An evaluation criterion for open source software projects: enhancement process effectiveness

An Evaluation Criterion for Open Source Software Projects: Enhancement Process Effectiveness

Amir Hossein Ghapanchi* and Aybuke Aurum

*Corresponding author

Biographical notes: Amir Hossein Ghapanchi is a PhD candidate at the School of Information Systems, Technology and Management, University of New South Wales, Australia. He holds a MSc in IT engineering. He has served in a number of national information systems projects. His main research interests include the success of open source software projects, IT human resource management, e-government planning and implementation, multi-criteria decision making (MCDM) and business process reengineering (BPR). He has published in several prestigious information systems journals such as *Journal of Systems and Software* and *International Journal of Information Management*.

Aybuke Aurum is an Associate Professor at the School of Information Systems, Technology and Management, University of New South Wales, Australia. She holds a PhD in Computer Science. She has over 80 publications including three edited books, namely Managing Software Engineering Knowledge, Engineering and Managing Software Requirements and Value-Based Software Engineering. Her research interests include value-based approach in software development, management of software development process and requirements engineering. She is on the editorial boards of Requirements Engineering Journal and Information and Software Technology Journal.

Abstract. Low responsiveness of project team to user needs has been reported as one of the critical concerns of open source software (OSS) adopters. Enhancement process is a key process in which OSS project responds to user needs in terms of suggesting and implementing software features, thus the dimension of enhancement effectiveness corresponds nicely to adopters' concern about open source software. Therefore, it is imperative to change the attitude of managers of OSS projects which have ineffective enhancement processes. One strategy that facilitates this change management issue is to provide project managers with a model to measure their enhancement processes. Although scholars have researched the enhancement process of OSS projects for almost one decade, the literature still lacks from rigorous ways to measure this process. Therefore this study aims to construct a valid, reliable measurement model for the enhancement process effectiveness in an open source environment through the scale development methodology introduced by Churchill (1979). We examine the validity and reliability of an initial list of indicators through two rounds of data collection and analysis from 240 and 750 OSS projects respectively, and come up with a measurement model for the effectiveness of enhancement process comprising four indicators. The

implication of this measurement model for practitioners is explained through a numerical example followed by implications for research community.

Key words: Open source software, Change management, Information system evaluation, Enhancement process.

1 Introduction

Adoption of open source software (OSS) has resulted in $60 billion per year savings to its consumers. Johnson (2008) states "… while it [OSS] is only 6% of estimated trillion dollars IT budgeted annually, it represents a real loss of $60 billion in annual revenues to software companies". Although OSS constitutes less than 1% of global software spend, it contributes to reduce more than 25% of such spending Tiemann (2009). Thus, a critical area of academic interest has been studying OSS projects (Chengalur-Smith et al. 2010; Subramaniam et al. 2009; Stewart et al. 2006). As Crowston et al. (2006) state enhancement process, "adding features" to the software, is one of the most important continuing processes that characterizes OSS projects. In an open environment, "Feature requests" are typically handled through a tracking system that provides the project with an infrastructure to manage reporting features, assigning the job of feature implementation, and finally implementing the feature. Enhancement is a never-ending process in which users suggest their functional needs to the project community. Hence, effective enhancement corresponds to high responsiveness to user community. That is why prior research on OSS projects has implied the importance of effective enhancement process in impacting OSS success (Crowston et al. 2006).

This study contributes to the literature by providing a measurement model for the effectiveness of the enhancement process in OSS projects. We focus on the enhancement process for three main reasons. Firstly, enhancement process is within the control of OSS project managers to a high extent, therefore having a model to measure it, is of high significance for OSS practitioners to evaluate their projects. Secondly, an effective enhancement process means a high responsiveness of an OSS project to the user community, in terms of adding new features, that has been reported

as key concerns of OSS adopters (Golden 2004). Accordingly a measurement model for enhancement effectiveness provides OSS adopters with a software evaluation criterion. Thirdly, enhancement is tied to users' perceptions of the project quality, activity and value (Mockus & Weiss 2008), thus effective enhancement process might contribute to attracting a higher user interest in OSS projects that has been often reported as a critical success factor for OSS projects (Stewart et al. 2006).

To our knowledge, although few papers have studied tasks involved in the enhancement process (e.g. feature suggestion, or feature assignment), there is a lack of study that views enhancement as a process. Moreover, the importance of enhancement effectiveness has been strongly highlighted by prior research (Crowston et al. 2003; Crowston et al. 2006), but the development of a valid and reliable measurement model to gauge this phenomenon has not been reported in OSS literature. Therefore, the current study takes the first step towards constructing a reliable and valid measurement model for the effectiveness of the enhancement process in OSS projects. It is hoped that this measurement model will be further improved by future researchers.

Effective enhancement process is a success Indicator for OSS projects (Crowston et al. 2003). Therefore, having a measurement model for enhancement process can help OSS project administrators to better gauge the effectiveness of their enhancement processes. Such measurement model can also provide an evaluation criterion to organizational users who are interested in adopting OSS projects. One criterion for these organizations to assess alternative OSS projects of the same type would be comparing the extent to which each software project's team operate the feature adding activity more effectively.

