CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Scholarship@Western

Western University
Scholarship@Western

Electrical and Computer Engineering Publications
Electrical and Computer Engineering Department

2010

Improvement of Open Source Software Usability: An Empirical Evaluation from Developers Perspective

Arif Raza
National University of Pakistan, arif_raza@mcs.edu.pk

Luiz Fernando Capretz
University of Western Ontario, lcapretz@uwo.ca

Faheem Ahmed
Thompson River University, fahmed@tru.ac

Follow this and additional works at: https://ir.lib.uwo.ca/electricalpub


Part of the Software Engineering Commons

Citation of this paper:

@article{DBLP:journals/advse/RazaCA10, author = {Arif Raza and Luiz Fernando Capretz and Faheem Ahmed}, title = {Improvement of Open Source Software Usability: An Empirical Evaluation from Developers' Perspective}, journal = {Adv. Software Engineering}, volume = {2010}, year = {2010}, ee = {http://dx.doi.org/10.1155/2010/517532}, bibsource = {DBLP, http://dblp.uni-trier.de} }

Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2010, Article ID 517532, 12 pages
doi:10.1155/2010/517532

Research Article

Improvement of Open Source Software Usability: An Empirical Evaluation from Developers' Perspective

Arif Raza,[1] Luiz F. Capretz,[1] and Faheem Ahmed[2]

[1] *Department of Electrical & Computer Engineering, University of Western Ontario, London, ON, Canada N6A 5B9*
[2] *Faculty of Information Technology, United Arab Emirates University, P.O. Box 17551, Al Ain, UAE*

Correspondence should be addressed to Arif Raza, araza7@uwo.ca

Received 30 January 2010; Accepted 7 July 2010

Academic Editor: Hongyu Zhang

Copyright © 2010 Arif Raza et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

User satisfaction has always been important for software success whether it is Open Source Software (OSS) or closed proprietary software. Even though we do not presume that OSS always has poor usability, as there are examples of good usable open source software, it would still be agreed that OSS usability has room for further improvement. This paper presents an empirical investigation to study the impact of some key factors on OSS usability from developers' points of view. This is one of the series of four studies that we are conducting regarding improvement of OSS usability from OSS developers, users, contributors, and industry perspectives. The research model of this empirical investigation studies and establishes the relationship between the key usability factors from developers' perspective and OSS usability. A data set of 106 OSS developers from 18 open source projects of varied size has been used to study the research model. The results of this study provide empirical evidence that the studied key factors play a significant role in improving OSS usability.

1. Introduction

The term open source software refers to software equipped with licenses that provide existing and future users the right to use, inspect, modify, and distribute (modified and unmodified) versions of the software to others. It is not only the concept of providing "free" access to the software and its source code that makes OSS the phenomenon that it is, but also the development culture [1]. Open source is a software development method that makes source code available to a large community that participates in development by following flexible processes and communicating via the Internet [2]. The favorable acceptance of OSS products by business and the direct involvement of major IT vendors in OSS development have transformed OSS from a fringe activity, developed for public good, to a mainstream, commercially viable form [3]. The collaborative nature of the OSS culture makes use of a wide volunteer community, which conducts its development activities in a decentralized environment that has the direct result of effectively lowering production costs and improving the software quality [4].

The International Organization for Standardization and The International Electrotechnical Commission ISO/IEC 9126-1 [5] categorize software quality attributes into six categories: functionality, reliability, usability, efficiency, maintainability, and portability. In the standard, usability is defined as "*The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.*" However usability is probably the least addressed area in OSS research and development. In its early days it was generally believed that OSS is for the technically adept users; that resulted in a blurred boundary between developers and users. Thus, usability has never been on the top priority list of OSS developers. Nichols and Twidale [6] have come up with another reason why usability aspects are not so enthusiastically addressed as compared to functionality issues. According to them it is because the latter have more challenges and recognition factors from the community in them. Being voluntary in nature of work, in OSS projects usability problems are found less interesting and less challenging.

Hedberg et al. [7] point out that it is no more the case as in the past when OSS users were the codevelopers who

used to expect frequent crashing of the applications and bugs in the code. They feel that although so far usability in OSS has not been tested enough, now OSS community has started to realize that their target audiences are no longer their codevelopers only. As a result, OSS systems need to be designed, keeping in mind the requirements, expectations, and demands of a common nontechnical user. Viorres et al. [8] also highlight a general trend in OSS development that is instead of following software engineering (SE) practice of design, specifications, testing, and prototyping; most OSS systems follow a "*bottom-up*" approach where the focus is on the development on technical issues and individual components whereas the modeling of the whole system comes later; plus, user interface and related issues get a relatively lower priority.

This paper contributes in increasing the understanding of the effects of some key usability factors through empirical investigation that they play a vital role in improving OSS usability. A quantitative survey of developers involved in different OSS projects has been conducted and reported here. The survey has been used to analyze the conceptual model and hypotheses of the study. The results provide the evidence that the stated key factors play an important role towards the improvement of OSS usability.

In Section 2, we present the literature review that motivated this research work as well as helped in selecting the key factors for the study. Section 3 illustrates the research model and the hypotheses of this study. Section 4 explains the research methodology, data collection process, and the experimental setup in its first part, reliability and validity analysis of the measuring instrument in the second, and data analysis procedures in its third part. In Section 5, we present the hypotheses testing and the analysis of the results. It is followed by the discussion in Section 6 that also includes the limitations of the study. Finally the paper concludes in Section 7.

2. Literature Review

2.1. Research Motivation and Related Work. Empirical studies regarding open source quality assurance activities and quality claims are rare [9]. Koponen [10] discusses defect management and version management system as an integral part of OSS maintenance process. Aberdour [11] observes that the open source software model has led to the creation of significant pieces of software, and many of these applications show levels of quality comparable to closed source software development. Raymond [4] suggests that the high quality of OSS can be achieved due to high degree of peer review and user involvement in bug/defect detection. Generally a popular or active project means that the community in the OSS project are interacting constantly and providing feedback to activities such as defect identification, bug fixing, new feature request, and support requests for the further improvement.

