December 2006

Managing Quality in the Free and Open Source Software Community

Anas Tawileh
Cardiff University

Omer Rana
Cardiff University

Wendy Ivins
Cardiff University

Steve McIntosh
Cardiff University

Managing Quality in the Free and Open Source Software Community

Anas Tawileh
Cardiff University
m.a.tawileh@cs.cardiff.ac.uk

Omer Rana
Cardiff University
o.f.rana@cs.cardiff.ac.uk

Wendy Ivins
Cardiff University
w.k.ivins@cs.cardiff.ac.uk

Steve McIntosh
Cardiff University
s.b.mcintosh@cs.cardiff.ac.uk

ABSTRACT

The Free and Open Source Software (F/OSS) paradigm has attracted much attention in the past few years. Software developed by the community is considered to be a serious rival to proprietary software developed by commercial organizations. While the adoption levels of F/OSS artifacts within the community are quite high, end users from outside the community are still skeptical about adopting F/OSS. Claims against the adoption of F/OSS are usually rooted in the apparent lack of quality assurance mechanisms required to guarantee the protection of users' interests. This paper aims to promote higher adoption of F/OSS artifacts outside the developers' community by exploring possibilities to provide appropriate assurances that F/OSS artifacts will meet the quality levels expected by users.

Keywords

Free Software, Open Source, FOSS, Quality, Software Development, SSM, UML.

INTRODUCTION

The software development process adopted using the F/OSS paradigm often has significant differences from processes used in the proprietary software development industry. Software with F/OSS is built by the collaborative efforts of large numbers of distributed developers, with diverse cultural backgrounds. In order to facilitate communication between such developers, many mechanisms have recently been proposed, in order to tackle the difficulties of managing such distributed collaborations. However, quality concerns within the F/OSS community remain an important unexplored area.

End users usually argue that the F/OSS community is technology-centric, it has a distinct alien vocabulary and although claimed to be open for participation, the high level of requisite knowledge for access renders entry difficult. Because of these perceptions, users from outside the community are highly skeptical about adopting and considering F/OSS as a viable alternative to proprietary software. Consequently, user skepticism about the community has resulted in low adoption levels. However, this could be rectified if the comparison criteria would be based on the merits realized by the user in adopting the F/OSS option, compared to the use of proprietary software. Concerns about quality, and subsequent trust in the adoption of software developed through the F/OSS process remain significant limitations to adoption.

These limitations could be attributed to the relative failure of the F/OSS community to promote the quality of its artifacts. Although members of the community may argue that the quality levels attainable within the F/OSS community are much higher than those of proprietary software, these claims will not have any significant value unless supported by evidence-based assurances, and communicated effectively to users from outside the community. It is necessary to develop appropriate mechanisms to demonstrate quality to end users in simple, easy to understand terms, assuring them that the produced artifacts will satisfy their needs and requirements. In this way, the community could overcome issues related to quality, and encourage users to embrace a more open attitude towards the adoption and reliance on F/OSS artifacts.

This paper aims to investigate the possibility of improving uptake of F/OSS artifacts by developing appropriate mechanisms to assure end users that such artifacts will fulfill their needs and satisfy their expectations. The paper considers modeling of the overall F/OSS development process using Soft Systems Methodology (Checkland, 1999) and the Unified Modeling Language (UML) (Object Management Group, 2005), and defines particular metrics that can be associated with critical points along this development process. We also describe how such metrics could be used to improve the software

development process within the F/OSS community, and demonstrate how automated extraction of such metrics could be employed using F/OSS repositories such as sourceforge.net (Sourceforge, 2005). A simple prototype of the supporting information system has been developed to demonstrate the practical applicability of the proposed approach.

BACKGROUND

During the past few years, adoption of F/OSS has increased considerably. This growth was attributed to factors that encouraged users to favor F/OSS over proprietary software, such as cost savings, performance and security. However, levels of adoption differ between different organizations. While organizations and individuals who were involved in the development of F/OSS artifacts are observed to have high levels of reliance on F/OSS (Golden, 2004), others who had less experience with the development process are still relatively skeptical about embracing this software.

The development models that evolved within the F/OSS community proved to be highly effective in managing complex, highly distributed projects. F/OSS development models were based on the ideas of intensive communication between developers, large dependency on peer review, and frequent release of source code (Raymond, 2000). Developers believed that reliance on these ideas would accelerate bug discovery and fix release time, and substantially enhance the quality of the produced software.