The remainder of this paper is structured as follows. Next section reviews the related literature on OSS as well as enhancement process. Section three introduces the enhancement process in the environment under the study. The scale development methodology employed in this research is presented in Section four. The empirical study undertaken including data collection, analysis and findings are presented in

Section five. Section six presents the result of post-hoc analysis. Section seven illustrates the measurement model proposed in this study using a numerical example followed by discussions and conclusions in Section eight.

2 Research Background

2.1 Open Source Software

OSS originated in the early 1960s, when key foundations of Internet were being constructed in academic settings like MIT and Berkeley (Ducheneaut 2003). That was probably of early attempts to share software source code by developers. OSS was termed in February 1998 by a group of "free software" supporters, including Eric S. Raymond and Tim O'Reilly (Midha 2007).

Open source software is best understood in contrast with closed source software (CSS). Although in CSS the program's source code is a trade secret and is protected by law, in OSS the source code is publicly available for anyone who would like to see it. CSS projects hire developers and pay them to develop software and try to sell it, while OSS projects seeks to attract volunteer programmers to develop a software under the terms of a license that eventually lets everybody have the outcome of the work and even use its source code.

A typical OSS project starts with what Raymond (Raymond 1999) calls "scratching a developer's personal itch". An OSS project initiator who has a software idea starts writing the code. Since the community is intended to be able to see the software, it is released under a license that allows the community to see the source code and use the software. The community users can contact the project team and request new features or report a bug in the system. As a result of this evolution, OSS is said to meet user needs better than traditional closed-source software (Loshin 2005). In addition, being involved with development process, users may be more satisfied with open source software (Midha 2007).

Even though, access to source code is normally open in open source software (OSS), there may be some exceptions like software developed under Microsoft's share source

initiative[1]. Normally open source software is developed by volunteers rather than paid developers, but there are some contradictory cases like Linux which is developed by volunteers as well as paid developers. However, researchers agree on the definition proposed by Open Source Initiative (OSI). OSI defines OSS as software released under a license approved by Open Source Initiative (2005). OSS could be free or commercial; however, the focus of this study will be on free OSS. Hence, hereafter when we use OSS we actually mean free open source software.

OSS is worth researching because of the large number of open source software that have been highly successful and are being used by millions of users (Ghapanchi & Aurum 2011a; Ghapanchi & Aurum 2011b). Apache, Mozilla Firefox, Linux, Unix, and Perl are examples of such software.

2.2 Enhancement Process in OSS Projects

Enhancement is defined as a process to augment a software product with features not originally incorporated (Notkin & Griswold 1988). Enhancement process aims the software project to quickly identify and develop new features to keep them competitive. The cost of enhancement phase accounts for nearly 40% of the total life-cycle costs of software development (Lientz & Swanson 1980).

In the traditional software development, the maintenance performer is responsible for all aspects of maintenance: correction, retargeting, and enhancement (Lientz & Swanson 1980). A user demanding maintenance submits a request to the maintenance performer, who modifies the source code to apply the change. In OSS development, the user can perform many enhancements directly.

Enhancement process, “adding features” to the software (Crowston et al. 2006), is one of the most important continuing processes that characterizes free OSS projects along with “fixing defects” and “release management”. “Feature requests” are handled through a tracking system that provides the project with an infrastructure to manage

1 See http://www.microsoft.com/resources/sharedsource/default.mspx

reporting features, assigning the job of feature implementation, and finally implementing the feature.

Compared with CSS, in OSS user features are more rapidly developed because creativity is more prevalent in OSS (Dalle & Jullien 2000; O'Reilly 1999). O'Reilly (1999) believes that open source software are usually more extensible than closed source ones because there is a tighter coupling in the latter. Therefore, adding a new feature to a closed-source software requires more changes in existing modules than in OSS (Paulson et al. 2004). Also, the number of features added is found to be greater in OSS compared with CSS; which means that OSS approach supports more features over time than CSS (Paulson et al. 2004).

In OSS literature, some researchers have taken enhancement process into account. Stewart & Gosain (2006a) looked at the percentage of feature requests completed as an indicator of OSS project effectiveness. They suggest that OSS project success comprises the extent to which a project receives input from the community (e.g. the number of developers), and the extent to which it creates an observable output such as new features. Herbsleb & A. Mockus (2003) stated that the progress in adding new features, fixing bugs reported and responding to user interests reflect the outcomes of an OSS project. Moreover, Stewart & Gosain (2006b) demonstrated that the percentage of feature requests completed impacts perceived effectiveness of OSS projects.

Although there are some studies attempting to measure the effectiveness of enhancement process for traditional closed-source software development (Kemerer & Slaughter 1997; Banker & Slaughter 2000), there is a lack of studies that measured that for open source software projects. For example Banker & Slaughter (2000) used application enhancement costs defined as the total dollars that were incurred to additions, modifications, and deletions of functionality of the application during a specified time frame. Moreover, Kemerer & Slaughter (1997) used the total number of adds, changes and deletes made to the functionality of the software module over its lifetime.