Wayner [12] finds that developers contribute from around the world, meet face-to-face infrequently if at all, and coordinate their activity primarily by means of computer-mediated communications. Crowston and Scozzi [13] investigate the coordination practices for software bug fixing in OSS development teams and observe that task sequences are mostly sequential and composed of a few steps, namely, submit, fix, and close, and effort is not equally distributed among process actors; as a result, a few contribute heavily to all tasks while the majority just submit one or two bugs. Cubranic and Booth [14] discuss major issues of coordinating open source development projects, including collaborative communication mediums and configuration management tools. Mockus et al. [15] provide a comprehensive comparison of Apache against five commercial products in terms of developer participation, team size, productivity and defect density, and problem resolution. The Floss Survey [16] identifies many other reasons why developers are involved in OSS development, including becoming part of the open source community, promoting the open source mode of development, supporting the idea of "*free*" as an alternative to proprietary software, gaining a reputation, and having fun. In proprietary software, software quality testing is limited within a controlled environment and specific scenarios [17]. However, OSS development involves much more elaborate testing as OSS solutions are tested in various environments, by various skills and experiences of different programmers, and are tested in various geographic locations around the world [17–20]. According to Feller and Fitzgerald [21], OSS is characterized by active developers' community living in a global virtual boundary. OSS has emerged to address common problems of traditional software development that includes software exceeding its budget both in terms of time, and money, plus making the production of quick, inexpensive, and high quality reliable software possible. Earlier, OSS was more about operating systems and development tools. However lately, entertainment applications have also been developed. Independent peer review by codevelopers in OSS makes its quality presumably better and is also proved by their achievement of "*significant market share without any conventional marketing or advertising campaigns*."

Koppelman and Van Dijk [22] focus on the role of clients and users in projects, how to deal with different stakeholders who look at the product from a different perspective, how to communicate with them, and how to involve the real users and clients in the design process. They suggest that the designers should not simply rely on their own experiences and instincts. Golden et al. [23] support the idea of addressing usability issues at software architecture design level. They believe that separating usability concerns from functionality at architecture level in order to consider them at a later stage of testing does not work. Rather, this approach leads to extensive restructuring and even "*re-architecting of software systems*." They have come up with Usability-Supporting Architectural Pattern (USAP) that supports specific usability issues at architecture level. Although they themselves state that USAP is quite detailed and complex to imply as a whole, they conclude basing on a test case study that it is a beneficial tool to address specific usability issues in software architecture designs. They also observe that usability concerns could be better addressed if "*implications of usability heuristics for software design*" are made clear and explicit to the software designers.

Nichols and Twidale [24] feel that usability can hardly be considered a resolved issue in proprietary software environment that has better resources, let alone in OSS that has relatively less resources and where most work is done on voluntary basis with no monetary benefits or rewards. Another factor they identify is the lack of resources in OSS to achieve high quality, particularly in the context of usability, as compared to the closed proprietary software. To have more participation to analyze and fix usability bugs, what is required is to make usability reporting easier plus have less efforts and lower "*cultural, technical and usability barriers*." Unlike functionality bugs, where duplication in bug reporting does not help, a large number of usability bug reporting can help in prioritizing the usability-related errors, to be fixed. However it is required to have some way to speed up the discussion about usability-related issues and to have an easier and faster solution and consensus.

Çetin and Göktürk [25] observe that being a non-functional quality attribute, usability cannot be measured directly; it could be measured through users' feedback and cognitive walkthroughs. So far, there seems to be no metrics available for the OSS developers against which they could measure usability of their projects. A standardized user interface guideline may be developed by usability experts that the developers can adhere to, in order to have consistency and conformity in the designs.

Zhao and Deek [26] hypothesize that exploratory method is an effective way to impart such knowledge to the users so that they could be able to inspect and report usability errors in OSS and hence play their part in a better way, towards OSS usability improvement. A model has also been proposed by them to adapt the exploratory learning method for such purposes.

Hussain et al. [27], in their recent survey about the integration of agile methods and usability, conclude that the integration of agile methods with usability/user-centered design not only adds value to the adopted processes and to the teams of the respondents but also increases the satisfaction of the end-users of the product developed.

Hedberg et al. [7] observe that there is a lack of strict plan and design process in OSS environment as software development mainly relies on developers' skills. Advocating the early user feedback, the authors recommend to "*understand and specify the user, his/her work practice/tasks and the context of use, and carefully redesign the work practice/tasks based on the understanding, actively involve the user, gather early user feedback and iterate the design solution based on the user feedback*."

The above literature review and recommendations have played a motivating role in this study. We have been able to identify some usability factors and analyze and validate them empirically based on OSS developers' perception, as presented in the following sections.

2.2. Usability Factors: Literature Review of Concepts. Understanding *users' requirements* and expectations by OSS developers is an important issue that needs to be addressed seriously. Realizing the different intuitive approach of programmers from that of end-users, Pemberton [28] observes that while developing software they are normally satisfied with its usability and interface. Referring to the problems in an OSS environment, he states: "*the general public will have an itch they cannot scratch; the programmers will not have that itch, and so will not scratch it*." Nichols and Twidale [24] also identify that generally developers do not realize the needs and expectations of end-users that may lead to poor usability in OSS. They refer to Human Interface Guidelines (HIGs) that cannot only prevent such discussions but can also be considered as an authority on what will be done. Çetin and Göktürk [25] also realize that the main theme of OSS is the software development through collaboration and cooperation. Traditionally, OSS users have had technical and computer-oriented background and needed less effort to use OSS systems like Apache, GNU C compilers, bash shell, and so forth. However, as OSS has become more popular, more need is being felt to have usable systems. Koppelman and Van Dijk [22] stress that software developers should not simply rely on their own experiences and instincts. They should learn how to communicate with users in order to better understand their expectations.

The importance of HCI and *Usability Experts' Opinion* cannot be undermined. Big commercial organizations generally employ such experts to address usability issues in their projects. However their representation is generally missing in OSS projects probably due to voluntary work environment of OSS. Nichols and Twidale [6] identify why usability experts are not generally involved in OSS projects, mainly because there are fewer such experts in OSS world; they are not "*incentivised by the OSS approach in the way that many hackers are,*" and they do not find themselves "*welcomed into OSS projects*." Hedberg et al. [7] emphasize on the need of usability experts' contribution and show concern regarding their lack of participation in OSS development. They point out that OSS users may report usability-related problems and bugs but without formal training, neither the users nor the developers can fix them. An expert's opinion and suggestion is thus required; that is currently missing from the scene of OSS development. They propose the incorporation of usability guidelines and active participation of usability experts in OSS projects, possibly from the platforms of large commercial organizations, as they have also started participating in OSS development.

Incremental design approach, that is, introduction of advance features of software to users in an incremental way would make them more comfortable. Gaming softwares use this approach all the time and allow their users to face advance levels in an incremental and gradual fashion. OSS developers need to realize this fact, as well, that their target audiences may include novice users for whom the software application would be more adaptable if advance features are introduced in a gradual and progressive way. Yunwen and Kishida [29] highlight the need of modularized software system design to enable the end-users to encounter the difficulty levels gradually and progressively. They believe that modularized OSS system architecture design with progressive introduction of difficult and advance features would attract more users. Aberdour [11] also finds code modularity a convenient way to add new features in software.

