How Do Open Source Software (OSS) Developers Practice and Perceive Requirements Engineering? An Empirical Study

Jaison Kuriakose
Faculty of Business Administration
Memorial University of Newfoundland
St. John's, Canada

Jeffrey Parsons
Faculty of Business Administration
Memorial University of Newfoundland
St. John's, Canada

***Abstract*—In open source software (OSS) development domain (a largely volunteer driven, geographically distributed, web based form of software development), it is mainly the OSS developers who are responsible for overseeing and managing the developmental activities. Existing OSS literature, based on qualitative analysis of web-based artifacts (e.g. data on discussion forums, issue databases) of a few OSS projects, report that requirements generation in OSS development is largely informal and ad hoc. But there is lack of an empirical study involving the practitioners themselves i.e. the OSS developers. We conducted a web-based survey among OSS developers in order to gain insights in to how they actually practice requirements engineering activities and what are their perceptions about it. For 57 requirements engineering practices obtained from closed source software development (CSSD) literature, the respondents indicated whether they currently used those practices in their OSS projects and whether those practices were useful for OSS development. The analysis of survey responses revealed that OSS developers used requirements engineering practices (from CSSD) significantly less in their developmental activities than what they believed they should have, indicated through usefulness ratings. We also asked participating OSS developers to indicate their perceptions about the usage of five informal requirements generation activities reported in OSS literature (e.g. developers simply asserting the requirements instead of eliciting). Subsequent analysis revealed that OSS developers used informal requirements generation activities significantly more than requirements elicitation practices (from CSSD) in their developmental activities. We use the survey findings to discuss implications for practice and research.**

***Index Terms*— requirements engineering practices from closed source software development, open source software development, current use, perceived usefulness.**

I. Introduction

The term ***open source software (OSS)*** refers to software whose source code is available to be freely used, modified or redistributed (e.g. Mozilla Firefox) [1, 2]. This paper focuses on the development processes used for OSS. OSS development is often carried out in an ad hoc manner [6], as OSS projects typically do not have any formal organizational structure [1]. OSS development often lacks the processes followed in conventional industrial style software development [3], but is marked instead by rapid development, frequent incremental releases, and massive parallel development and debugging [2].

There are important differences between OSS development and conventional closed source software development [3]. OSS development is often carried out by geographically distributed developers and other contributors, who voluntarily participate in the development, driven by different kinds of motivation including need for certain functionalities, learning opportunities, career development and fun [1, 2, 5]. These volunteering contributors, often working from arbitrary locations, collaborate almost entirely over the internet using tools such as communication artifacts (e.g. discussion forums), software code tracking tools and issue trackers [4, 1]. Non-developer users also contribute by providing bug reports and feature requests [1]. OSS contributors often choose to do work that is of interest to them instead of work assigned to them [3, 4]. Most contributors leave after contributing for a certain period of time and only a small group of developers remain to oversee the further evolution of the project [5]. OSS projects often have no explicit system level design, project plans, schedules or list of deliverables [4]. There are many web based OSS development environments such as Sourceforge, GitHub, Google Code and Codeplex within which OSS developmental activities often take place.

Crowston et al [1], in their review of the empirical work done on OSS development, highlight that much research is needed on the use of closed source software development practices (e.g. requirements engineering) in OSS development. Alspaugh and Scacchi [7] mention that the answer to the question of whether OSS domain would benefit from requirements engineering practices (from CSSD) is not clear and awaits further research [7]. In OSS development domain, it is mainly the OSS developers who are responsible for overseeing and managing the developmental activities [25]. Hence we expect that, by gathering information from practicing OSS developers about the current usage of RE practices in their developmental activities and their perceptions about whether requirements engineering activities (from CSSD) are beneficial for OSS development could provide valuable insights, in addition to what we already know. The research questions that we attempted to answer through the research reported in this paper are:

1. What are OSS developers' perceptions of the extent to which requirements engineering activities (from CSSD) are present in their developmental activities?
2. What are OSS developers' perceptions of the extent to which requirements engineering activities (from CSSD) are beneficial for OSS development?

In order to obtain data on OSS developers' perceptions, we conducted a web based survey among OSS developers registered on OSS development environments and OSS projects' websites. Our research setting and methodology is described in detail section three. In the next section (section two), we provide an overview of some of the findings reported in OSS literature about RE activities in OSS development.

II. An Overview of Findings Reported in Existing Literature on Requirements Engineering Practices in OSS Development

Crowston et al, in their review of OSS literature, mention that OSS projects often lack formal requirements engineering practices [1]. For example, OSS projects often do not have formal requirements document; instead, requirements may exist only informally as part of the communication messages posted on discussion forums and similar artifacts. OSS users can submit requirements data using artifacts such as bug reports and feature requests.