3 Enhancement Process in Sourceforge.net

The setting chosen for this research is the largest OSS repository, Sourceforge.net. As of May 2011, Sourceforge has 260,000 registered open source software projects, and it also has more than 2.7 million registered members (Source: www.sourceforge.net). Sourceforge.net doesn't clearly specify the enhancement process through its feature tracking system, but pre-defines four statuses for a feature including open, closed, pending, and deleted. "Open" status is used when a feature is first suggested. Subsequently, someone (e.g. a project administrator) either assigns it to a developer to implement, or "delete" the feature if it is duplicate or not legitimate; "Pending" status is also used when the feature is legitimate but it is better to be implemented at a point of time in future. Finally, when the feature is implemented, he changes the status to "closed". Enhancement process in Sourceforge.net typically involves three tasks: suggesting a feature, reviewing and assigning the feature, implementing the feature. Following we explain each task.

Suggesting a feature:

It starts with a community member suggesting a new feature to be added to the software. As Figure 1 shows a feature reporter enters a summary of the problem as well as a description. The system also allows the feature reporter to attach a file.

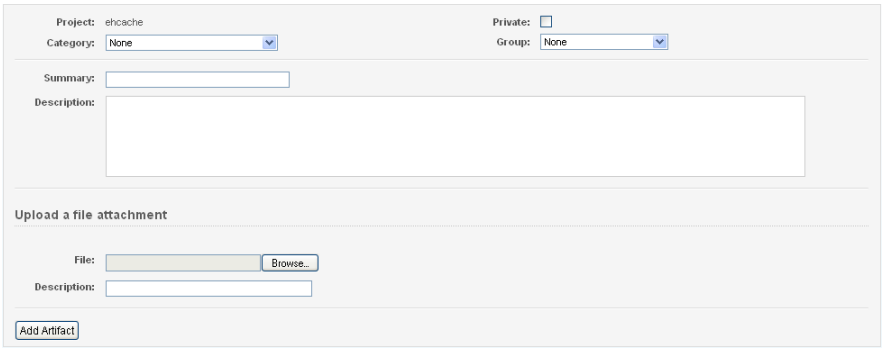


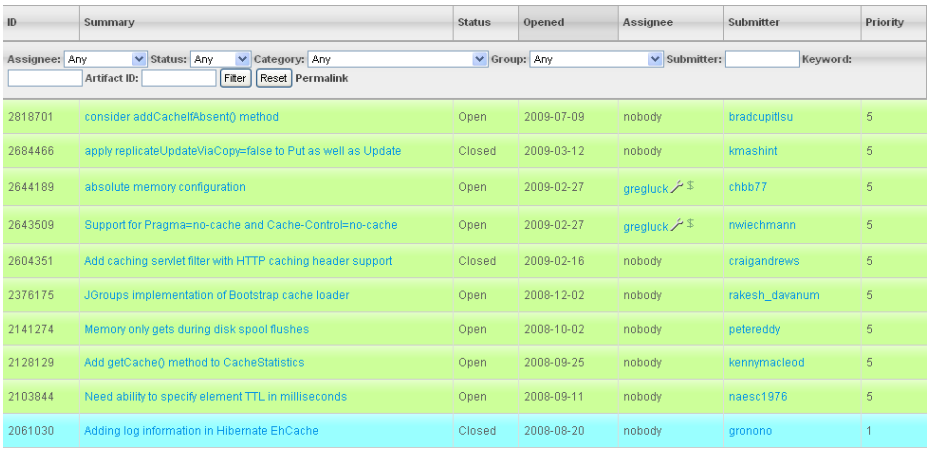
Fig. 1. A feature reporting page on Sourceforge.net

Reviewing and assigning the feature:

When a feature is requested, the project administrator (or whoever is in charge) reviews it. If it is worth implementing, s/he assigns it to a developer or a group of developers. Here, developers' motivation is highly important since majority of them are not paid and no one can force them to fulfill a requested feature that they don't like to work on.

Implementing the feature:

Implementing the feature is the final task of enhancement process. When a feature report is assigned to a developer, he starts working on it. When the feature is completed, the developer changes the status to "closed". Figure 2 displays the feature tracking system that SF.net freely offers to the projects that register on it. Each record presents a reported feature in terms of its summary, status, date opened, assignee, submitter, and priority. Each feature also gets a priority that ranges from 1 (less important) to 9 (more important).



ID	Summary	Status	Opened	Assignee	Submitter	Priority
Assignee: Any Status: Any Category: Any Group: Any Submitter: Keyword: Artifact ID: Filter Reset Permalink						
2818701	consider addCacheIfAbsent() method	Open	2009-07-09	nobody	bradcupitlsu	5
2684466	apply replicateUpdateViaCopy=false to Put as well as Update	Closed	2009-03-12	nobody	kmashint	5
2644189	absolute memory configuration	Open	2009-02-27	gregluck	chbb77	5
2643509	Support for Pragma=no-cache and Cache-Control=no-cache	Open	2009-02-27	gregluck	nwiechmann	5
2604351	Add caching servlet filter with HTTP caching header support	Closed	2009-02-16	nobody	craigandrews	5
2376175	JGroups implementation of Bootstrap cache loader	Open	2008-12-02	nobody	rakesh_davanum	5
2141274	Memory only gets during disk spool flushes	Open	2008-10-02	nobody	petereddy	5
2128129	Add getCache() method to CacheStatistics	Open	2008-09-25	nobody	kennymacleod	5
2103844	Need ability to specify element TTL in milliseconds	Open	2008-09-11	nobody	naesc1976	5
2061030	Adding log information in Hibernate EhCache	Closed	2008-08-20	nobody	gronono	1

Fig. 2. A typical feature tracking system on SF.net

4 Research Methodology

We thoroughly reviewed prior measures used to operationalize the effectiveness of enhancement process. However we found most of the indicators to comprise only one or two simple measure; in addition, most of them seemed to be developed not as a

result of a rigorous scale development and validation process. Hence, we decided to apply Churchill 's (1979) guideline of scale development to operationalize the effectiveness of enhancement process. This guideline has been widely used in information system literature (Limayem et al. 2007). According to Limayem et al. (2007), this guideline has six steps (See Figure 3). Following the scale development process is explained as per each methodology stage.