It reduces code complexity and allows different programmers to extend the program by working in parallel and without interfering in others' work.

Usability aspects cannot be improved in OSS unless there are ways to test and measure them quantitatively. Çetin and Göktürk [25] highlight the importance of testing and measurement by stating: "*one cannot improve what is not measured*." Holzinger et al. [30] observe that "*the evaluation of consistency within an e-learning system and the ensuing eradication of irritating discrepancies in the user interface redesign is a big issue*." They have also come up with the Shadow Expert Technique (SET) to evaluate the consistency of the user interface and have applied it to a university learning management system. Nichols and Twidale [6] identify that fixing an interface needs an extra care so that it should not lead to inconsistency as "*a major success criterion for usability is consistency of design*." Usability problems are neither easier to specify nor very convenient to be fixed, particularly considering virtual boundaries of OSS where developers mostly do their work autonomously. In their other study, Nichols and Twidale [24] observe the bias in treating usability bugs as compared to functionality bugs that could crash the system. Usability issues, as expected, are more subjective in nature and more debatable as a user interface element may be more confusing to some people and less to others. Such issues could prolong the discussion of analyzing and fixing usability bugs. To have more participation to analyze and fix usability bugs, what is required is to make usability reporting easier plus have less efforts and lower "*cultural, technical and usability barriers*." Unlike functionality bugs, where duplication in bug reporting does not help, a large number of usability bug reportings can help in prioritizing the usability bugs to be fixed. Hedberg et al. [7] suggest evaluation methods under the guidance of usability experts, usability testing, and bug reporting. They feel the need of in-depth empirical research to understand the challenges related to usability and quality assurance in OSS. Viorres et al. [8] also highlight a few OSS usability issues such as to improve bug reporting facilities in software, to improve the analysis procedure of usability errors by OSS community through application of human computer interaction (HCI) principles, and to support argumentation to resolve such issues.

As a long-term solution, students of the Software Engineering and Computer Science disciplines should be taught how to address user-centric issues in their software development projects to increase their understanding of the users' point of view. They should be encouraged to appreciate the fact that finding a solution to a particular programming problem is not the ultimate goal. They should rather come up with design that could meet the expectations of end-users. Faulkner and Culwin [31] observe that HCI and Software Engineering educators have always been in different camps. Although the growth of HCI in terms of books and as a subject taught in computer science courses is the recognition of importance of HCI, they suggest that there is a need of more interaction between HCI and SE by adopting HCI as the underlying principle to the systems development. According to them, the aim of usability engineering education must be to ensure that effectiveness, efficiency, and user satisfaction are present in software. The guidance from HCI specialist needs be provided to the software developer in a useful form, which is only possible through the unification of knowledge and vocabulary of both. However, Rosson et al. [32] realize that the main challenge in teaching usability engineering is to come up with realistic projects for the students, such that meaningful issues could be addressed in a manageable time of a semester.

Markov [33] argues that usability is about "*total user experience*," not only about the user interface, as it is commonly but incorrectly assumed. It should be involved in all the phases of the product such as installation, use, and maintenance. Although it is not the case that every OSS must have a poor user interface, usability of OSS projects requires improvement, in general. In their research work, Nichols and Twidale [6] observe that OSS is growing and has developed a repute of being reliable, efficient, and functional. However, still common novice computer user prefers to use proprietary software for many reasons: their better usability is one of them. They talk about usability from applications like word processors and web mail servers which are basically aimed at serving a common user. They also realize that, considering fewer resources of OSS, it could take long for an OSS project to be mature and comparable with closed proprietary software. Another point they make is that, in OSS culture, coding starts earlier and refinement of design depends on constant reviews. They advocate that to improve OSS usability, designing of interface should be done before the start of the coding, to keep it consistent. Viorres et al. [8] refer to various reasons why software developers go for OSS. These include educational reasons, reusability, and developing reputation. However, they highlight concerns about software usability and complexity in installation and maintenance of OSS development tools, their nonadherence to backwards compatibility, and limited documentation. Hedberg et al. [7] propose the adaptation of proven methods in OSS environment to ensure higher quality and address usability issues. Holzinger et al. [34] discuss a user-centered system developed at the clinical department of dermatology at the Medical University Hospital in Graz. The system not only improved the existing system but also helped elderly people to overcome their computer fear.

3. Research Model and the Hypotheses

In this paper, we present a research model to analyze the relationship between the key usability factors and the open source software usability. This work empirically investigates the association between these key usability factors and the OSS Usability. The theoretical model to be empirically tested in this paper is shown in Figure 1. Our aim is to investigate the answer to the following research question:

How OSS developers can improve software usability?

There are five independent and one dependent variables in this research model. The five independent variables are called

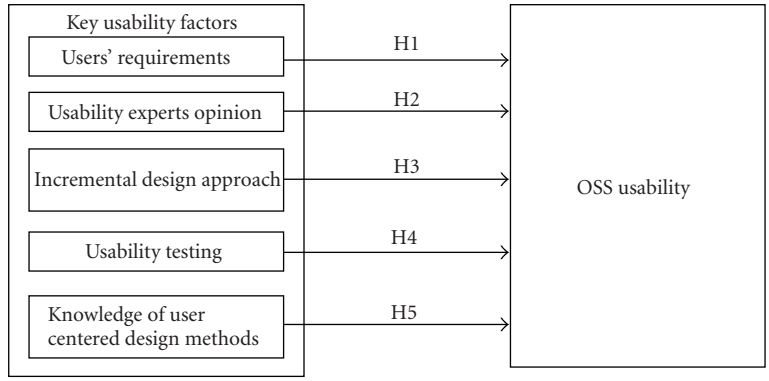

Figure 1: Research Model.

"key usability factors" in the rest of this paper. They include users' requirements, usability experts' opinion, incremental design approach, usability testing, and knowledge of user-centered design methods. The dependent variable of this study is the OSS usability. The multiple linear regression equation of the model is as follows:

$$\text{OSS Usability} = f_0 + f_1 v_1 + f_2 v_2 + f_3 v_3 + f_4 v_4 + f_5 v_5, \quad (1)$$

where f_0, f_1, f_2, f_3, f_4, and f_s are the coefficients and v_1, v_2, v_3, v_4, and v_s are the five independent variables. In order to empirically investigate the research question we hypothesize the following.

(H1) understanding users' requirements by the software developers is positively related with improving usability in OSS.

(H2) seeking usability experts' opinion by the software developers is positively related with improving usability in OSS.

(H3) incremental approach in OSS design plays a positive role in improving usability in OSS.

(H4) usability testing by project managers/software developers has a positive impact on usability in OSS.