These beliefs are perfectly sufficient for developers to convince themselves of the levels of quality and reliability they can expect from software developed within the F/OSS community. However, the community has mistakenly believed that all users of software are developers. This assumption is very far from being true. Although availability of source code can help with checking the claimed quality of the application, non-developers could not be expected to have the requisite time and knowledge to pursue such activities. Non-developers usually focus on evidence-based quality and measurable performance metrics that could indicate the level of trust that can be placed in the software, without going through the usually exhaustive task of performing code-level inspections. It is therefore necessary to extend the availability of source code with specific evidence-based metrics that could be used to evaluate a given F/OSS artifact.

Availability of source code is not enough to guarantee quality of software to users other than developers. Golden argues that "source code access is not usually enough for pragmatic IT organizations" (Golden, 2004), he claims that even when the software source code is available, there remains a question about who created that code. Golden suggests that F/OSS products contain code that was written by many different contributors, and that users have questions about the trustworthiness of the source code that results from these contributions (Golden, 2004). Consequently, many organizations that adopted F/OSS considered the various risks associated with this move to be tolerable.

Geoffrey Moore suggested a classification of technology users into three types: early adopters, pragmatists, and late adopters (Moore, 1999). Generally, F/OSS software is still used by early adopters, i.e. those who have sufficient technical knowledge to configure and build software as new versions and updates are released. However, because software is an essential element in the operation of any commercial entity, such organizations are usually less tolerant of failures and risks, they are not usually willing to undertake risk by adopting technology early, and they are mostly considered to be pragmatists. End users with less technical knowledge fall within the same category, their main interest is to achieve their goals with minimum effort, and they always try to avoid frustration they usually feel when a technical problem arises.

For F/OSS to become an attractive option to less technically-oriented people (the pragmatists), new mechanisms are required to signal the quality levels of F/OSS artifacts in a simple way that could be easily understood by users from outside the community. These mechanisms should enable the users to evaluate the level of confidence they can place in an artifact without going through the exhausting process of analyzing source code or delving into information contained within the artifacts' communication/documentation repositories. Golden describes this requirement as: "technology providers must cross the chasm to sell pragmatists… Open Source must cross the chasm as well" (Golden, 2004).

CURRENT EFFORTS

Many efforts have been initiated to improve the quality and uptake of F/OSS artifacts. Most current initiatives suggest a formal methodology to assess the suitability and maturity of F/OSS artifacts for a particular use within an organization. These methodologies are based on standardized analytical frameworks to compare multiple potential F/OSS artifacts in order to choose the most suitable option.

The Open Source Maturity Model (OSMM) was developed by Navica (Open Source Maturity Model, 2003) to aid organizations in determining the maturity of F/OSS artifacts (referred to as "products"). It proposes the assessment of a product's maturity based on criteria such as product software, support, documentation, training, integration and professional services. The methodology suggests a three phase process where the organization conducts assessments for each of these

criterion, followed by defining a weighting for each criterion based on the organization's requirements, and finally calculating an overall maturity score.

CapGemini Open Source Maturity Model (Duijnhouwer and Widdows, 2003) is another initiative to enable organizations to determine whether an open source artifact is suitable for their needs. The model suggests 27 indicators within 4 groups: product, integration, use and acceptance, upon which an overall score could be calculated to determine the maturity level of any open source artifact. The model also mandates a feedback phase where the effectiveness of the utilized indicators could be evaluated. It proposes that the continuous evaluation of indicators would provide the required calibration within a changing environment – particularly important for F/OSS.

Another recent initiative by Intel Corporation, the Centre for Open Source Investigation at Carnegie Mellon University and SpikeSource proposes an open standard to facilitate assessment and adoption of F/OSS artifacts, called the Business Readiness Rating (BRR) (Business Readiness Rating, 2005). This initiative intends to give companies a trusted, unbiased source for determining whether the open source software they are considering is mature enough to adopt, and help adopters assess which F/OSS software would be most suitable for their needs. The rating weighs the important factors for successful implementation of F/OSS software, and includes: functionality, quality, performance, support, community size, and security (among other factors). Initial analysis suggests that this effort provides more detailed evaluation data and scoring than the previously mentioned models.

All the previous initiatives aim to provide a systematic methodology to assist adopters of F/OSS artifacts in assessing and evaluating the suitability of the different available artifacts according to their own requirements. Although they use slightly different methods and techniques, they all rely on the same underlying concept. Such methodologies attempt to evaluate the quality or maturity of F/OSS artifacts using information that is already available. They rely on factors such as the number of releases, number of security vulnerabilities, reference deployments, market penetration and others. These factors provide useful indicators about the project's characteristics, such as maturity and quality, and could provide valuable assistance in the evaluation process. These approaches are mainly criticized because the evaluation of the proposed factors is highly subjective, the models proposed also attempt to evaluate F/OSS artifacts after they have been developed using information that is already available about the artifact. They do not provide any guidance on the improvement of the artifact itself or the enhancement of the F/OSS development processes in order to produce higher quality artifacts.