TABLE 1.Key themes from existing literature on RE in OSS

Requirements engineering activities reported in OSS literature	Scacchi (2002) [8]	Noll (2008) [9]	Ernst and Murphy (2012) [10]	Llanos and Castillo (2012) [11]	Massey (2002) [12]	Noll and Liu (2010) [13]
Requirements are asserted by developers	√	√	√	√	√	
Requirements exist informally as part of communication messages (e.g., emails, discussion forums)	√			√		√
Lack of formal requirements elicitation			√	√		
Lack of formal requirements validation	√			√		
Lack of formal requirements prioritization			√			
Requirements are contributed by users through bug reports and feature requests		√		√	√	
A source of requirements is features appearing in other software		√		√	√	

Table 1 summarizes some key themes in the existing literature on RE in OSS development. Overall, requirements engineering appears to be informal and ad hoc in OSS development. The findings reported in most studies listed in Table 1 are based on qualitative analysis of project artifacts (e.g., data on discussion forums, issue databases) of a limited number of OSS projects hosted within OSS development environments such as Sourceforge. For example, the findings of Noll [9] are based on analysis of archival data in discussion forums, web logs and issue databases of Firefox. There is lack of an empirical study directly involving practicing OSS developers, providing perceptual data on how the OSS developers do (if they do) and perceive these practices. We believe that such an empirical study would be a useful addition to and complement the existing qualitative literature on RE in OSS.

The next section describes the research setting and methodology in detail.

III. Research Setting and Methodology

A web-based survey was used to gather data from OSS developers. Surveygizmo, a web based survey tool, was used to carry out the survey. The complete survey questionnaire is available from the first author on request. The survey consisted of four sections.

The first section contained questions about which RE practices were used during the development of OSS projects. That the OSS developers had worked on. The list of RE practices used in closed source software development was obtained from Sommerville and Sawyer [14] (these are detailed in Tables 3 to 8). The list of RE practices provided by Sommerville and Sawyer is fairly comprehensive and there is empirical evidence for the use of these practices in industries (e.g. [15]). The RE practices have been categorized by Sommerville and Swayer in to requirements documentation practices (table 3), requirements elicitation practices (table 4), requirements analysis and negotiation practices (table 5), requirements modeling practices (table 6), requirements validation practices (table 7) and requirements management practices (table 8). Informal requirements generation activities reported in OSS literature were also included: (1) assertion of requirements by OSS developers based on personal experience; (2) assertion of requirements by OSS developers based on personal knowledge of user needs; (3) requirements information submitted by OSS users in the form of bug reports; (4) requirements information submitted by OSS users in the form of feature requests; and (5) requirements derived from features found in other software [9]. Participants were asked to indicate their level of usage of each requirements engineering practice within open source software development by selecting a single option on a scale from 1 (never used) to 5 (always used) (participants also had the option to select 'not applicable' or 'I do not know'; coded as 0). There were very few responses in 'not applicable' and 'I do not know' categories (generally less than 5% of total respondents for each question).

The second section of the survey asked participants whether they perceived the RE practices as beneficial for OSS development. The practices listed were the same as those in the first section, excluding the five informal practices reported in OSS literature. Participants were asked to rate the usefulness of each requirements engineering practice by selecting a single option on a scale from 1 (harmful) to 5 (extremely useful) (participants also had the option to select 'not applicable' or 'I

do not know'; coded as 0). There were very few responses of 'not applicable' and 'I do not know' (less than 5% of total respondents for each question).

Sections three asked OSS developers to indicate their perceptions about problems and challenges that could occur during requirements generation in OSS development. The list of potential problems and challenges was obtained from Damian and Zowghi [28] and Schmid [29]. The context investigated in [28], while being CSSD, was a geographically distributed context. Hence we expect that problems and challenges identified in [28] could be relevant to the context of requirements discovery in OSS development. Participants were asked to check all potential problems or challenges that they experienced or expected to experience during requirements discovery in OSS development. We briefly discuss about these problems and challenges as potential factors that could contribute to some of the findings from section one and two, that could be investigated empirically in future research, for example through experimental manipulation. Section four had demographic questions.

Email invitations containing the link to the survey were send to OSS developers registered on OSS development environments such as GitHub and Sourceforge. In addition, the link to the survey was posted on the webpages, discussion forums and mailing lists of many OSS projects such as Mozilla and Apache. No incentives were provided for completion of the survey. The findings reported below are based on analysis of survey data obtained from 84 respondents. About 65% of the respondents were from North America while remaining came from South America, Europe, Asia and rest of the world. 54.7% of the respondents had worked on less than ten OSS projects while remaining had worked on more than ten projects. 54.1% of the respondents on average, spent more than 10 hours working on OSS projects (remaining <ten hours).