(H5) knowledge of user-centered design (UCD) methods is positively related with improving software.

4. Research Methodology

Open source software projects deal with different categories of applications like communications, database, desktop environment, education, financial, games/entertainment, networking, and so on. We sent personalized emails to OSS developers of different projects on sourceforge.net. The projects differed in size and range from small to large scale. However, we selected the projects having activity of 90% and more. We sent our questionnaire to the OSS developers working on the projects in the categories of Database (118), desktop environment (127), development (135), testing (83), communications (104), games/entertainment (309), education (309), financial (236), and enterprise (35) as shown in Figure 2.

We assured the participants that our survey neither required their identity nor would it be recorded. However to support our analysis of data in terms of experience of the developers and the project size, they have been working on, we asked them to share with us their OSS development experience and their development team size. These two questions were optional for the participants to respond to unlike the questions related to OSS usability which were mandatory to respond to in the survey. We received 106 responses altogether and 104 of them chose to respond to these two questions. 63 of the 104 respondents had less than or equal to 5 years of OSS development experience; 31 had more than 5 years but less than or equal to 10 years of experience whereas 10 of the respondents stated that they had more than 10 years of experience in OSS development as reflected in Figure 3.

In the survey, 56 respondents had less than or equal to 10 team members as developers in their project, 27 had more than 10 but less than or equal to 20 team members as developers, and 21 had more than 20 members in their development team as represented graphically in Figure 4.

The above statistics have been presented to reflect the experience of the respondents as well as the size of the OSS project they belong to.

4.1. Data Collection and the Measuring Instrument. In this study, we have collected data on the key usability factors and the perceived level of OSS usability improvement. The questionnaire presented in the appendix requires respondents to indicate the extent of their agreement or disagreement with statements using a five-point Likert scale. The Likert scale ranges from "strongly agree" (1) to "strongly dis-agree" (5) for all items associated with each variable. For each of the independent variables as well as the dependent variable, four statements are presented. These statements elaborate the specific key factor and its related issues. The statements are designed to collect measures on the extent to which the variable is practiced within each project. We have thus used twenty separate items to measure the independent variables and four items to measure OSS developers' point of view regarding OSS usability improvement. Although not much work on such lines is available, we have reviewed

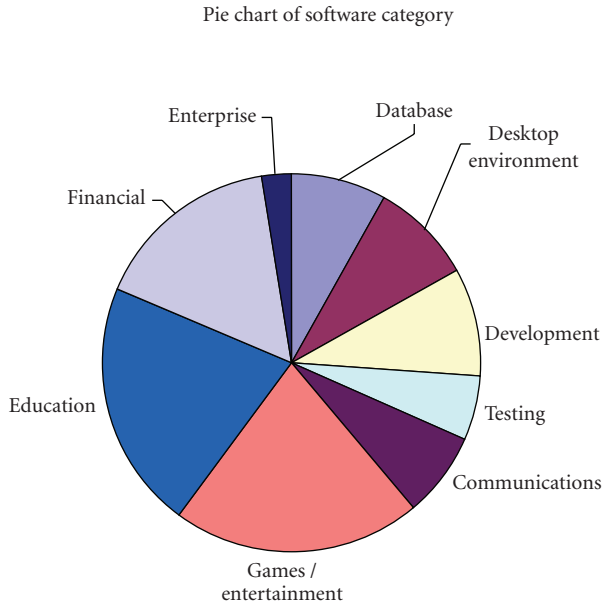

Figure 2

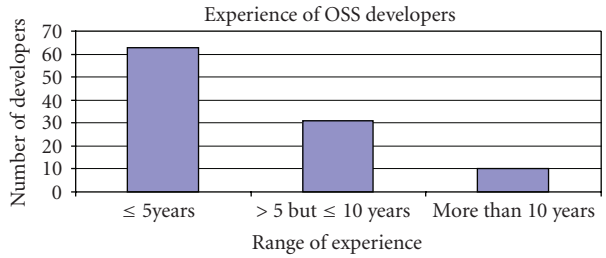

Figure 3

previous researches on the subject of OSS usability, so that a comprehensive list of measuring factors could be constructed.

4.2. Reliability and Validity Analysis of Measuring Instrument. The two integral features of any empirical study are reliability, which refers to the consistency of the measurement, and the validity, that is the strength of the inference between the true value and the value of a measurement. For this empirical investigation, we have used the most commonly used approaches in empirical studies to conduct reliability and validity analysis of the measuring instruments. The reliability of the multiple-item measurement scales of the five usability factors is evaluated by using internal-consistency analysis, which is performed using coefficient alpha [35]. In our analysis, the coefficient alpha ranges from 0.55 to 0.67 as shown in Table 1. van de Ven and Ferry [36] state that a reliability coefficient of 0.55 or higher is satisfactory, and Osterhof [37] suggests that 0.60 or higher is satisfactory. Therefore, we conclude that the variable items developed for this empirical investigation are reliable.

Campbell and Fiske [38] state that convergent validity occurs when the scale items are correlated and move in the same direction in a given assembly. The principal component

Table 1: Coefficient alpha and principal component analysis (PCA) of variables.

Usability factors	Item no.	Coefficient α	PCA eigen value
Users requirements	1–4	0.67	2.19
Usability experts opinion	5–8	0.64	1.22
Incremental design approach	9–12	0.55	1.08
Usability testing	13–16	0.55	0.99
Knowledge of UCD methods	17–20	0.59	1.05

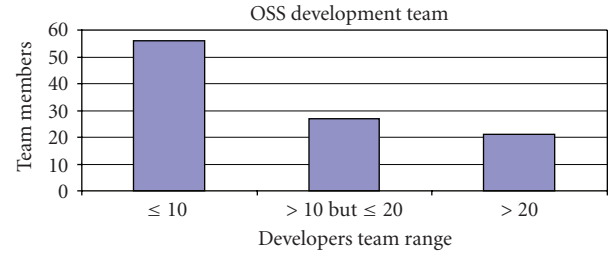

Figure 4

analysis [39] is performed for all five key usability factors and reported in Table 1. We have used eigenvalue [40] as a reference point to observe the construct validity using principal component analysis. In this paper, we have used eigenvalue-one criterion, also known as Kaiser Criterion [41, 42], which means any component having an eigenvalue greater than one is to be retained. Eigenvalue analysis reveals that four out of five variables completely form a single factor whereas eigenvalue for the usability testing is 0.99, that is very close to the threshold of 1.0. Therefore, the convergent validity has been regarded as sufficient.

