment to be important in assessing the success of a project, consistent with the view of OSS as "software that scratches an itch". More interestingly, some new themes did emerge from the coding.

- First, a number of respondents suggested recognition (e.g., mention on other sites), as a measure of project success. Similarly, another suggested measure was the influence of the product or project's process on other OSS groups and other commercial settings. These responses are consistent with the literature on OSS developers' motivations that suggest recognition as a primary motivation for involvement.
- A second category that emerged was the level of involvement of the users as indicated by involvement of the users in submitting bug reports and participating in the project mailing lists. We had considered contributions from developers, but these responses reflect that fact that OSS projects are also dependent on help from users to identify problems and post suggestions.

- A final category that emerged from the data was the issue of porting. Developers consider porting of a product to different systems (especially to Windows), and requests for such ports as a measure of the success of the product. This theme might be considered a special case of popularity.

What was also surprising to us was what respondents did not say. A few of the measures of success we identified were not mentioned by respondents. For example, though several authors have suggested that developers are motivated by the chance to learn and perhaps get a better job, none of the respondents mentioned these factors. A possible explanation is the strong community norm that endorses altruism over expressions of self-interest, which may have restricted discussion in the non-anonymous and community-moderated Slashdot forum.

Conclusion

This paper makes a contribution to the developing body of empirical research on OSS by identifying a collection of success measures that might be applied to OSS. We have identified a range of possible measures by applying a popular model of IS success and by more detailed consideration of the vision of software development underlying that model. Furthermore, we have identified where data is available for particular measures. Finally, we validated our list by comparing it to opinions of community, which suggested additional measures beyond those we identified.

We emphasize again that we do not view any single measure as the final word on success. As the measures focus on different aspects of the process, we expect that they will offer different perspectives on the process. Therefore, we suggest using a mix of measures, or perhaps developing synthetic measures that draws on different perspectives. Using multiple measures might be particularly interesting for examining how projects change their emphasis from one measure to another at different points in their evolution (Heo & Han, 2003).

References

Arent, J., & Nørbjerg, J. (2000). Software Process Improvement as Organizational Knowledge Creation: A Multiple Case Analysis. In *Proceedings of the 33rd Hawaii International Conference on System Sciences* (pp. 11 pages): IEEE Press.

Boehm, B. W., Brown, J. R., & Lipow, M. (1976). Quantitative evaluation of software quality. In *Proceedings of the 2nd International Conference on Software Engineering, October 13-15* (pp. 592–605). San Francisco, CA.

Crowston, K., & Scozzi, B. (2002). Open source software projects as virtual organizations: Competency rallying for software development. *IEE Proceedings Software, 149*(1), 3–17.

Davis, A. M. (1990). *Software Requirements Analysis and Specification*. Englewood Cliffs, NJ: Prentice-Hall.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use and user acceptance of information technology. *MISQ, 13*, 319–340.

DeLone, W. H., & McLean, E. R. (1992). Information Systems Success: The Quest for the Dependent Variable. *Information Systems Research, 3*(1), 60–95.

DeLone, W. H., & McLean, E. R. (2002). *Information systems success revisited.* Paper presented at the Proceedings of the 35th Hawaii International Conference on System Sciences.

Ewusi-Mensah, K. (1997). Critical issues in abandoned information systems development projects. *Communication of the ACM, 40*(9), 74–80.

Gacek, C., Lawrie, T., & Arief, B. (n.d.). *The many meanings of Open Source* (Unpublished manuscript). Newcastle upon Tyne, United Kingdom: Centre for Software Reliability, Department of Computing Science, University of Newcastle.

Ghosh, R. A. (2002). *Free/Libre and Open Source Software: Survey and Study. Report of the FLOSS Workshop on Advancing the Research Agenda on Free / Open Source Software*, from http://www.infonomics.nl/FLOSS/report/workshopreport.htm

Gorton, I., & Liu, A. (2002). Software component quality assessment in practice: Successes and practical impediments. In *Proceedings of the 24th International Conference on Software Engineering* (pp. 555–558). Orlando, FL.

Guinan, P. J., Cooprider, J. G., & Faraj, S. (1998). Enabling software development team performance during requirements definition: A behavioral versus technical approach. *Inf. Syst. Res., 9*(2), 101-125.

Heo, J., & Han, I. G. (2003). Performance measure of information systems (IS) in evolving computing environments: an empirical investigation. *Inf. Manage., 40*(4), 243-256.

Hertel, G., Niedner, S., & Herrmann, S. (n.d.). *Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel*. Kiel, Germany: University of Kiel.

Jackson, M. (1995). *Software Requirements and Specifications: Practice, Principles, and Prejudices*. Boston, MA: Addison-Wesley.