We propose a completely different approach to tackle the quality issues within the F/OSS community. The proposed approach is based on providing evidence-based quality assurance mechanisms that can facilitate the objective evaluation of different artifacts by end users. By providing sufficient assurances about the quality levels of F/OSS artifacts, users can better evaluate the suitability of any given artifact to their specific needs and requirements.

PROBLEM ANALYSIS

Defining problems in an open context such as the F/OSS community is not an easy task. The many interacting (and sometimes contradicting) technical, social and ideological aspects magnify the difficulty. Also, the field of F/OSS is relatively new, and although the amount of research performed is increasing at a substantial rate, the number of available empirical studies is comparatively low. To lay the foundation for the investigation, extensive interaction with both the F/OSS community and end users was conducted to determine the requirements of each party. Consequently, the main issues related to the quality aspect of the F/OSS development process were identified and formulated in the following questions:

- How can the maintainer of a F/OSS project accept contributions from community members in a timely manner while maintaining the required quality levels?
- How can the F/OSS community assure end-users that its artifacts will satisfy their requirements in order to encourage them to embrace these artifacts?

Figure 1 demonstrates the various processes involved in the F/OSS community, and in particular the types of interactions that take place between a project “Maintainer”, various “Contributors” and a non-developer “End User”.

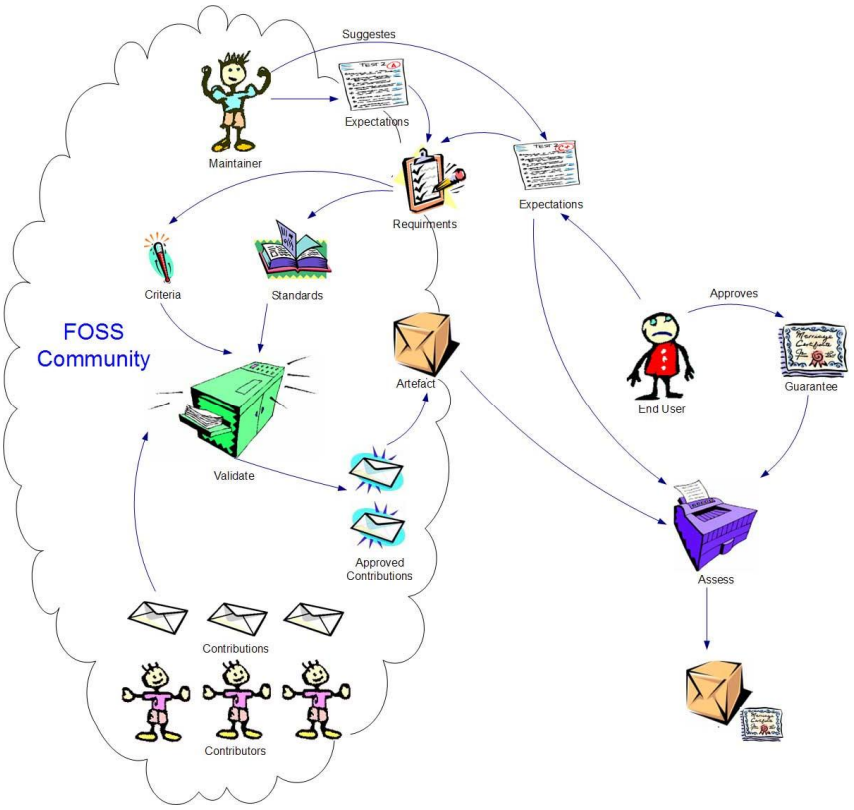

Figure 1. Coordinator, Contributor and End User Role in the F/OSS Community

MODELING

In order to provide appropriate answers to the previous questions which capture the essence of the quality issues in the F/OSS development process, modeling techniques are used to develop various conceptual models to illustrate the system that would provide such answers. These conceptual models would then be investigated to provide better understanding of the problem, and suggest possible actions that would improve it. Based on these findings, recommendations for suggested appropriate courses of action could be made. Appropriate actions might include the design and implementation of a supporting information system, the introduction of suggested changes or other possible solutions.