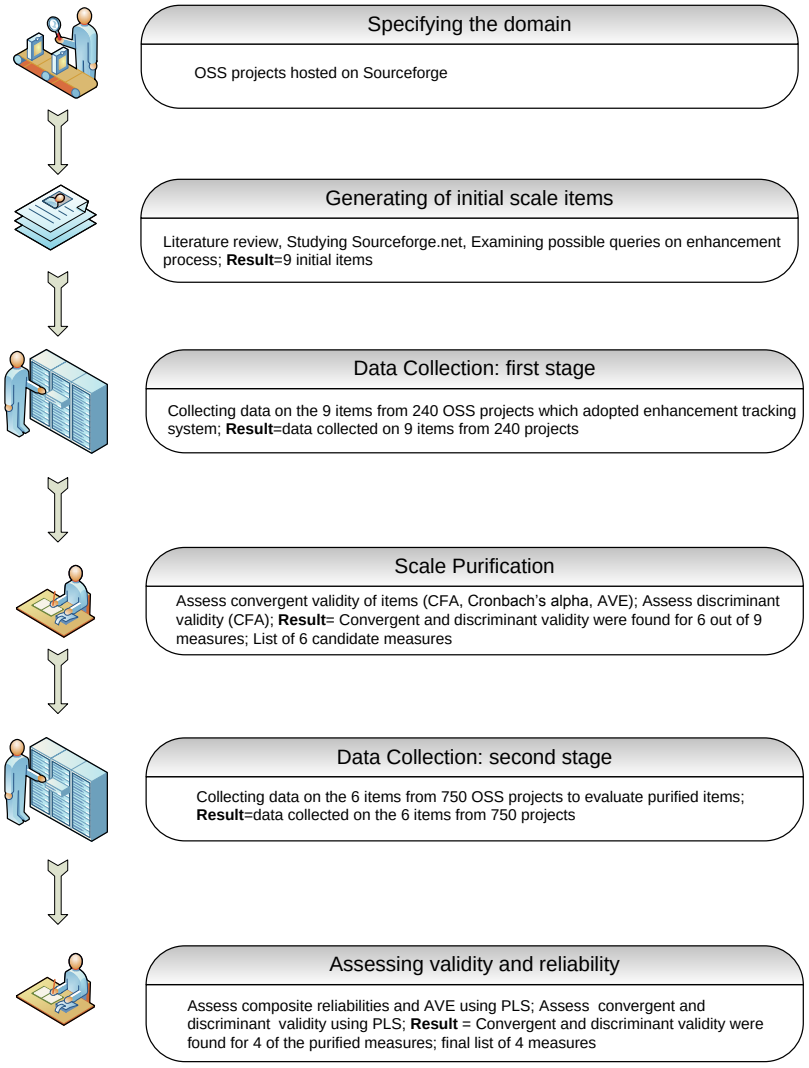


Fig. 3. Scale development process following Churchill (1979) guideline

5 Empirical Study

In what follows, we will explain data collection and analysis of the research as per each methodology stage in the Churchill 's (1979) guideline for scale development.

5.1 Stage 1 and 2

We specified OSS projects hosted on Sourceforge as the domain of the study. Next, 9 items were chosen using literature review and examining possible queries on feature tracking system of the projects hosted on Sourceforge.net. Table 1 shows an initial list of measures produced at stage 2.

Table 1. Initial measures for enhancement effectiveness

Item	Item Name	Source	Formula
X1	Total number of features submitted	(Paulson et al. 2004); (J. Long 2004); (Garousi 2009)	
X2	Number of feature report completed	(Rainer & Gale 2005)	
X3	Percentage of features report completed	(Stewart & Gosain 2006a)	$\frac{\text{Number of features completed}}{\text{Total number of features submitte}}$
X4	Percentage of features implemented	(Garousi 2009); (Stewart & Gosain 2006a)	$\frac{\text{Number of features implemented}}{\text{Total number of features submitte}}$
X5	Number of implemented features	(Garousi 2009)	
X6	Number of features submitted by team	Sourceforge.net	
X7	Total number of features assigned	Sourceforge.net	
X8	Percentage of features assigned	(Midha 2007)	$\frac{\text{Number of features assigned}}{\text{Total number of features submitte}}$
X9	Number of assigned features closed	Sourceforge.net	

5.2 Stage 3

Sourceforge divides OSS projects into various categories including: communication, database, desktop, education, formats and protocols, games and entertainments, Internet, multimedia, office/business, religion and philosophy, scientific/engineering, security, social science, software development, system, terminal, and text editor. In order to increase generalizability of the results, we decided to take sample from various categories. This was in line with prior research that studied projects hosted on OSS repositories including Sourceforge (e.g. Stewart et al. 2006; Long 2006; Stewart et al. 2005). However because collecting data from projects in all categories was beyond our limited time and resources, we chose to focus on three categories namely: communication, software development, and scientific/engineering.