IV. Requirements Generation Practices of Oss Developers: A Quantitative Analysis

A paired samples t-test was run to compare usage scores of the informal requirements generation practices reported in the OSS literature and the formal requirements elicitation practices (from CSSD) described in [14]. A sample of results is shown in Table 2. We ran the paired samples test for all pairs of informal/formal practices. For most of the pairs, the informal practice had significantly higher usage in comparison to the formal elicitation practice.

Table 2 show that survey respondents reported that informal requirements generation activities discussed in the OSS literature had significantly higher usage in OSS development than formal requirements elicitation practices (from CSSD). For example, requirements were significantly more likely to be simply asserted by OSS developers rather than being generated through formal practices such as identifying relevant users and having requirements elicitation sessions with them. The results in Table 2 provide quantitative evidence supporting the qualitative claim in the OSS literature that requirements generation in OSS development is largely informal and ad hoc.

TABLE 2.Formal Vs. Informal Requirements generation practices

Informal practice/formal practice	Mean (/5) Inf. / Form.	Std. Dev. Inf. /Form.	t	p
Requirements asserted by OSS developers based on personal experience / Identify and consult software users	4.09 / 2.85	1.031/ 1.314	8.609	.000
Requirements asserted based on his or her personal experience / Use business concerns to drive requirements elicitation	4.09 / 2.35	1.031/ 1.320	9.735	.000
Requirements contributed by users through feature requests / Identify and consult software users	3.65 / 2.85	1.043/ 1.314	4.785	.000
Requirements contributed by users through feature requests / Record requirements rationale	3.65 / 2.55	1.043/ 1.330	7.279	.000
Requirements derived from features found in some other software/ Identify and consult software users	3.36 / 2.85	1.143/ 1.314	3.323	.001
Requirements derived from features found in some other software/ Collect requirements from multiple viewpoints	3.36 / 2.61	1.143/ 1.232	4.894	.000

The next section describes survey findings about the perceptions of OSS developers about whether the requirements engineering practices are beneficial for OSS development and whether their perceptions match their actual practice.

V. Findings on Oss Developers' Practices and Perceptions of Requirements Engineering Activities in Oss Development

Paired samples t test was run between usage ratings and usefulness ratings for each practice. Similar analysis methods for ratings data have been used in past research from other domains (e.g. [26]; [27]). In general, the reported usage of RE practices in developmental activities carried out by OSS developers did not match their perceptions about the extent to which these practices could be advantageous for OSS development. Thus when it comes to requirements related activities in open source software development, OSS developers do not appear to be actually practicing what they believe could be beneficial. There is empirical evidence that usefulness perceptions of programmers drive their usage intentions (e.g. [30]). Thus OSS developers do not appear to incorporate requirements engineering activities (from CSSD) in OSS developmental activities to the extent they believe they should be doing. The significant differences between practice and perceptions of OSS developers are observed in all major categories of requirements engineering activities. These findings are reported in detail in tables 3-8 below. We want to mention here that we also applied Holm-Bonferroni correction for multiple comparisons and

found that most of the significant results were indeed significant, even with the adjusted alphas. For all categories of RE practices, the total scores on use scale was significantly lower than total scores on usefulness scale (please see last row of tables 3-8; values shown are mean total score).

Table 3 shows that the reported usage of requirements documentation practices by OSS developers do not match their perceptions about the extent to which these practices would be beneficial, if incorporated in OSS development.

TABLE 3.OSS developers' perceptions of current practice and usefulness of Requirements documentation in OSS development

Requirements documentation practices [14]	Current use*	Perceived Usefulness*	SD±	t±±/(p value)
Define a standard document structure	2.34	3.18	1.110	6.924/(.000)
Explain how to use the document	2.17	3.11	1.391	6.154/(.000)
Include a summary of the requirements	2.78	3.5	1.558	4.183/(.000)
Make a business case for the software	1.99	2.7	1.3	4.925/(.000)
Define specialized terms	2.63	3.33	1.462	4.333/(.000)
Make document layout readable	3.09	3.77	1.498	4.128/(.000)
Help readers find information	2.85	3.84	1.544	5.795/(.000)
Make the document easy to change	3.27	3.79	1.517	3.130/(.002)
TOTAL	21.5385	27.6282	7.82468	6.874/(.000)

- **Mean values**

± Standard deviation of the difference between current use and perceived usefulness