The problem at hand is characterized by complexity, involving multiple interacting elements and roles. The whole process is enacted by interacting human activities and aims to satisfy a particular need (the development of a specific artifact that fulfils a specific requirement). Existence of humans complicates the situation as every human actor has his own perception of the intended aims of the process, and his own distinct expectations, this will result in the process being perceived differently, with sometimes contradicting viewpoints (Checkland, 1999).

These characteristics of the problem situation seamlessly fit within Peter Checkland's definition of a “purposeful” Human Activity System (HAS) (Checkland, 1999). Checkland suggests the inclusion of different perceptions of the stakeholders when tackling problematic situations in Human Activity Systems by considering the worldview (Weltanschuuang) of each stakeholder. He proposed the Soft Systems Methodology (SSM) as “a general problem solving approach appropriate to human activity systems” (Checkland, 1999). SSM is particularly useful when the problem being considered limits the applicability of systems engineering techniques, and when it is difficult to identify measurable factors to assess the situation quantitatively. SSM suggests an intellectual construct called “Root Definition” (RD), which aims to provide a clear and unambiguous textual definition of the system to be modeled. It provides a way to capture the essence (root) of the purpose to be served (Wilson, 2001). Different root definitions could be developed to accommodate the different viewpoints or activities of the system. The developed root definitions are used as the basis of building a conceptual model of the system which is defined in these root definitions. The conceptual model should illustrate what the system should do to be the system described in the root definitions. It should include all the activities required and the logical links among them to achieve that goal. One weakness of SSM is that it only provides insight into what should be done to improve a particular problem

scenario, and does not provide any guidance on how to build the required support system or implement the proposed solutions.

In order to describe the proposed solution in such a way that facilitates its implementation, different tools and techniques should be used in conjunction with the artifacts of SSM. Several studies attempted to bridge this gap, including the work of Bustard, Zhonglin and Wilkie (Al-Humaidan and Rossiter, 2004) and Wade and Hopkins (Wade and Hopkins, 2002) who argue that SSM lacks the detailed information required to develop the intended systems, and that the use of the Unified Modeling Language (UML) (Object Management Group, 2005) would compensate for these weaknesses. Mingers proposed embedding structured methods into SSM (Mingers, 1992). We developed two root definitions to address the issues in F/OSS identified previously:

> *RD-1: A system to provide guarantees acceptable by end-users that developed artifacts within the F/OSS community will meet users' expectations by deriving appropriate mechanisms to assess these artifacts and produce such guarantees, while considering the resources constraints inherent in the F/OSS community.*
>
> *RD-2: A system to ensure that artifacts developed within the F/OSS community will meet determined quality criteria by deriving mechanisms to evaluate and approve submitted contributions to these artifacts against the acceptance criteria.*

Based upon the above root definitions, the conceptual model that represents the system described in these root definitions was developed. The conceptual model illustrates any activities that are required to specify the system described by the root definitions, in addition to any logical relationships between these activities. The resulting conceptual model is reproduced in Figure 2.

The conceptual model provides valuable insights, as it illustrates the required processes that should be identified and implemented to achieve the purpose captured within the root definitions. Comparing the required processes and the situation in the real world will describe the possible actions that could be introduced. Identifying these processes will allow requirements elicitation for the supporting information system which will, in turn, be utilized to support the implementation of the proposed actions for improvement.

Each activity in the conceptual model was evaluated to determine whether it currently exists in a F/OSS process, and how effectively it is performed. Such evaluation identifies missing activities that should be added, and activities that could be improved. This is performed by mapping each activity in the conceptual model to the real world of the F/OSS community.

Analysis of the conceptual model indicates a significant lack of end user involvement within the current development processes within the F/OSS community. In order to improve the quality levels of the F/OSS artifacts, end user requirements and expectations should be considered from the early stages of the project development process. Quality may be perceived differently by different parties (Michlmayr, Hunt and Probert, 2005), and our proposed approach suggests a systematic method to consider and integrate the different perspectives of developers and end users.

The F/OSS development process relies on the contributions submitted by volunteer developers who might become interested in a particular project. Because of the volatility of such contributions, not much confidence can be placed in their quality. Therefore, the maintainer of each project should establish strict quality standards and acceptance criteria for the approval of submitted contributions. These standards and criteria should fulfill end user requirements and expectations, and be communicated effectively to all potential contributors.

Validating each submitted contribution against the set of acceptance criteria can be an exhausting task, particularly when the project grows in size and scope. By proposing automated validation tools, the acceptance process would become more effective. When a contribution to the project is submitted, it can be validated against the acceptance criteria and a decision made about whether it is useful to be included into the final artifact. Validation results should be archived to enable tracking of any incorporated contributions.