In order to narrow down our sample, we then impose some restrictions as below: we exclude projects that have not had any file release within last 2 years (to discard inactive projects); we exclude projects whose development status is planning, pre-alpha, or alpha (because they normally don't have any software release); we exclude projects whose development status is mature (because they normally have not much activity and are already mature in terms of features and less activity is done on them for these purposes); we focus on those projects that have had at least 5 records in their feature-tracking system. A random sampling method was then used to select OSS projects. As a result, data on 240 projects was collected (80 projects from Communication category, 80 projects from Software development category, and 80 projects from Scientific/Engineering category). Data on all 9 items listed in Table 1 was collected from the 240 projects sampled for stage 3.

5.3 Stage 4

Factor analysis is a method primarily used for summarization and data reduction (Hair et al. 2006). Confirmatory factor analysis (CFA) is a kind of factor analysis that aims to measure to what extent a prior structure of indicators highly load on their associated constructs (Fabrigar et al. 1999). In this study, CFA was applied on the

indicators introduced in Table 1 to determine an overarching construct of enhancement process effectiveness. To do that, a step-by-step process suggested by Hair et al. (2006) was applied. PLS-Graph version 3.00 (Chin 2001) was used to conduct CFA (Sun & Zhang 2008; Chan et al. 2005; Lee et al. 2007). Bootstrap re-sampling procedure with the number of samples 200 was also employed to test the significance of all paths (Cotterman & Senn 1992).

Data on the 9 indicators collected from 240 OSS projects was used to do the analysis of stage 4. Confirmatory factor analysis was undertaken to examine loadings of each item on the construct of enhancement process effectiveness as well as the validity and reliability. The result of confirmatory factor analysis is presented in Table 2. Out of 9 items, 3 items were removed because their loadings were less than 0.7 (items removed: X3, X4, and X8). The reason why the items with a loading lower than 0.7 have been removed is that they show a low convergent validity with the other items. Next, we examined validity and reliability of the 6 remaining items through the same sample (See Table 3). According to Table 3, convergent validity exist among the 6 items since (1) all of them highly load on the construct (all loadings are higher than 0.7); (2) all of these reflective indicators were found to be significant (t-value>1.96 at Alpha level of 0.05); and also AVE value is greater than 0.5 (AVE=0.574). Our measurement model is also highly reliable with composite reliability greater than 0.7 (composite reliability=0.913). Moreover, the only way to examine discriminant validity here is to make sure that all items highly load on the construct under study (e.g. 0.7 or higher). According to Table 3, therefore, discriminant validity is also achieved because all the loadings are 0.7 or higher.

Table 2. The result of CFA on 9 items using data on 240 projects; (Composite Reliability = 0.913, AVE = 0.574); *Significant at Alpha level of 0.05

Item	Loading	Mean of sub-sample	Standard Error	T-value*	Decision
X1	0.8409	0.8220	0.0611	13.7562	Selected
X2	0.9586	0.9543	0.0130	73.7347	Selected
X3	0.3495	0.3522	0.0549	6.3713	Dropped

X4	0.3462	0.3499	0.0561	6.1750	Dropped
X5	0.9630	0.9589	0.0136	70.9727	Selected
X6	0.7663	0.7676	0.0647	11.8477	Selected
X7	0.9097	0.9120	0.0249	36.5262	Selected
X8	0.2727	0.2798	0.0659	4.1407	Dropped
X9	0.9408	0.9442	0.0125	75.2735	Selected

Table 3. The result of CFA on 6 items using data on 240 projects; (Composite Reliability = 0.967, AVE = 0.829); *Significant at Alpha level of 0.05

Item	Loading	Mean of sub-sample	Standard Error	T-value*
X1	0.8830	0.8724	0.0508	17.3659
X2	0.9678	0.9659	0.0107	90.2139
X5	0.9709	0.9697	0.0109	89.3054
X6	0.7834	0.7870	0.0512	15.3033
X7	0.9100	0.9142	0.0225	40.4848
X9	0.9359	0.9397	0.0136	69.0084

5.4 Stage 5

At stage 5, we sampled 750 projects for the Stage 5 of the methodology similar to the sampling for Stage 3. Out of 750 projects, 250 projects belonged to Communication category, 250 projects to Software development category, and 250 projects to Scientific/Engineering category. We collected data on all of the 750 projects. Tables 4 show some demographic information on projects sampled for stage 5 of the model.

Table 4. Distribution of the projects collected for stage 5 in terms of number of downloads

Number of downloads	Frequency	Percentage
50-1000	50	7%

1000-20,000	307	41%
20,000-100,000	221	29%
>100,000	172	23%
Total	750	100%

5.5 Stage 6

Data on the 6 indicators collected from 750 OSS projects was used to do the analysis of stage 6. Confirmatory factor analysis was run to assess validity and reliability of the items (See Table 5). According to Table 5, the loadings of 2 out of 6 purified items were higher than 0.7. Hence, X1 and X2 were removed and X5, X6, X7, and X8 were kept. We again ran CFA to assess validity and reliability of the 4 remaining items (See Table 6). As Table 6 shows, the 4 items show a high level of convergent validity because of three reasons. Firstly, all the loadings are 0.7 or higher. Secondly, all the 4 indicators are found to be significant at Alpha level of 0.05 (t-value>1.96). Thirdly, average variance extracted (AVE) has a high value of 0.779 (AVE>0.5 is accepted). Having high loadings on the construct of "enhancement process effectiveness" also is a cue of discriminant validity (0.7 or more). Our measurement model is also highly reliable with composite reliability greater than 0.7 (composite reliability=0.934).