±± t is ratio of difference (Current use – Usefulness) to Standard error

In OSS domain, there may be some informal documentation, such as project read me files having some screenshots about the project and some instructions or requirements information may exist as part of some messages or discussions posted on forums in OSS development environments ([16], [8]). This is also evident from the comment of one survey respondent: "*I have never been involved in an open source project that had a formal requirements document. For these small scale projects, requirements are intrinsically linked to purpose and features which are listed on the project website (if available), or more commonly, the README*". A comment from another developer is "*many open source projects that I have worked on have a read me. Occasionally a wiki with ad-hoc other documentation*". These informal ad hoc ways of documenting that is prevalent in OSS development could potentially explain some of the observed differences. For example, such ad hoc documentation would not have a standard structure; any one could post anything on the discussion forums which may or may not contain requirements information. A potential research direction could be a longitudinal research (preferably within some web based OSS development environment such as Source forge) that compares outcomes from the evolution of three versions of the same open source software project, one with a formal rigorously specified requirements document with a standard structure, one with an ad hoc informal document such as a read me and a control condition with no requirements information, everything else being same. The treatment condition could be gradually refined to incorporate other practices perceived as beneficial such as providing requirements summary.

The requirements documentation practice rated highest on usefulness scale by OSS developers was "help readers find information" (Mean = 3.84). This could imply that OSS developers perceive information search facilities, especially for requirements information such as feature requests as highly beneficial. In fact, Cleland-Huang et al. found that users often created unnecessary new threads for posting feature requests when in fact, very similar information already existed on the project forums [17]. They advocate the provision of better information search facilities within project forums that can deliver more accurate search results [17].

TABLE 4.OSS developers' perceptions of current practices and usefulness of Requirements elicitation in OSS development

Requirements elicitation practices [14]	Current use*	Perceived Usefulness*	SD±	t±±/(p value)
Assess software feasibility	2.7	3.19	1.549	2.905/(.005)
Be sensitive to political and organizational considerations	2.23	2.73	1.356	3.400/(.001)
Identify and consult software users	2.84	3.71	1.322	5.932/(.000)
Record requirement sources	2.48	3.10	1.378	4.063/(.000)
Define the software's operating environment	3.62	3.5	1.598	-.691/(.491)
Use business concerns to drive requirement elicitation	2.35	2.62	1.287	1.888/(.063)
Look for domain constraints	2.7	3.18	1.451	3.025/(.003)
Record requirements rationale	2.57	3.39	1.371	5.397/(.000)
Collect requirements from multiple view points	2.61	3.45	1.232	6.185/(.000)
Prototype poorly understood requirements	2.83	3.51	1.359	4.498/(.000)
Use scenarios to elicit requirements	2.66	3.13	1.493	2.868/(.005)
Define operational processes	2.35	3.04	1.357	4.586/(.000)
Reuse requirements	2.6	3.1	1.415	3.140/(.002)
TOTAL	34.1067	40.8800	11.66257	5.030/(.000)

Table 4 above shows that for most of the requirements elicitation practices, the reported usage by OSS developers do not

match their perceptions about the extent to which the elicitation practices would be advantageous to OSS development.

A potential contributor to the observed differences between practice and perceptions of requirements elicitation could be the widespreadness of informality in OSS development as discussed in section four. This is also illustrated from the following comments: "*I had to think hard about some of these. These practices we do, we certainly don't think of in the terms you used. Our development is mostly informal, but sometimes use some of those if they sound like a good idea, rather than as a conscious decision to do so*". A comment from another OSS developer: "*Again these questions assume a high degree of formality around requirements elicitation. This has not been my experience*". Other potential factors that could contribute to the observed differences in table 4 emerged from problems and challenges section of survey, something that is worth further investigating. For example, in problems and challenges section of the survey, 48% of the respondents indicated that geographical distance and time difference between countries could make it challenging for the members to interact for requirements generation purposes, a potential challenge in *identifying and consulting users*. In [18] it is reported that a establishing a two-way communication between OSS project administrators and users is a major challenge. They further report that OSS developers were often not able to understand the real needs of OSS users and users were often dissatisfied because their requirements information was ignored by OSS developers [18]. Interestingly, users tried to resort to tricks such as specifically mentioning the word feature request in the issue information that they submitted to get the attention of developers [18]. In our survey, 59% of respondents indicated differences in educational background as a barrier, 56% indicated language differences as a barrier and 35% indicated diminished understanding of the working context of other members as a barrier for requirements generation in OSS development. 65% indicated differences in corporate culture of members as a barrier and 25% indicated differences in national culture as a barrier for requirements generation in OSS development. It may be difficult for OSS contributors from different educational, corporate or national backgrounds to understand each other, a potential challenge to *collecting requirements from multiple viewpoints*.