In order to assure the end user that the developed artifact satisfies her requirements and expectations, the F/OSS community should provide sufficient guarantees to prove this compliance. These guarantees should be acceptable to end users, reproducible and produced in an open and transparent manner. In consultation with end users, project maintainers must propose and select possible measurement metrics that can be guaranteed for a particular artifact, in addition to mechanisms that are required to produce these guarantees. Such mechanisms may be used to provide the end user with appropriate guarantees to determine if the artifact passes the assessment, and complies with her requirements and expectations. Automated test tools, where available, should be used to support the whole process (examples of such automated testing tools include SimpleTest for PHP (SimpleTest, 2005) and JUnit for Java (JUnit, 2005)), which should be kept as open and transparent as possible.

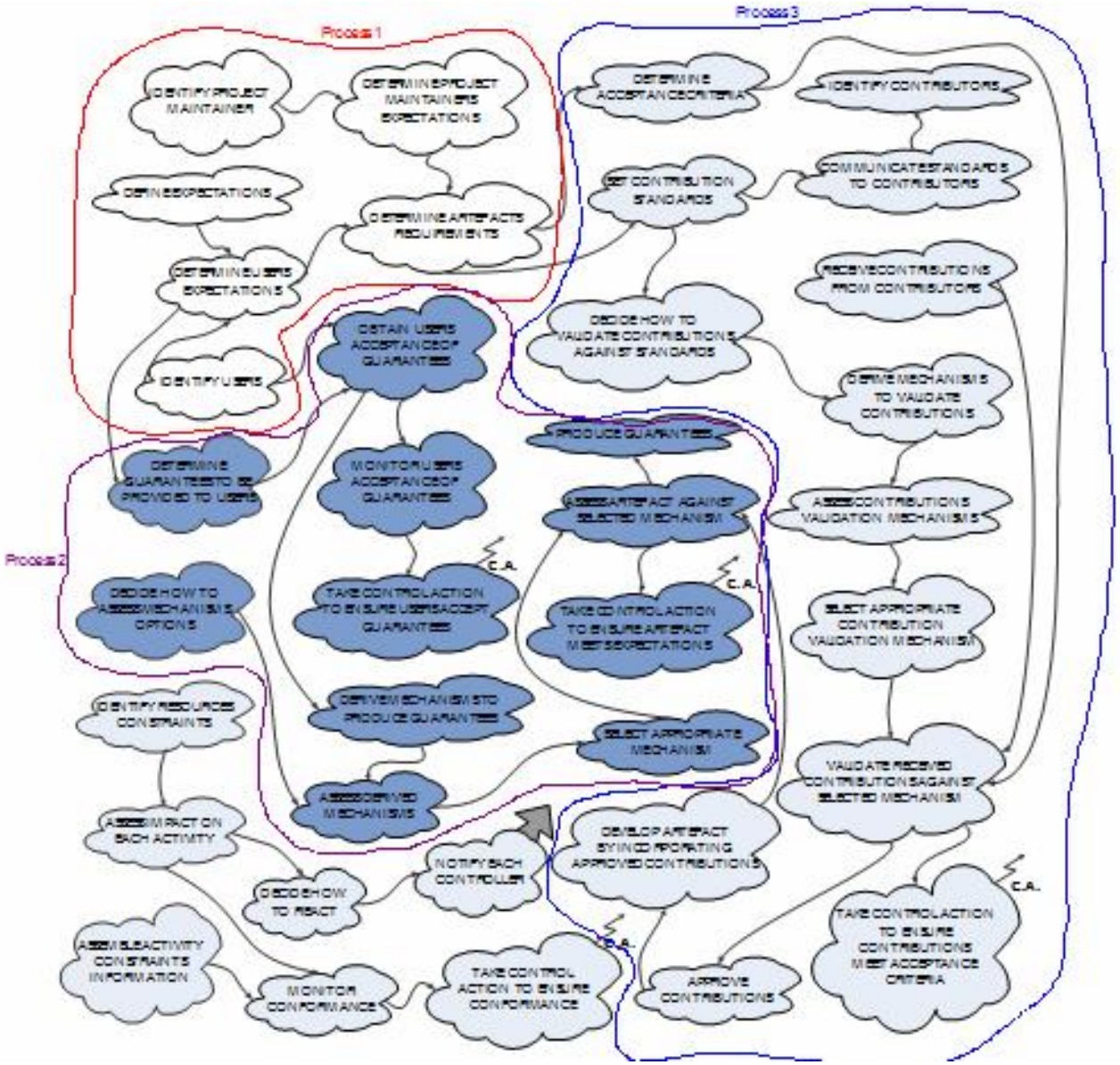

Figure 2. SSM Conceptual Model

A PROTOTYPE

A simple prototype was developed to demonstrate the applicability and functionality of the proposed information system that supports the implementation of the recommended courses of action. The prototype acts as an integrated platform to allow communication between developers and end users at the different stages of the project development lifecycle, and an easy and convenient way to communicate test results and quality information.