4.3. Data Analysis Procedure. We have analyzed the research model and the significance of hypotheses H1–H5 through different statistical techniques in three phases. In phase I, we have used normal distribution tests and parametric statistics whereas, in phase II, nonparametric statistics have been implemented. Due to a relatively small sample size, both parametric as well as nonparametric statistical approaches are used to reduce the threats to external validity. As our measuring instrument has multiple items for all the five independent variables as well as the dependent variable (refer to the appendix), their ratings by the respondents are summed up to get a composite value for each of them. Tests are conducted for hypotheses H1–H5 using parametric statistics by determining the Pearson correlation coefficient. For nonparametric statistics, tests are conducted for hypotheses H1–H5 by determining the Spearman correlation coefficient. To deal with the limitations of a relatively small sample size and to increase the reliability of the results, hypotheses H1–H5 of the research model are tested using partial least square (PLS) technique in phase III. According to Fornell and Bookstein [43] and Joreskog

and Wold [44], the PLS technique is helpful in dealing with issues such as complexity, nonnormal distribution, low theoretical information, and small sample size. The statistical calculations are performed using minitab-15. (*Minitab is a statistics software package (see* http://en.wikipedia.org/wiki/List_of_statistical_packages*) and is often used in conjunction with the implementation of Six Sigma (see* http://en.wikipedia.org/wiki/Six_Sigma*), CMMI (see* http://en.wikipedia.org/wiki/CMMI*), and other statistics-based process improvement methods. Minitab is available in 7 different languages.*)

5. Hypotheses Testing and Results

5.1. Phase I. To test hypotheses H1–H5 of the research model (shown above in Figure 1), parametric statistics is used in this phase by examining the Pearson correlation coefficient between individual independent variables (key usability factors) and the dependent variable (OSS usability). The results of the statistical calculations for the Pearson correlation coefficients are displayed in Table 2. It is to be noted that, *"In statistical* (*see* http://en.wikipedia.org/wiki/Statistics) *hypothesis testing* (*see* http://en.wikipedia.org/wiki/Hypothesis_test), *the P-value is the probability* (*see* http://en.wikipedia.org/wiki/Probability) *of obtaining a test statistic.* (*see* http://en.wikipedia.org/wiki/Test_statistic) *The lower the P-value, the less likely the result is if the null hypothesis is true, and consequently the more "significant" the result is, in the sense of statistical significance* (*see* http://en.wikipedia.org/wiki/Statistical_significance) *"* [45].

The Pearson correlation coefficient between users' requirements and OSS usability is found positive (0.264) at $P < .05$ and hence justified hypothesis H1. The Pearson correlation coefficient of 0.084 is observed at $P = .393$ between usability experts' opinion and OSS usability and hence found insignificant at $P < .05$. Therefore hypothesis H2 that deals with usability experts' opinion and OSS usability is rejected. Hypothesis H3 is accepted based on the Pearson correlation coefficient (0.274) at $P < .05$, between the incremental design approach and OSS usability. The positive correlation coefficient of 0.338 at $P < .05$ is also observed between the OSS usability and usability testing which meant that H4 is accepted. Hypothesis H5 is found significant too and thus accepted after analyzing the Pearson correlation coefficient of 0.439 at $P < .05$ between knowledge of UCD methods and OSS usability. Hence, as observed and reported above, hypotheses H1, H3, H4, and H5 are found statistically significant and are accepted whereas H2 is not supported and is therefore rejected.

5.2. Phase II. Nonparametric statistical testing is conducted in this phase by examining Spearman correlation coefficient between individual independent variables (key usability factors) and the dependent variable (OSS usability). The results of the statistical calculations for the Spearman correlation coefficient are also displayed in Table 2.

The Spearman correlation coefficient between users' requirements and OSS usability is found positive (0.480) at $P < .05$ and hence justified hypothesis H1. For hypothesis H2, the Spearman correlation coefficient of 0.122 is observed with $P = .213$; hence at $P < .05$ no significant relationship is found between usability experts' opinion and OSS usability in this test as well. Hypothesis H3 is accepted based on the Spearman correlation coefficient (0.420) at $P < .05$, between the incremental design approach and OSS usability. The positive Spearman correlation coefficient of 0.390 at $P < .05$ is also observed between the OSS usability and usability testing, which means that H4 is accepted. Hypothesis H5 is found significant too and thus accepted after analyzing the Spearman correlation coefficient of 0.485 at $P < .05$ between knowledge of UCD methods and OSS usability.

Table 2: Hypotheses testing using parametric and nonparametric correlation coefficients.

Hypothesis	Usability factor	Pearson correlation coefficient	Spearman correlation coefficient
H1	Users requirements	0.264*	0.480*
H2	Usability experts opinion	0.084**	0.122**
H3	Incremental design approach	0.274*	0.420*
H4	Usability testing	0.338*	0.390*
H5	Knowledge of UCD methods	0.439*	0.485*

* Significant at $P < .05$. ** Insignificant at $P > .05$.

Hence, as observed and presented above, H1, H3, H4, and H5 are found statistically significant and are accepted whereas H2 is not supported and hence rejected in nonparametric analysis, as well.

5.3. Phase III. In order to do the cross-validation of the results obtained in Phase I and Phase II, partial least square (PLS) technique has been used in this phase of hypotheses testing. The direction and significance of hypotheses H1–H5 are examined. In PLS, the dependent variable of our research model, that is, OSS usability, is placed as the response variable and independent key usability factors as the predicate. The test results that contain observed values of path coefficient, R^2, and F-ratio are shown in Table 3. The *"users' requirements"* is observed to be significant at $P < .05$ with path coefficient of 0.302, R^2 of 0.070, and F-ratio of 7.782. Usability experts' opinion has path coefficient of 0.129 with R^2 of 0.007, and F-ratio of 0.737 and is found insignificant at $P < .05$ (with observed $P = .393$). Incremental design approach is observed to have the same direction as proposed in hypothesis H3 with path coefficient 0.244, R^2 0.075, and F-ratio 8.429 at $P < .05$. Usability testing is also found in conformance with hypothesis H4 with observed values of path coefficient of 0.310, R^2 of 0.114, and F-ratio of 13.412 at $P < .05$. And finally knowledge of UCD methods (path coefficient: 0.446, R^2: 0.193, and F-ratio: 24.888 at $P < .05$) is also found in accordance with H5. Hence in this phase, like in phase I and phase II, hypothesis H2 that deals with

Table 3: Hypotheses testing using PLS regression.

Hypothesis	Usability factor	Path coefficient	R^2	F-ratio
H1	Users' requirements	0.302	0.070	7.782*
H2	Usability experts opinion	0.129	0.007	0.737**
H3	Incremental design approach	0.244	0.075	8.429*
H4	Usability testing	0.310	0.114	13.412*
H5	Knowledge of UCD methods	0.446	0.193	24.888*

* Significant at $P < .05$. ** Insignificant at $P > .05$.