The requirements elicitation practice rated highest on usefulness scale by OSS developers was "identify and consult software users" (Mean = 3.71). The actual benefits of this practice for OSS development could be analyzed by using experimental manipulation. In CSSD, analysts commonly use questionnaires (e.g. during interview sections) to elicit requirements information from stakeholders [18]. An example is context independent prompts proposed by Browne and Rogich [19]. A two group experiment could be designed with treatment being presence/absence of requirements elicitation questionnaire (such as the context independent prompts) and the outcomes from both conditions could be compared. A more realistic investigation could be embedding such questionnaires within issue repository of a real OSS project and to analyze the outcomes after a certain period of time. Similar experimental designs or field studies could be used to investigate the potential outcomes from usage of other practices such as scenario based elicitation, prototype based elicitation or requirements reuse in OSS development. The practice rated lowest on usefulness scale was "using business concern to drive requirements elicitation" (mean = 2.62) with non-significant difference between practice and perceptions. This could be because of the largely non-commercial nature of OSS development with no direct controlling by business firms (c.f. [1]). OSS developers may prefer keeping it that way as evident from the following comment: "*This question barely makes sense in the context of personal open source. The investment is in a programmer's attention. The organizational value can just be programmer pleasure. If the programmer(s) are experienced, the feasibility of a project is often known a-priori, since it is not meant to serve institutional needs. For most small open source projects, the software is developed first, and if it happens to serve institutional needs it becomes more widely adopted and supported*".

Table 5 below shows that the reported usage of requirements analysis and negotiation practices by OSS developers does not match their perceptions about the extent to which these practices would be beneficial, if incorporated in OSS development.

TABLE 5.OSS developers' perceptions of current practice and usefulness of Requirements analysis & negotiation in OSS development

Requirements analysis and negotiation practices [14]	Current use*	Perceived Usefulness*	SD±	t±±/(p value)
Define software boundaries	3.11	3.48	1.651	2.061/(.042)
Use checklists for requirements analysis	2.36	2.99	1.197	4.770/(.000)
Provide software to support negotiations	2.47	2.78	1.615	1.767/(.081)
Plan for conflict and conflict resolution	2.11	3.01	1.302	6.279/(.000)
Prioritize requirements	3.57	3.85	1.363	1.863/(.066)
Classify requirements using a multidimensional approach	2.11	2.67	1.399	3.689/(.000)
Use interaction matrices to find conflicts and overlaps	1.8	2.36	1.150	4.487/(.000)
Assess requirements risk	2.34	2.9	1.222	4.223/(.000)
TOTAL	19.6341	23.7927	7.34422	5.127/(.000)

The observed differences in table 5 could be representing the general informality of requirements analysis reported in OSS literature. For example, Noll and Liu [13] report that negotiations may happen only informally through brief discussions between developers (through postings on forums) to agree upon some requirement (such as some proposed feature)

[13]. This is also evident from the comments of a respondent: "*it's a small project…our process for conflict avoidance is 'ask that guy or that guy, depending on domain, or both' on irc*" irc here referring to internet relay chat. In another comment, the OSS developer explicitly agrees that usage of the analysis and negotiation practices are not prevalent in OSS development "*a lot of these are pretty heavyweight processes that I would associate with large-slow moving software companies than with free software projects*". There is some evidence for OSS developers ignoring users' suggestions and comments in the decision making about requirements and decisions being made based on developers' interests [18, 10]. A comment from a responding OSS developer in our survey seems to support this: "*As an open source developer, I rarely negotiate on my own projects with others as they are built by my-self only. I do not need to negotiate or compromise most of the time*". Such perceptions at times may also contribute towards low usage of formal requirements analysis and negotiation practices in OSS development. Additional contributing factors emerge from the responses from the problems and challenges section of our survey. 35% indicated that it is difficult to manage conflicts and have open discussions about interests which indicate that requirements conflict planning and management is a challenge in general in OSS domain. So is negotiating and prioritizing requirements as indicated by 32% of respondents. These reported perceptions of OSS developers indicate that many OSS developers consider these activities to be challenging to implement in OSS domain in spite of perceiving them as beneficial. This highlights new venues of research, especially investigating whether and how feature support could be provided within OSS development environments for carrying out these activities. 40% of respondents indicated that "*the use of diverse terminologies and diverse levels of details by members in expressing requirements*" makes it difficult to analyze requirements in order to discover potential conflicts and redundancies. 36% respondents indicated that it was difficult to arrive at a common understanding about requirements. Other potential factors that emerged were difficulty in achieving cohesion/forming coalition with others (34%) and difficulty in trusting others (29%).