Because end users play a major role within the new system, technical complexity was kept to the minimum, and functionality was provided in as much user friendly a way as possible. Hiding technical complexity and jargon usually used within the F/OSS community will encourage end users with little knowledge of such complexity to actively participate in the development process by aiding the direction of F/OSS projects to better satisfy their own expectations.

Participation of end users starts from the very early stages of software development, by end users providing their initial expectations or requirements. The prototype allows the suggestion of new applications by both users and developers. It also facilitates the process of consensus building around the requirements, as described in the conceptual model (Process 1 in

Figure 2) by prioritization of requirements through discussion with end users. The project maintainer creates a list of prioritized requirements and kick starts the development. Project requirements will then advance as development proceeds using the current evolutionary development mechanisms. Figure 3 shows the interface for requirements proposition.

The next level of user participation is provided by facilitating the acceptance of the proposed assurances. The project maintainer, after having a clear statement of requirements that are approved by end users, will propose a set of guarantees that could be used to assess the adherence of the developed artifact to these requirements. Users will evaluate the suggested guarantees and approve what they deem sufficient. Upon obtaining users approval on some or all of the suggested guarantees, the project maintainer will derive mechanisms to produce such guarantees (in the context of F/OSS software development, guarantees for quality are usually considered to be satisfactory test results, and the mechanisms to produce the guarantees are usually testing tools). Users are then consulted to identify the most appropriate mechanisms to verify the criteria of success (for a particular artifact) from their perspective. After the selection of acceptable guarantees and the appropriate mechanisms to produce them, developed artifacts can be evaluated for conformance to requirements through the approved mechanisms, and guarantees are produced to confirm this conformance. These activities are described in Process 2 in Figure 2.