Table 4: Multiple linear regression analysis of the research model.

Model coefficient name	Model coefficient	Coefficient value	t-value
Users' requirements	f_1	0.277	2.800*
Usability experts opinion	f_2	−0.006	−0.045**
Incremental design approach	f_3	0.116	1.218*
Usability testing	f_4	0.111	1.097*
Knowledge of UCD methods	f_5	0.355	3.740*
Constant	f_0	1.796	1.003*

* Significant at $P < .05$. ** Insignificant at $P > .05$.

usability experts' opinion and OSS usability is not found to be statistically significant at $P < .05$.

5.4. Testing of the Research Model. The multiple linear regression equation of our research model is depicted by (1). The purpose of research model testing is to provide empirical evidence that our key factors play a significant role in improving open source software usability. The testing process consists of conducting regression analysis and reporting the values of the model coefficients and their direction of association. OSS usability is placed as response variable and key factors as predicators. Table 4 displays the regression analysis results of the research model. The path coefficient of four out of five variables, users' requirements, incremental design approach, usability testing, and knowledge of user-centered design methods, is found positive, and their t-statistics are also observed statistically significant at $P < .05$. The path coefficient of usability experts' opinion is found negative. Negative t-statistics and $P > .05$ make usability experts' opinion statistically insignificant in this research model. R^2 and adjusted R^2 of overall research model are observed as 0.294 and 0.259, respectively, with an F-ratio of 8.335 significant at $P < .05$.

6. Discussion

The use of open source software has increased in the recent years, mainly due to the easy access and availability of the Internet. Although it has been a common belief that OSS is popular with technically adept users, which results in a blurred boundary between its developers and users, the users of OSS are no more limited to this category alone: novice and nontechnical users are using OSS as well than ever before. As more and more people use OSS, usability and its related issues need to be addressed more seriously. Through empirical investigation, this research enables the OSS developers and project managers to realize the relationship of key factors of our research model and the OSS usability process. The results provide the empirical evidence and support for the theoretical foundations that the stated key factors play an important role in the institutionalization of usability within an OSS project.

Users' satisfaction plays a major role in the success of software, whether it is an open source or closed proprietary software. The more satisfied a target user is, particularly in application software, the more acceptability the software would get. And we believe that a path to achieve users' satisfaction goes through understanding their expectations and requirements. OSS is no longer a "*reserved arena*" for technically adept users; novice and nontechnical users from all over the world use open source software as well. As Koppelman and Van Dijk [22] identify that in order to know end-users' requirements and expectations, there is a need of more communication between the software developers and their target users, instead of relying on the former's instincts. 87% of our respondents support this observation that getting users' requirements helps in improving OSS usability. In our empirical investigation too, we have found a positive relationship between users' requirements and the OSS usability. *Users' requirements* could thus be taken by OSS developers' community as a key issue to improve usability of their projects.

Role of HCI and usability experts cannot be undermined in software development. This becomes more important in application software, where end-users are the direct audiences. In proprietary software development, particularly in big organizations, such experts are hired to have their valuable opinion to make their software more usable and acceptable to end-users. Considering voluntary nature of work and fewer resources in OSS development, we do not find such experts actively playing their role in OSS field. It might be because they do not find themselves "*welcomed into OSS projects*" as identified in [6]. Anyhow, our statistical findings do not significantly support the positive association of usability experts' opinion and OSS usability. In the parametric and nonparametric statistical analysis as well as in PLS and multiple regression testing, the results were not supported by a significant statistical level of confidence (refer to Tables 2, 3, and 4). Therefore, we conclude that our study has not been able to prove a positive association of *usability experts' opinion* and OSS usability.

Gradual and incremental introduction of advance features in software makes users feel more comfortable. It increases the acceptability and adaptability of the application. Yunwen and Kishida [29] advocate the modularized

system design, such that users encounter the difficulty levels gradually and progressively. Gaming softwares use incremental approach in their design all the time. Only after user completes one level, s/he is encouraged to move on to the more difficult levels. Using same approach in all designs can make software more accommodating for a common novice user. 69% of the respondents in our survey agree that gradual introduction of advance features in software would enhance its adaptability. Our research study has also found a positive impact of incremental design approach on OSS usability. We thus have considered *incremental design approach* as a key attribute towards improving OSS usability.

Software testing is an integral part of software life cycle. Holzinger [46] emphasizes the earlier usability testing in software life cycle and maintains that *"the earlier critical design flaws are detected, the more likely they can be corrected."* However, being a subjective matter, software usability cannot be directly measured. Furthermore, difficulty being faced by users in reporting errors makes the situation worse. Nichols and Twidale [24] refer to such difficulties faced by the users in reporting usability bugs by stating "*Difficulties that a User May Experience with a Graphical User Interface May Not be Easy to Describe Textually*." 72% of the respondents in our survey agree that formal usability testing should be an integral part of software testing procedure. The findings of our empirical investigation also confirm a positive association between usability testing and OSS usability. We thus take *usability testing* as a key issue to improve usability of OSS projects.

Students of computer science and software engineering being the future software managers and developers need to understand the importance of usable systems more seriously. They should be encouraged to realize that coming up with a programming solution to a problem is not the ultimate goal; any system developed should meet users' expectations. The earlier they would incorporate the usability features in their designs, the better it would be for their projects, from maintenance point of view, too. We also have found a positive impact of knowledge of user-centered design methods on OSS usability, in our empirical investigation. Not a single respondent of our survey disagreed with our question that "*Computer Science/Software Engineering students (future software developers) must learn how to incorporate usability aspects in their software designs*." This could be a part of long-term solution to improve software usability and would be equally beneficial to both OSS and closed proprietary software organizations. We thus take *knowledge of UCD methods* as one of the key factors to improve usability of OSS projects.

6.1. Limitations of the Study and Threats to External Validity. Surveys, experiments, metrics, case studies, and field studies are examples of empirical methods used to investigate both software engineering processes and products [47]. Empirical investigations are subject to certain limitations which is the case of this study as well.

Threats to external validity are conditions that limit the researcher's ability to generalize the results of his/her experiment to industrial practice [48], which is the case with this study. Specific measures have been taken to support external validity; for example, a random sampling technique is used to select the respondent from the population in order to conduct experiments. We retrieve the data from the most active and well-known OSS reporting website, sourceforge.net, which has huge amount of projects listed.