Kelty, C. (2001). Free Software/Free Science. *First Monday, 6*(12).

Rai, A., Lang, S. S., & Welker, R. B. (2002). Assessing the validity of IS success models: An empirical test and theoretical analysis. *Information Systems Research, 13*(1), 50–69.

Raymond, E. S. (1998). The cathedral and the bazaar. *First Monday, 3*(3).

Scacchi, W. (2002a). *Software Development Practices in Open Software Development Communities: A Comparative Case Study (Position Paper)*.

Scacchi, W. (2002b). Understanding the Requirements for Developing Open Source Software Systems. *IEE Proceedings Software, 149*(1), 24–39.

Seddon, P. B. (1997). A Respecification and Extension of the DeLone and McLean model of IS Success. *Information Systems Research, 8*(3), 240-253.

Seddon, P. B., Staples, S., Patnayakuni, R., & Bowtell, M. (1999). Dimensions of information systems success. *Communications of the Association for Information Systems, 2*(20), 61 pages.

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.

Tables and Figures

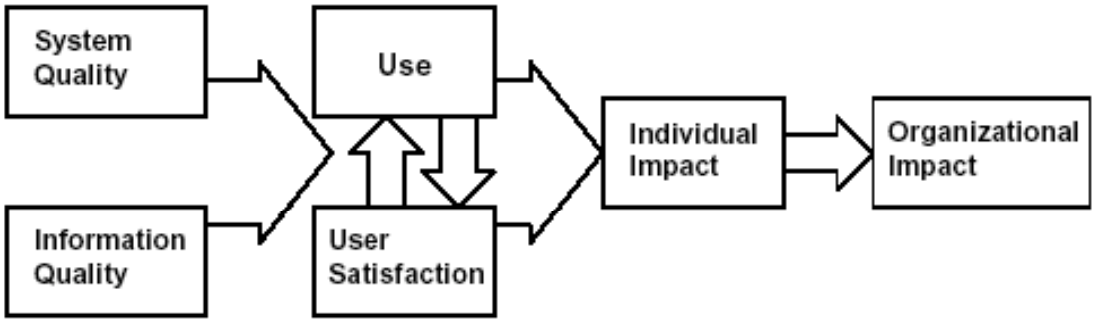

Figure 1: DeLone and McLean's Model of IS Success
[DeLone and McLean [1992], Figure 2, p.87]

Table 1. Success measures suggested by the literature review.

Measure of Success	**Indicators**
System and information quality	Code quality (e.g., understandability, completeness, conciseness, portability, consistency, maintainability, testability, usability, reliability, structuredness, efficiency)
User satisfaction	User ratings Opinions on mailing lists User surveys
Use	Use (e.g., Debian Popularity Contest) Surveys of deployment Downloads Inclusion in distributions Popularity or views of information page Package dependencies
Individual and organizational impacts	Economic and other implications

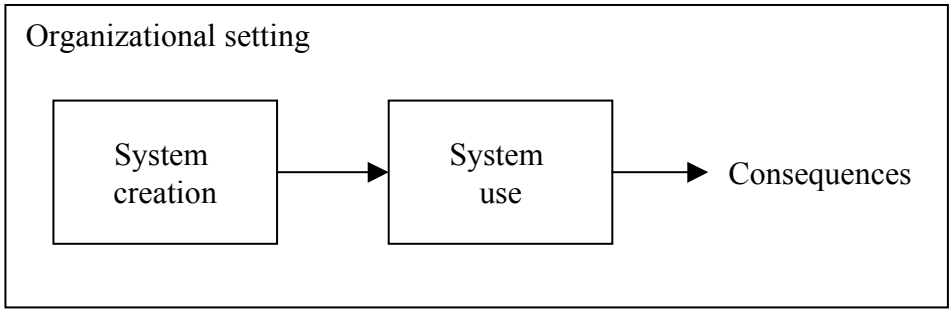

Figure 2. Process model underlying the DeLone & McLean (1992) model of success.

Table 2. Measures suggested by a reexamination of the OSS process.

Measure of Success	**Indicators**
Project output	Movement from alpha to beta to stable Achieved identified goals Developer satisfaction
Process	Number of developers Level of activity (developer and user contributions, number of releases) Time between releases Time to close bugs or implement features
Outcomes for project members	Individual job opportunities Individual reputation Knowledge creation

Table 3. Results of the content analysis of SlashDot responses.