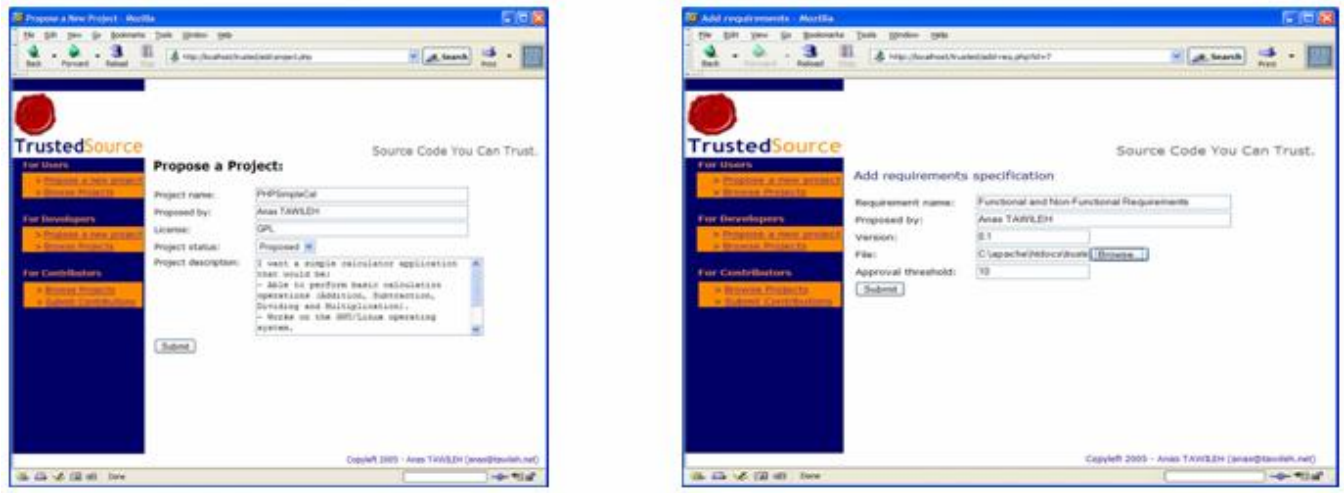

Figure 3. Prototype Screenshots – Project Requirements

The prototype also allows the acceptance of submitted contributions to the project by defining, at the early stages of project initiation, a set of contribution standards and acceptance criteria against which the maintainer will assess the quality of submitted contributions. These standards and acceptance criteria are then communicated to all potential contributors in order to maintain the required quality levels in their contributions. The maintainer will then propose appropriate validation mechanisms that could automate the assessment of the submitted contributions against the contribution standards (examples may include CheckStyle for Java and PHPCheckStyle for PHP (2005)). These mechanisms aim to confirm the conformance of submitted contributions to the identified standards. When a contributor submits some code for inclusion in the project, the maintainer can apply the validation mechanisms to the contribution in order to determine if it conforms to standards, and whether to include it in the main project. The prototype therefore allows the communication of validation results to both developers and end users, so that they can repeat the evaluation themselves and ensure the validity of the maintainer's decision. These results can be integrated in the current code repositories such as the Concurrent Versioning Systems (CVS) of the official release of the project to provide evidence-based assurance of the approved contributions' conformance to the project standards. The contributions management interface is illustrated in Figure 4 and is based on Process 3 in Figure 2.

Preserving the essential F/OSS process features of openness and transparency within the system is of significant importance. The developed prototype provides openness by not imposing any predetermined structure on the selection and use of testing and validation mechanisms. Users and maintainers can propose the most appropriate set of tools and categories and still benefit from the system as a central repository of quality related information. Transparency is supported by requiring maintainers to disclose any requirements and acceptance criterion, and to justify any decision they make and support it with evidence.

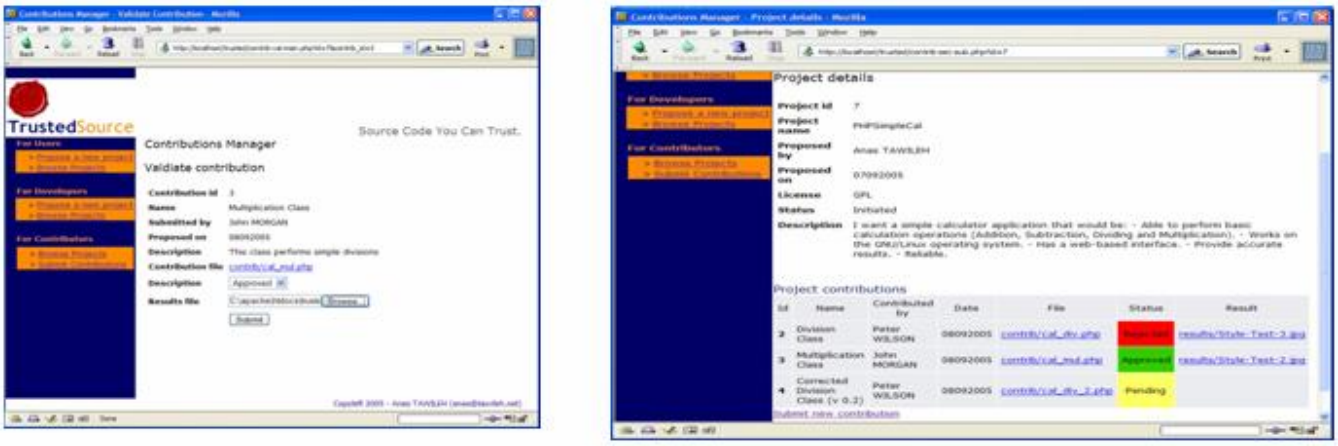

Figure 4. Prototype Screenshots – Contributions Manager

However, this prototype can be greatly enhanced by adding an integrated feedback system that could collect user and developer feedback, and include it as an indicator of the project's quality. Such a system would promote the continuity of user participation and quality improvement. In addition, quality metrics may be introduced to facilitate the evaluation and comparison of different projects – thereby providing a mechanism to compare quality across projects.

Our prototype provides an easy and convenient way to communicate testing outcomes and quality information; it could be improved by integrating the different proposed testing tools within a single interface to simplify the reproduction of any test and verification at a future time. Testing automation may be implemented where appropriate to accelerate the approval process and save human time and effort. Results produced by the assessment methodologies proposed by previous initiatives can be supplemented by the results stored within the system, to provide for better and more extensive evaluation that could overcome some of the shortcomings in each approach.

CONCLUSIONS

Free and Open Source Software (F/OSS) has attracted much attention during the past few years. It has been argued that the completely different development processed adopted by the F/OSS community produce higher quality artifacts than those developed within the traditional software development organizations. However, not much research has been undertaken into the quality aspects of F/OSS. The lack of adequate quality assurance guarantees that effectively communicate quality of a certain F/OSS artifact has resulted in a limited adoption of F/OSS outside its developer community. Many initiatives were proposed to tackle this problem by providing formal analytical frameworks that assess the maturity of an F/OSS artifact by analyzing information that is already available within a F/OSS project.