An interesting observation is the non-significant difference in practice and perceptions of "prioritizing requirements". In [18], it is reported that OSS projects may have some prioritization mechanisms such as voting button and ability to specify priority while submitting some feature request information [18], indicating that there is some tool support available in OSS development environments for requirements prioritization, a potential facilitator for usage of requirements prioritization. This could be a reason for the non-significant observed difference. Interestingly, "prioritize requirements" was also the practice that was rated highest on usefulness scale by responding OSS developers (mean = 3.85), indicating that they perceive it as highly beneficial for OSS development. Laurent and Cleland-Huang et al mention that the existing prioritization techniques have their own problems and a lack of sophisticated support for requirements prioritization [18]. Hence future research could empirically investigate new innovative mechanisms of prioritization. For example, one direction could be investigating whether collective intelligence of crowd could be utilized to arrive at more efficient prioritization decisions; aggregating individual prioritization scores to form a collective prioritization score and empirically examining how effective it is.

Table 6 below shows that the reported usage of requirements modeling practices by OSS developers does not match their perceptions about the extent to which the incorporation of these practices would be advantageous to OSS development.

TABLE 6.OSS developers' perceptions of current practice and usefulness of Requirements modeling for OSS development

Requirements modeling practices [14]	Current use*	Perceived Usefulness*	SD±	t±±/(p value)
Develop complementary models of the proposed software	2.13	2.74	1.245	4.436/(.000)
Model the software's environment	2.57	3.10	1.045	4.544/(.000)
Model the software's architecture	2.93	3.32	1.069	3.327/(.001)
Use structured methods for software modeling	2.26	2.7	1.380	2.881/(.005)
Use a data dictionary	2.1	2.61	1.381	3.358/(.001)
Document the links between user requirements and models	1.87	2.76	1.314	6.176/(.000)
TOTAL	13.7439	17.0976	5.28537	5.746/(.000)

Table 6 points towards low usage of requirements modeling practices in OSS domain which is also in line with the findings of Badreddin et al. [20], who analyzed twenty active OSS projects such as Chromium and GNOME and found no evidence of any modeling practice in the analyzed projects [20]. This could be perhaps an outcome of the informality observed in earlier stages, i.e. the general lacking of a well specified requirements document outputted through formal elicitation, analysis and documentation. Models capture knowledge about the domain of the software [23]. Human working memory can hold only a small number of information items [21] and external memory such as models can help overcome the limitations of human working memory [c.f. [21]). When OSS developers develop OSS software for which they themselves are potential users, they can be expected to have good domain knowledge but when potential user base expands to include users other than the developers, the OSS developers may be lacking relevant domain knowledge [24]. OSS developers without realizing this may simply assert requirements and proceed to coding (c.f. [8]) which could be another reason for low usage of modeling in spite of being perceived as beneficial. This is also evident from the following comments of responding OSS developers: "*The engine guy knew what he was doing and did it*". Another comment: "*if the project is in the head of developer, it is a waste of time to model the project out using UML*". From the comments, OSS developers appear to be confident that they

have a thorough knowledge of the domain which might be a faulty perception.

The requirements modeling practice that was rated highest on the usefulness scale by the respondents was "*Model the software's architecture*" (mean = 3.32). This could be because of the in general, code centric nature of OSS development. Future research could analyze in field settings, the outcomes from providing feature support (e.g. ability to construct models using UML) for architecture modeling. For example, whether the source code quality of an experimental OSS project for which an architecture model was created using UML significantly better than a control OSS project that did not had any model constructed.

Table 7 tells same story for requirements validation as other practices.

TABLE 7.OSS developers' perceptions of current practice and usefulness of Requirements validation for OSS development

Requirements validation practices [14]	Current use*	Perceived Usefulness*	SD±	t±±/(p value)
Check that the requirements document meets your standards	2.23	2.94	1.224	5.206/(.000)
Organize formal requirements inspection	1.96	2.54	1.376	3.738/(.000)
Use multidisciplinary teams to review requirements	2.03	2.97	1.348	6.258/(.000)
Define validation checklists	1.89	2.77	1.187	6.633/(.000)
Use prototyping to animate requirements	2.51	3.08	1.117	4.488/(.000)
Write a draft user manual	2.64	3.38	1.465	4.504/(.000)
Propose requirements test cases	2.99	3.73	1.276	5.203/(.000)
Paraphrase models	1.89	2.78	1.302	6.095/(.000)
TOTAL	18.2533	24.0133	6.23382	8.002/(.000)

Overall, usage of requirements validation practices appear to be low in OSS domain. This is in line with what Noll and Liu [13] report, who in their case study of an OSS project, found instances when no validation happened; a requirement message would be posted by someone and after sometime, an implementation of it emerged. The comments from survey respondents point towards some potential reasons, for example, small OSS projects not being able to afford domain experts to conduct validation: "*the standards are ad-hoc and non-explicit in most cases. Most open source projects I worked on have been too small to bother (or be able to afford) domain experts and what not or even have sizeable teams*".