The increased popularity of empirical methodology in software engineering has also raised concerns on the ethical issues [49, 50]. We have followed the recommended ethical principles to ensure that the empirical investigation conducted and reported here would not violate any form of recommended experimental ethics. Another aspect of validity is concerned with whether or not the study reports results that correspond to previous findings. First of all is the selection of independent variables in this work. We have used five independent variables to relate with the dependent variable of OSS usability. We realize that there could be other key factors that influence OSS usability, but we have kept the scope of this study within open source software as well as OSS developers' point of view. Some other contributing factors like OSS development culture, less resources of OSS projects as compared to resources of closed proprietary software projects developed in big organizations, voluntary involvement of developers in OSS projects, and so forth have not been considered in this study. Another limitation of this study is a relatively small sample size. Although we sent our survey to notable number of OSS developers subscribed to 18 different categories of software, we received only 106 responses. The relatively small sample size in terms of number of respondents has a potential threat to the external validity of this study. Although the proposed approach has some potential to threaten external validity, we have followed appropriate research procedures by conducting and reporting tests to improve the reliability and validity of the study, and certain measures were also taken to ensure the external validity.

7. Conclusion

In this paper, we empirically investigate the effect of key factors on OSS usability and find answer to the research question stated in this investigation. Results of this empirical investigation exhibit that the stated key factors of our research model assist in improving OSS usability. Empirical results of this study strongly support the hypotheses that *users' requirements, incremental design approach, usability testing, and knowledge of UCD methods* are positively associated with the usability of an OSS project. However we could not find any significant statistical support for *usability experts' opinion* on OSS usability.

The study conducted and reported here will enable OSS development teams to better understand the effectiveness of the relationships of the stated key factors and usability of their projects. The OSS developers need to take into consideration multiple key usability factors to improve usability aspect of software in general and their projects in particular. Currently we are working on to develop a maturity model

to assess the usability of open source software projects. This empirical investigation provides us some justification to consider these key factors as a measuring instrument. This study is one of the series of four studies that we are conducting in parallel, regarding OSS usability from users, contributors, and software industry's points of view.

Appendix

Key Usability Factors from OSS Developers' Point of View (Measuring Instrument)

Users' Requirements:

(1) users' requirements help in increasing software usability;

(2) understanding community expectations by the code contributors support the software usability;

(3) taking community feedback before and after formal release of every major version of software is vital in improving software usability;

(4) recording users' profile is crucial in understanding their requirements and expectations and hence supports OSS usability;

Usability experts' Opinions (Usability experts are those personnale who have formal training and expertise in usability and HCI).

(5) usability features can better be incorporated if usability experts' opinions are taken during every life-cycle phase;

(6) seeking usability experts' opinions will compromise freedom of OSS developers;

(7) OSS designs based on usability experts' opinions end up with GUI having standard usability norms but lacking innovation;

(8) usability experts' opinions are equally important and applicable for OSS as they are for closed proprietary software;

Incremental Design Approach (Introduction of advanced features of software to users in an incremental way).

(9) incremental increase in the difficulty level of software always makes user feel more comfortable;

(10) a novice user needs only basic features of software;

(11) gradual introduction of advance features will enhance adaptability of the software; however it is not always possible;

(12) every user should explore advance features of software gradually;

Usability Testing.

(13) formal usability testing should be an integral part of software testing process;

(14) although software success is dependent on users' response, usability-related bugs mostly reflect personal demands;

(15) I will fix the usability-related bug only if I am convinced that the reported bug is worth fixing;

(16) usability bugs reflect users' requirements and expectations; therefore they need to be fixed on priority;

Knowledge of User-Centered Design Methods ("UCD processes focus on users through the planning, design and development of a product" [51]).

(17) computer science/software engineering students (future software developers) must learn how to incorporate usability aspects in their software designs;

(18) designing of user friendly GUI is an art not every programmer can learn;

(19) CS/SE curriculum needs to be revised to implant importance of usercenteredness in software designs;

(20) poor usability of OSS systems is not due to lack of knowledge of user-centered design methods; it is because they are not implemented and systems are not designed with people in mind.

OSS Usability ("The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions" [5]).

(1) improving OSS usability will result in reducing the overall cost, bug reporting and defects of the software;

(2) one reason of poor OSS usability is because it is developed free; OSS designers should have some incentive (e.g., award or recognition) to look for to;

(3) successful software project means usable software with satisfied users;

(4) software having improved usability and adaptability for less technical and novice users will end up benefiting all users.

References

[1] E. S. Raymond, *The Cathedral and the Bazaar*, O'Reilly, Sebastopol, Calif, USA, 1999.

[2] A. G. Koru and J. Tian, "Defect handling in medium and large open source projects," *IEEE Software*, vol. 21, no. 4, pp. 54–61, 2004.

[3] B. Fitzgerald, "The transformation of open source software," *MIS Quarterly*, vol. 30, no. 3, pp. 587–598, 2006.

[4] E. S. Raymond, *The Cathedral and the Bazaar*, O'Reilly, Sebastopol, Calif, USA, 2001.

[5] International Standard ISO/IEC 9126-1, *Software Engineering—Product Quality—Part 1: Quality Model*, 1st edition, 2001.

[6] D. M. Nichols and M. B. Twidale, "The usability of open source software," *First Monday*, vol. 8, no. 1, 2005.

[7] H. Hedberg, N. Iivari, M. Rajanen, and L. Harjumaa, "Assuring quality and usability in open source software development," in *Proceedings of the 1st International Workshop on Emerging Trends in FLOSS Research and Development*, Washington, DC, USA, May 2007.

[8] N. Viorres, P. Xenofon, M. Stavrakis, E. Vlachogiannis, P. Koutsabasis, and J. Darzentas, "Major HCI challenges for open source software adoption and development," in *Proceedings of the 2nd International Conference on Online Communities and Social Computing (OCSC '07)*, D. Schuler, Ed., pp. 455–464, Beijing, China, July 2007.

[9] R. Glass, "Is open source software more reliable? An elusive answer," *The Software Practitioner*, vol. 11, no. 6, 2001.

[10] T. Koponen, "Life cycle of defects in open source software projects," in *Proceedings of the 2nd International Conference on Open Source Systems*, pp. 195–200, 2006.

[11] M. Aberdour, "Achieving quality in open-source software," *IEEE Software*, vol. 24, no. 1, pp. 58–64, 2007.

[12] P. Wayner, *Free for All*, HarperCollins, New York, NY, USA, 2000.

[13] K. Crowston and B. Scozzi, "Bug fixing practices within free/libre open source software development teams," *Journal of Database Management*, vol. 19, no. 2, pp. 1–30, 2008.

[14] D. Cubranic and K. Booth, "Coordinating open-source software development," in *Proceedings of the 8th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 61–69, 1999.