Level 1	**Level 2**	**Frequency**	**Percentage**
User	Satisfaction	14	8%
	Involvement	25	15%
Product	Meets requirements	9	5%
	Code quality	11	6%
	Portability	1	1%
	Availability	2	1%
Process	Activity	5	3%
	Adherence to process	10	6%
	Bug Fixing	4	2%
	Time	2	1%
	Age	1	1%
Developers	Involvement	16	9%
	Varied developers	2	1%
	Satisfaction	29	17%
	Enjoyment	8	5%
Use	Competition	4	2%
	Number of users	2	1%
	Downloads	3	2%
Recognition	Referral	3	2%
	Attention and recognition	9	5%
	Spin offs	6	4%
Influence		4	2%
	Total	170	

Appendix I: Content Analytic Scheme

Level 1	Level 2	Description	Examples
User	Satisfaction	Users of the product satisfied (Code serves their need)	Every piece of software has an intended client, user or audience. Are the users happy, overall? Useful...to someone
	Involvement	Users of the product are involved and interested, by submitting to mailing lists or bug reports or other forms of contribution	Second: I've had dozens of email's asking for support as well as asking how to contribute. Traffic: both developer and user. Is there a relatively continuous level of input/interest in the project? If developers don't want to develop, and users don't want to use, it's probably going nowhere, even if it's the best thing since the BeOS. more activity in a mailing list usually indicates the size/success of a project. small error in the makefile which causes something liek 50% of people come back for help on compiling it. This gives me pretty good estimate of how many people are actually using the package
Product	Meets requirements	The product meets the requirement of design	how well does it fit the need for which it was designed?
	Code quality	Code structure and documentation is organized, clear, maintainable	does what it is supposed to do, cleanly and efficiently, then by definition it is successful it is well documented and maintained Is the code maintainable? Take a look at the end product. Does it do what it's supposed to without too many bugs? Stable in relation to the time invested.
	Portability	Software portable to and compatible with other systems and programs	successful Open Source Software tends to have a greater scope of use than it's original conception. The programs I find myself using are programs that can interact with each other in a modular fashion; whether that be throught a piped command, or simply support for "generic" file formats (such as XML, CSV etc etc). Win32 port: Win32 port of a project

	Availability	The product is available through a number of avenues	availability around the internet. (if your program is on say on most of the distributions)
Process	Activity	The project is active; fixing bugs, writing updates, documentation and releases	If there hasn't been an update to it in like 2 years, then chances are, unless it was perfect the first time around, it will fail.
	Adherence to process	The project has goals and objectives and have an established process that members adhere to	How is the process? Are there goals and are they being met? How is testing coverage and how often is testing being done? Milestones: establish concrete goals when you start the project, along with a timeline. Many goals, and projects evolve
	Bug fixing	Bug reports are attended to and fixed in reasonable time	Are issues being addressed in a timely manner?
	Time	How established is the software and how often do they release new features	Time is where you can measure your progress. This is where you can do things like determine milestones, develop feature lists and so on, then during the project you have a standard to compare yourself to.
	Age	How long has the group been active	
Developers	Contribution	There are a number of developers contributing to the project	Second: I've had dozens of email's asking for support as well as asking how to contribute Traffic: both developer and user. Is there a relatively continuous level of input/interest in the project? If developers don't want to develop, and users don't want to use, it's probably going nowhere, even if it's the best thing since the BeOS.
	Varied developers	Developers from different projects, having different expertise contribute	Software developers!
	Satisfaction	Developers satisfy their need to innovate and develop code	Open source is scratching an itch, right? Is the itch scratched? If yes, then its a success

	Enjoyment	Developers enjoy working on the code and with the group	Do you enjoy working on it? Then it's successful.
Use	In Relation to Competition	Replaced competitive products	so clearly and overwhelmingly superior to its predecessors that it supplanted them
	Number of Users	How many users are using the product in addition to the developers	Are there more people using the project than developers? If so, it's successful. If a project has no user base, then it is doomed to fail. how is a project going to succeed without a user base.
	Downloads	How many downloads of the product	can usually judge the success of your project by counting downloads from your site First: I've had hundreds of downloads, and since I run this project on a Cable Modem connection, my ISP hasn't become unhappy :)
Recognition	Referral	Other sites, projects, organizations recognize and refer to the project	Links to site: and Third and finally (I think this one is a very good indicator): There are other websites out there that link to my site. Oh, and there's a fourth optional measure of success... more for bragging rights... my site is THE FIRST result when querying google with "Java X10". use Google to see how often the name of the project comes up. Discussion in Google groups is also a good sign.
	Attention	The project attracted negative or positive attention from other institutions	You have been sued by a huge mega corp with a team of lawyers over patent infringement and the EFF comes to your rescue. Stallman demands that people call it GNU
	Spin offs	New projects or spins off original project	
Influence		Other projects adopt code or process from the project	Also adoption by other developers into the development group shows others are interested, so you must be doing something right.