As a result of the above-mentioned methodology, four items (X5, X6, X7, and X9) were extracted to measure enhancement process effectiveness namely: the number of features submitted by team members, the number of assigned features, the number of implemented features, the number of assigned features implemented.

Table 5. The result of CFA on 6 items using data on 750 projects; (Composite Reliability = 0.856, AVE = 0.533); *Significant at Alpha level of 0.05

Item	Loading	Mean of sub-sample	Standard Error	T-value*	Decision
X1	0.3317	0.5171	0.3002	1.1049	Dropped
X2	0.3017	0.5121	0.3282	0.9191	Dropped

X5	0.8270	0.8748	0.0652	12.6802	Kept
X6	0.7752	0.7729	0.0610	12.7030	Kept
X7	0.9246	0.9195	0.0275	33.6645	Kept
X9	0.9260	0.9241	0.0272	34.0579	Kept

Table 6. The result of CFA on 4 items using data on 750 projects; (Composite Reliability = 0.934, AVE = 0.779); *Significant at Alpha level of 0.05

Item	Loading	Mean of sub-sample	Standard Error	T-value*
X5	0.8226	0.8266	0.0732	11.2355
X6	0.7973	0.8093	0.0585	13.6222
X7	0.9497	0.9512	0.0110	86.5508
X9	0.9505	0.9558	0.0092	102.9905

6 Post-Hoc Analysis

After the completion of the above-mentioned methodology, we decided to run a post-hoc analysis on the five items that had not turned to be measures of enhancement effectiveness (i.e. X1, X2, X3, X4, and X8) to find out if they measure a different construct. Table 7 shows the result of this analysis. According to Table 7, items X3 and X4 are the only ones with a loading higher than 0.7. Looking at the definition of X3 (i.e. $\frac{\text{Number of features completed}}{\text{Total number of features submitted}}$) and X4 (i.e. $\frac{\text{Number of features implemented}}{\text{Total number of features submitted}}$), it appears that these two items are both measuring the latent construct of enhancement efficiency.

Table 7. The result of CFA on the 5 remaining items (post-hoc analysis)

Item	Loading	Mean of sub-sample	Standard Error	T-value	Decision
X1	0.434	0.365	0.153	2.83	Dropped
X2	0.630	0.592	0.105	5.97	Dropped
X3	0.876	0.890	0.034	25.65	Selected
X4	0.873	0.887	0.035	24.39	Selected
X8	0.514	0.536	0.080	6.36	Dropped

7 Numerical Example

As mentioned previously, a "latent variable" (here enhancement effectiveness) is a weighted composites of the manifest variables (e.g. the number of features implemented). PLS approach can provide an explicit factor score that represents the value of the latent variable for each case in the sample. To compute the factor score for a given case, PLS generates a weight matrix W for X, manifest variables matrix, such that F=XW, in which F is the corresponding factor score. These weights are computed so that each of them maximizes the covariance between responses and the corresponding factor scores. Given that, the enhancement effectiveness can be computed using an unstandardised factor score calculated by PLS.

Equation 1 shows a measurement model to calculate enhancement effectiveness based on the data used and the measures developed in this research. In Equation 1, enhancement effectiveness is the factor score computed by the PLS algorithm, the coefficients (e.g. 0.17) are the weights calculated by PLS, and X5, X6, X7, and X9 are the manifest variables proposed in this research to measure the effectiveness of the enhancement process (the number of features submitted by team members, the number of assigned features, the number of implemented features, the number of assigned features implemented). In the following, we present a hypothetical example that demonstrates practical use of the measurement model developed in this research.

*Enhancement Effectiveness = 0.17 * x5 + 0.25 * x6 + 0.27 * x7 + 0.32 * x9* **Equation (1)**

A high-technology manufacturing firm wants to select an open-source human resource management (HRM) software to automate its HRM processes. They decide to choose one of the options available on the Sourceforge.net, the largest OSS repository in the world.

After collecting initial information, the IT department selects 16 alternative HRM systems to choose from. Preliminary screening shows that five candidates, projects HR1 through HR5, out of the 16 fulfill a large proportion of their requirements. In order to choose one out of the five available software, they use a number of criteria suggested in OSS literature such as project activity, number of downloads, sponsorship and license. Subsequently, the IT team compares the five projects based on the criterion proposed in this paper, "enhancement effectiveness". The committee uses the measurement model proposed in this study and Equation 1 to calculate enhancement effectiveness for the alternatives. Data on the four measures introduced in this study is collected from the five projects, and then using the Equation 1 enhancement effectiveness is computed for each project (See Table 8). As Table 8 illustrates, project "HR4" has the most effective enhancement process among the others indicating that the project values user community's functional needs more than the other four. It should be noted that the criterion of enhancement effectiveness is just one of the several key decision criteria to evaluate and select a software, thus it should be used along with other decision criteria introduced in the literature such as project activity and license.