[15] A. Mockus, R. T. Fielding, and J. Herbsleb, "A case study of open source software development: the Apache server," in *Proceedings of the 22nd International Conference on Software Engineering*, pp. 263–272, June 2000.

[16] P. A. David, A. Waterman, and S. Arora, "FLOSS-US, the Free/Libre/Open Source Software Survey for 2003," http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf FLOSS-US-Report.pdf.

[17] J. Lerner and J. Tirole, "Some simple economics of open source," *Journal of Industrial Economics*, vol. 50, no. 2, pp. 197–234, 2002.

[18] A. Mockus, R. T. Fielding, and J. D. Herbsleb, "Two case studies of open source software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology*, vol. 11, no. 3, pp. 309–346, 2002.

[19] K. R. Lakhani and E. Von Hippel, "How open source software works: "free" user-to-user assistance," *Research Policy*, vol. 32, no. 6, pp. 923–943, 2003.

[20] J. West, "How open is open enough? Melding proprietary and open source platform strategies," *Research Policy*, vol. 32, no. 7, pp. 1259–1285, 2003.

[21] J. Feller and B. Fitzgerald, "A framework analysis of the open source software development paradigm," in *Proceedings of the 21st Annual International Conference on Information Systems*, pp. 58–69, Brisbane, Australia, 2000.

[22] H. Koppelman and B. Van Dijk, "Creating a realistic context for team projects in HCI," *SIGCSE Bulletin*, vol. 38, no. 3, pp. 58–62, 2006.

[23] E. Golden, B. E. John, and L. Bass, "The value of a usability-supporting architectural pattern in software architecture design: a controlled experiment," in *Proceedings of the 27th International Conference on Software Engineering (ICSE '05)*, pp. 460–469, St. Louis, Mo, USA, May 2005.

[24] D. M. Nichols and M. B. Twidale, "Usability processes in open source projects," *Software Process Improvement and Practice*, vol. 11, no. 2, pp. 149–162, 2006.

[25] G. Çetin and M. Göktürk, "A measurement based framework for assessment of usability-centricness of open source software projects," in *Proceedings of the 4th International Conference on Signal Image Technology and Internet Based Systems (SITIS '08)*, pp. 585–592, December 2008.

[26] L. Zhao and F. P. Deek, "Exploratory inspection: a learning model for improving open source software usability," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '06)*, 2006.

[27] Z. Hussain, W. Slany, and A. Holzinger, "Current state of agile user-centered design: a survey, HCI and usability for E-inclusion," in *Proceedings of the 5th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society (USAB '09)*, vol. 5889 of *Lecture Notes in Computer Science*, pp. 416–427, Springer, Berlin, Germany, 2009.

[28] S. Pemberton, "Scratching someone else's itch: (why open source can't do usability)," *Interactions*, vol. 11, no. 1, p. 72, 2004.

[29] Y. Yunwen and K. Kishida, "Toward an understanding of the motivation of open source software developers," in *Proceedings of the 25th International Conference on Software Engineering*, pp. 419–429, May 2003.

[30] A. Holzinger, C. Stickel, M. Fassold, and M. Ebner, "Seeing the system through the end users' eyes: shadow expert technique for evaluating the consistency of a learning management system, HCI and usability for E-inclusion," in *Proceedings of the 5th Symposium of of the Austrian Computer Society (USAB '09)*, vol. 5889 of *Lecture Notes in Computer Science*, pp. 178–192, Springer, Berlin, Germany, 2009.

[31] X. Faulkner and F. Culwin, "Integrating HCI and SE," *ACM SIGCSE Bulletin*, vol. 32, no. 3, pp. 61–64, 2000.

[32] M. B. Rosson, J. M. Carroll, and C. M. Rodi, "Case studies for teaching usability engineering," *ACM SIGCSE Bulletin*, vol. 36, no. 1, pp. 36–40, 2004.

[33] N. Markov, "An introduction to the UCD methodology in the current environment," CASCON Workshop Report, 2003.

[34] A. Holzinger, P. Sammer, and R. Hofmann-Wellenhof, "Mobile computing in medicine: designing mobile questionnaires for elderly and partially sighted people," in *Proceedings of the 10th International Conference on Computers Helping People with Special Needs (ICCHP '06)*, vol. 4061 of *Lecture Notes in Computer Science*, pp. 732–739, Springer, Berlin, Germany, 2006.

[35] L. J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychometrika*, vol. 16, no. 3, pp. 297–334, 1951.

[36] A. H. van de Ven and D. L. Ferry, *Measuring and Assessing Organizations*, John Wiley & Sons, New York, NY, USA, 1980.

[37] A. Osterhof, *Classroom Applications of Educational Measurement*, Prentice-Hall, Upper Saddle River, NJ, USA, 2001.

[38] D. T. Campbell and D. W. Fiske, "Convergent and discriminant validation by the multitrait-multimethod matrix," *Psychological Bulletin*, vol. 56, no. 2, pp. 81–105, 1959.

[39] A. L. Comrey and H. B. Lee, *A First Course in Factor Analysis*, Psychology Press, Hillsdale, NJ, USA, 2nd edition, 1992.

[40] H. F. Kaiser, "A second generation little jiffy," *Psychometrika*, vol. 35, no. 4, pp. 401–417, 1970.

[41] H. F. Kaiser, "The application of electronic computers to factor analysis," *Educational and Psychological Measurement*, vol. 20, pp. 141–151, 1960.

[42] J. Stevens, *Applied Multivariate Statistics for the Social Sciences*, L. Erlbaum Associates, Hillsdale, NJ, USA, 1986.

[43] C. Fornell and F. L. Bookstein, "Two structural equation models: LISREL and PLS applied to consumer exit-voice

theory," *Journal of Marketing Research*, vol. 19, pp. 440–452, 1982.

[44] K. Joreskog and H. Wold, *Systems under Indirect Observation: Causality, Structure and Prediction*, Elsevier, North Holland, The Netherlands, 1982.

[45] http://en.wikipedia.org/wiki/P-value.

[46] A. Holzinger, "Usability engineering methods for software developers," *Communications of the ACM*, vol. 48, no. 1, pp. 71–74, 2005.

[47] J. Singer and N. G. Vinson, "Ethical issues in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 1171–1180, 2002.

[48] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering*, Kluwer Academic Publishers, Norwell, Mass, USA, 2000.

[49] R. R. Faden, T. L. Beauchamp, and N. M. P. King, *A History and Theory of Informed Consent*, Oxford University Press, Oxford, UK, 1986.

[50] J. Katz, *Experimentation with Human Beings*, Russell Sage Foundation, New York, NY, USA, 1972.

[51] http://www.upassoc.org/usability_resources/about_usability/what_is_ucd.html.