The requirements validation practice that was rated highest on the usefulness scale by the respondents was "propose requirements test cases" (mean = 3.73). This may be because OSS developers could easily write pieces of code to test out some requirement. A comment from an OSS developer appears to illustrate this perception: "*We unit test the living beans out of our software, if that counts*". Future research could look at the benefits of providing tools for writing test cases (e.g. templates) for requirements during OSS development.

Table 8 shows that the reported usage of requirements management practices by OSS developers do not match their perceptions about the extent to which the incorporation of requirements management practices would be advantageous to OSS development. Requirements management appear to be informal as well as evident from the following comments of responding OSS developers: "*Much of what is done, is done in a non-formal manner*" and another comment: "*no formal requirements management has been done for this very small project*". Respondents pointed towards issue repositories as potential databases: "*we are using the Google code issue tracker to keep track of requirements*". Requirements information may get generated in issue repositories and such artifacts during the life of an OSS projects and lie there but the subsequent management of these requirements appears to be ad hoc as illustrated from the comments above. This is also in line with findings by Laurent and Cleland-Huang [18]. They found cases where the requirements submission and management process was inefficient (e.g. requirements information submitted by users getting ignored) and consequently, users were dissatisfied. They also report that often there was no separate feature request submission mechanism [18]. Such poor requirements management can have undesirable consequences. For example, Herzig et al [22] report that often feature requests and documentation requests are misclassified as bug reports.

TABLE 8.OSS developers' perceptions of current practice and usefulness of Requirements management for OSS development

Requirements management practices [14]	Current use*	Perceived Usefulness*	SD±	t±±/(p value)
Uniquely identify each requirement	2.6	3.48	1.338	5.989/(.000)
Define policies for requirements management	2.05	2.85	1.319	5.527/(.000)
Define traceability policies	1.83	2.56	1.207	5.487/(.000)
Maintain a traceability manual	1.57	2.32	1.040	6.477/(.000)
Use a database to manage requirements	2.28	3.18	1.122	7.337/(.000)
Define change management policies	2.13	3.01	1.373	5.789/(.000)
Identify global software requirements	2.52	3.23	1.110	5.834/(.000)
Identify volatile requirements	2.27	3.06	1.079	6.712/(.000)
Record rejected requirements	2.43	3.30	1.332	5.932/(.000)
TOTAL	19.6375	27.0250	6.72704	9.822/(.000)

The unwanted consequences of ineffective requirements management such as user dissatisfaction points towards the need to have efficient requirements management mechanisms

in place which OSS developers also seem to have an understanding about as illustrated through significantly higher usefulness ratings.

The requirements management practice that was rated highest on the usefulness scale by the respondents was "uniquely identify each requirement" (mean = 3.48). This seems to be already present in external issue trackers such as JIRA as evident from following comment: "*the database is typically an issue manager, e.g. GitHub issue tracker, Bugzilla or JIRA which automatically includes an identifier*". This could be replicated in web based OSS development environments. Comments by responding OSS developers also highlight possible ways of incorporating some of the requirements management practices perceived as beneficial. For example, traceability could be incorporated in issue repositories by having feature support for storing dependencies between requirements. This is illustrated through the following comment: "*Traceability is trivial assuming the database used to store the requirements permits dependency links (hence no need for a traceability manual)*".

VI. Conclusion and Future Work

This study reported findings about OSS developers' current practice and perceptions about requirements engineering practices (from CSSD). We found evidence that the current practice of requirements engineering practices by OSS developers do not match their perceptions about it. Most of the practices were perceived as beneficial for OSS development in spite of significantly low reported usage. We are currently investigating the actual benefits of some of the RE practices for requirements discovery in OSS development, using more rigorous methods such as experiment.

References

[1] K. Crowston, K. Wei, J. Howison, and A. Wiggins. Free/Libre open source software development: what we know and what we do not know. *ACM Computing Surveys*. 44(2): 1-35, 2012.