Table 8. Enhancement effectiveness for the five alternative HRM software

Project	Measures				Enhancement effectiveness score*	Rank
	X5	X6	X7	X9		
HR1	93	108	133	79	104	2
HR2	67	78	67	49	64	4
HR3	66	14	108	49	59	5

HR4	116	144	125	111	124	1
HR5	49	67	101	48	67	3

*Calculated using Equation 1.

8 Discussion and Conclusions

This research has taken the first step in OSS literature towards creating a reliable and valid measurement model for the construct of enhancement process effectiveness. An initial list of measures to make up the construct was generated through literature review and analyzing feature-tracking system of Sourceforge.net. Those measures were then validated and regarded reliable through a rigorous statistical methodology proposed by Churchill (1979). Undertaking 6 steps suggested by Churchill (1979), 9 initial items were reduced to 4 items through testing validity and reliability. The 4 items extracted by this research to measure the effectiveness of enhancement process are: number of features submitted by team, number of features implemented, number of assigned features implemented, and number of features assigned. Finally, a post-hoc analysis of the non-selected items showed that two items of percentage of features completed and percentage of features implemented can be used as measures of the efficiency of the enhancement process.

8.1 Implications for Theory

Our study has important implications for OSS research community. Firstly, the measurement model proposed in this study can be used by other researchers to examine relationships between the effectiveness of enhancement process and its potential antecedents and consequences (e.g. OSS project performance). Secondly, although the enhancement process has been studied and measured using one or two tasks involve in it (e.g. feature suggestion, feature assignment and etc.), to our knowledge, no empirical study has been reported to develop a measurement model for the enhancement as a process.

The current study has taken the first step towards constructing a reliable and valid measurement model for the effectiveness of the enhancement process in OSS projects. We call future research to study enhancement activities of OSS projects as a process rather than just taking its tasks into account.

Theory building and testing which is the ultimate goal of research are highly tied to measurement (Ghapanchi & Aurum 2011a). Therefore, taking measurement into account leads to advancement of information technology research (Ghapanchi & Aurum 2011a). The measurement model proposed in this research to gauge the effectiveness of the enhancement process is one step towards this goal.

8.2 Lessons for Practitioners

This study has key implications for OSS project administrators as well as organizations who want to adopt OSS software. Firstly, the validated measurement model suggested in this study can help OSS project administrators to better gauge the effectiveness of their enhancement processes. We advise OSS project administrators that traditional success measures introduced in the OSS literature (e.g. high download rate, or development interest) might not be adequate to have a comprehensive view of project success. Effectiveness of the enhancement process is another measure introduced in this paper as a potential indicator of success for OSS projects.

Secondly, the dimension of enhancement process effectiveness responds nicely to the concern of organizational users regarding open source software products in that responsiveness to customer needs is one of the most frequently cited concerns of IT practitioners adopting OSS (Golden 2004). The measurement model proposed in this study to compute the effectiveness of the enhancement process provides an evaluation criterion to organizational users who are interested in adopting OSS projects. Therefore, one criterion for these organizations to assess alternative OSS projects of the same type would be comparing the extent to which each software project's team operate the feature adding activity more effectively.

According to project management literature, performance is composed of effectiveness and efficiency (Crawford & Bryce 2003). Efficiency simply refers to the extent to which output is created out of a particular amount of input ($\text{Efficiency} = \frac{\text{Output}}{\text{Input}}$). In other words, efficiency means doing things in the most economical way (Nichols 1999). Effectiveness, on the other hand, means the capability of producing an effect. In other words, effectiveness means getting the right things done (Nichols 1999). The current study identified 4 and 2 items for measuring enhancement effectiveness and efficiency. A thorough examination of enhancement performance would include incorporating both sets of measures (i.e. effectiveness as well as efficiency measures).

References

Banker, R.D. & Slaughter, S.A., 2000. The Moderating Effects of Structure on Volatility and Complexity in Software Enhancement. *Information System Research*, 11, pp.219-240.

Chan, H.C., Teo, H.H. & Zeng, X.H., 2005. An evaluation of novice end-user computing performance: Data modeling, query writing, and comprehension. *Journal of the American Society for Information Science and Technology*, 56(8), pp.843-853.

Chengalur-Smith, I., Daniel, S. & Sidorova, A., 2010. Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. *Journal of the Association for Information Systems*, 11(11).

Chin, W.W., 2001. PLS-Graph Manual Version 3.

Churchill, G.A., 1979. A Paradigm for Developing Better Measures of Marketing Constructs. *Journal of Marketing Research*, 16(1), pp.64-73.

Cotterman, W.W. & Senn, J.A. eds., 1992. *Challenges and strategies for research in systems development*, Chichester: Wiley Series in Information Systems.

Crawford, P. & Bryce, P., 2003. Project monitoring and evaluation: a method for enhancing the efficiency and effectiveness of aid project implementation. *International Journal of Project Management*, 21(5), pp.363-373.

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

Crowston, K., Howison, J. & Annabi, H., 2006. Information systems success in free and open source software development: theory and measures. *Software Process: Improvement and Practice*, 11(2), pp.123-148.