[2] J. Feller, and B. Fitzgerald. A framework analysis of the open source development paradigm. In *Proc. of the 21st International Conference on Information Systems*. 2000

[3] 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*. 11(3): 309-346. 2002

[4] F. Detienne, J. M. Burkhardt, and F. Barcellini. Open source software communities: current issues. *PPIG Newsletter*, 1-7. 2006

[5] S. K. Shah. (2006). Motivation, governance and the viability of hybrid forms in open source software development. *Management Science*. 52(7): 1000-1014. 2006

[6] L. Zhao, and S. Elbaum. Software quality assurance under the open source model. *Journal of Systems and Software*. 66(1): 65-75. 2003

[7] T. A. Alspaugh and W. Scacchi. Ongoing software development without classical requirements. In *Proc. of IEEE RE*, 165-174. 2013

[8] W. Scacchi. Understanding the requirements for developing open source software systems. *IEEE Software*, 149(1): 24–39. 2002

[9] J. Noll. Requirements acquisition in open source development: Firefox2.0. In *Open Source Development, Communities and Quality*. B Russo, E. Damani, S. Hissam, B. Lundell, and G. Succi, Eds. Springer, Boston, 275: 69-79. 2008

[10] N, Ernst. And G, Murphy. Case studies in just-in-time requirements analysis. In *Proc. of IEEE Second International Workshop on Empirical Requirements Engineering*. 2012

[11] J. W. C. Llanos, and S. T. A. Castillo. Differences between traditional and open source development activities. In *Proc. of 13th International conference on Product-Focused Software Process Improvement,* 131-144. 2012

[12] B. Massey. (2002). Where do open source requirements come from (And what we should do about it). In *Proc. of Second ICSE Workshop on Open Source Software Engineering*. 2002.

[13] J. Noll, and W. Liu. Requirements elicitation in open source software development: a case study. In *Proc. of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*. 2010

[14] I. Sommerville, and P, Sawyer. *Requirements engineering: a good practice guide*. Wiley. 1997

[15] K. Cox, M. Niazi, and J. Verner. Empirical Study of Sommerville and Sawyer's Requirements Engineering Practices. *IET Software*, 3(5): 339–355. 2009

[16] A. Begel, J. Bosch, and M. A. Storey. Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange and TopCoder. *IEEE Software*, 30(1): 52-66. 2013

[17] J. Cleland-Huang, H. Dumitru, C. Duan, and C. Castro-Herrera. Automated support for managing feature requests in open forums. *Communications of the ACM*. 52(10): 68-74. 2009

[18] P. Laurent, and J. Cleland-Huang. (2009). Lessons learned from open source projects for facilitating online requirements processes. In *Proc. of 15th International working conference REFSQ*. 2009

[19] G. J. Browne, and M. B. Rogich. An empirical investigation of user requirements elicitation: Comparing the effectiveness of prompting techniques. *Journal of Management Information Systems*, 17(4): 223-249. 2001

[20] O. Badreddin, T. C. Lethbridge, and M. Elassar. (2013). Modeling practices in open source software development. In *Proc. of Open source software: Quality verification*, 127-139. 2013

[21] G. B. Davis. Strategies for information requirements determination. *IBM Systems Journal*. 21(1): 4-30. 1982

[22] K. Herzig, S. Just, and A. Zeller. It's not a bug, it's a feature: how misclassification impacts bug prediction. In *Proc. of the international conference on software engineering*. 392-401. 2013

[23] J. Parsons and L. Cole. What do the pictures mean? Guidelines for experimental evaluation of representation fidelity in diagrammatical conceptual modeling techniques. *Data and Knowledge engineering*, 55(3):327-342. 2005

[24] A. Rantalainen, H. Hedberg and N. Iivari. A review of tool support for user-related communication in FLOSS development. In *Open source systems: grounding research*, S A Hissam et al (Eds.) Springer, Berlin, 90-105. 2011

[25] K. Crowston and J. Howison. The social structure of free and open source software development. *First Monday*, 10(2). 2005

[26] B. Bruce and J. Ritchie. Nurses' practices and perceptions of family centered care. *Journal of pediatric nursing*. 12(4):214-222. 1997

[27] R. K. S. Chu and T. Choi. An importance performance analysis of hotel selection factors in the Hong Kong hotel industry: a comparison of business and leisure travellers. *Tourism management*, 21(4): 363-377. 2000

[28] D. E. Damian and D. Zowghi. RE challenges in multi-site software development organizations. *Requirements engineering*. 8, pp.149-160. 2003

[29] K. Schmid. Challenges and solutions in global requirements engineering – a literature survey. *SWQD*, pp.85-99. 2014

[30] D. Gefen and M. Keil, M. The impact of developer responsiveness on perceptions of usefulness and ease of use: an extension of the technology acceptance model. *ACM SIGMIS Database*, 29(2):35-49. 1998