Dalle, J. & Jullien, N., 2000. Windows vs. Linux: Some Explorations into the Economics of Free Software. In Proceedings of Acts of SSII.

Ducheneaut, N., 2003. *The Reproduction of Open Source Software Communities*. PhD dissertation. Berkeley: University of California.

Fabrigar, L.R. et al., 1999. Evaluating the Use of Exploratory Factor Analysis in Psychological Research. *Psychological Methods*, 4, pp.272-299.

Garousi, V., 2009. Evidence-Based Insights about Issue Management Processes: An Exploratory Study. In the International Conference on Software Process. Vancouver, Canada.

Ghapanchi, A.H. & Aurum, A., 2011a. Measuring the Effectiveness of the Defect-Fixing Process in Open Source Software Projects. In The 44th Hawaii International Conference on System Sciences. Hawaii.

Ghapanchi, A.H. & Aurum, A., 2011b. The impact of project licence and operating system on the effectiveness of the defect-fixing process in open source software projects. *International Journal of Business Information Systems*, (forthcoming).

Golden, B., 2004. *Succeeding with open source*, Addison-Wesley.

Hair, J.F. et al., 2006. *Multivariate Data Analysis*, New Jersey: Pearson Education Inc.

Herbsleb, J.D. & Mockus, A., 2003. An empirical study of speed and communication in globally distributed software development. *Software Engineering, IEEE Transactions on*, 29(6), pp.481-494.

Johnson, J., 2008. Available at: http://www.standishgroup.com/newsroom/open_source.php [Accessed April 16, 2008].

Kemerer, C.F. & Slaughter, S.A., 1997. Determinants of software maintenance profiles: an empirical investigation. *Journal of Software Maintenance: Research and Practice*, 9(4), pp.235-251.

Lee, M.K.O., Cheung, Christy M.K. & Chen, Z., 2007. Understanding user acceptance of multimedia messaging services: An empirical study. *Journal of the American Society for Information Science and Technology*, 58(13), pp.2066-2077.

Lientz, B. & Swanson, E., 1980. *Software Maintenance Management: A Study of the Maintenance of Computer Application Software in 487 Data Processing Organizations*, Acldison-Wesley.

Limayem, M., Hirt, S.G. & Cheung, C.M.K., 2007. How habit limits the predictive power of intention: The case of information systems continuance. *MIS Quarterly*, 31(4), pp.705-738.

Long, J., 2004. *Understanding the creation and adoption of information technology innovations: The case of Open Source software development and the diffusion of mobile commerce*. Texas: The University of Texas at Austin.

Long, Y., 2006. *Social structure, knowledge sharing, and project performance in open source software development*. Nebraska: The University of Nebraska - Lincoln.

Loshin, P., 2005. Something for Everyone! Open Source Isn't Just For Linux Users. *Computer Power User*, 5(5), pp.66-71.

Midha, V., 2007. *Antecedent to the success of open source software*. North Carolina: The University of North Carolina at Greensboro.

Mockus, Audris & Weiss, D., 2008. Interval quality: relating customer-perceived quality to process quality. In *the 30th international conference on Software engineering*. Leipzig, Germany: ACM, pp. 723-732.

Nichols, P., 1999. *An introduction to the logframe approach: course workbook & materials*, Melbourne: IDSS.

Notkin, D. & Griswold, W.G., 1988. Extension and software development. In The 10th International Conference on Software Engineering. Singapore, pp. 274-283.

O'Reilly, T., 1999. Lessons from open-source software development. *Commun. ACM*, 42(4), pp.32-37.

Open Source Initiative, 2005. Available at: http://www.opensource.org.

Paulson, J.W., Succi, G. & Eberlein, A., 2004. An Empirical Study of Open-Source and Closed-Source Software Products. *IEEE Transactions on Software Engineering*, 30(4), pp.246-256.

Rainer, A. & Gale, S., 2005. Evaluating the quality and the quantity of data on open source software projects. In *Proc. of the 1st International Conference on Open Source Systems*. Genova, Italy.

Raymond, E.S., 1999. The Cathedral and the Bazaar. *First Monday*, 3(3).

Stewart, K.J., Ammeter, A.P. & Maruping, L.M., 2005. A Preliminary Analysis of the Influences of Licensing and Organizational Sponsorship on Success in Open Source Projects. In the 38th Annual Hawaii International Conference on System Sciences.

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

Stewart, K.J. & Gosain, S., 2006b. The moderating role of development stage in free/open source software project performance. *Software Process: Improvement and Practice*, 11(2), pp.177-191.

Stewart, K.J., Ammeter, A.P. & Maruping, L.M., 2006. Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects. *Information Systems Research*, 17(2), pp.126-144.

Subramaniam, C., Sen, R. & Nelson, M.L., 2009. Determinants of open source software project success: A longitudinal study. *Decision Support Systems*, 46(2), pp.576-585.

Sun, H. & Zhang, P., 2008. An exploration of affect factors and their role in user technology acceptance: Mediation and causality. *Journal of the American Society for Information Science and Technology*, 59(8), pp.1252-1263.

Tiemann, M., 2009. *How Open Source Software Can Save the ICT Industry One Trillion Dollars per Year*, Available at: http://www.opensource.org/files/OSS-2009.pdf [Accessed March 3, 2010].