We propose a systemic approach to the analysis of the problem of F/OSS quality. It aims to facilitate the communication of evidence-based quality information to users outside the developers' community in order to overcome the obstacles inhibiting the wider adoption of F/OSS. The systemic analysis revealed the lack of end user participation in the early stages of the F/OSS development process, and suggested interventions that would facilitate end user involvement, and the consideration of their requirements and expectations as the basis of the development process. This will increase user confidence in the development process and will eventually lead to better adoption of the F/OSS artifacts.

We have developed a prototype of an information system that could support the introduction of the proposed interventions in order to assess the applicability and effectiveness of our approach. The developed system promotes higher levels of end user involvement during the different stages of the development process. It provides end users with a simple way to communicate their requirements and expectations to developers. In addition, it allows developers to communicate the quality of their developed artifacts to users from outside the F/OSS community in an evidence-based way, thereby reducing the alienation usually felt by end users, and increasing the adoption of F/OSS outside its developers' community.

The resulting increased level of acceptance will motivate other F/OSS projects to improve on their quality and user participation, either by using the same proposed platform and recommendations or by developing novel quality assurance and

communication mechanisms. Either way will promote higher quality in F/OSS development and better adoption by end users. Hopefully, this will enable F/OSS to "cross the chasm" (Moore, 1999).

In the context of benefiting from previous efforts and initiatives, the proposed solutions could be integrated within the currently available techniques and methodologies (such as the Business Readiness Rating – BRR- (Business Readiness Rating, 2005)) to increase the objectivity of their outcomes by adding more evidence-based elements to the assessment criteria. By combining the advantages of utilizing currently available project information to assess maturity and introducing new measure to increase quality, more defensible recommendations could be derived.

Another implication of this study is that by stimulating user-centric development processes, other aspects of quality that were overlooked to some extent by the current development practices (such as usability and ease of use) will by better understood, and the community will be guided towards the aspects that are usually more important from an end user point of view.

REFERENCES

1. Al-Humaidan F. and Rossiter, N. (2004) Business Process Modeling with OBPM: Combining Soft and Hard Approaches *Workshop on Computer Supported Activity Coordination (CSAC), 6th International Conference on Enterprise Information Systems*, April 13-14, Porto, 253 - 260.
2. Business Readiness Rating (2005) http://www.openbrr.org/, (accessed September 4, 2005).
3. Checkland, P. (1999) Systems Thinking, Systems Practice, Wiley, West Sussex, UK.
4. CheckStyle (2005) http://checkstyle.sourceforge.net, Oliver Burn, (accessed September 4, 2005).
5. Duijnhouwer, F. W. and Widdows C. (2003) Capgemini Expert Letter Open Source Maturity Model, Capgemini, http://www.seriouslyopen.org/nuke/html/modules/Downloads/htmldocs/osmm1.html, (accessed September 4, 2005).
6. Golden, B. (2004) Succeeding with Open Source, Addison Wesley, Indiana, USA.
7. JUnit (2005) http://www.junit.org/index.htm, Object Mentor Inc., (accessed September 4, 2005).
8. Michlmayr, M., Hunt F. and Probert, D. (2005) Quality Practices and Problems in Free Software Projects *Proceedings of the First International Conference on Open Source Systems*, 2005, Genova, Italy, 24 - 28.
9. Mingers, J, (1992) SSM and Information Systems: An Overview, *Systemist* Vol 14 (3), August 1992.
10. Moore, G. A. (1999) Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers, HarperBusiness, New York, USA.
11. Object Management Group (2005) http://www.omg.org/, (accessed September 4, 2005).
12. Open Source Maturity Model (2003) http://www.seriouslyopen.org/, (accessed September 4, 2005).
13. Raymond, E. S. (2000) The Cathedral and the Bazaar http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/, (accessed September 4, 2005).
14. SimpleTest (2005) http://www.lastcraft.com/simple_test.php, Marcus Baker, (accessed September 4, 2005).
15. Sourceforge open source repository (2005) http://www.sourceforge.net, (accessed September 4, 2005).
16. Spike PHPCheckStyle (2005) http://www.spikesource.com/projects/phpcheckstyle/, SpikeSource, (accessed September 4, 2004).
17. Wade, S. and Hopkins, J. (2002) A Framework for Incorporating Systems Thinking into Object Oriented Design *Seventh CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'02)*, May ,27-28, Toronto, Canada.
18. Wilson, B. (2001), Soft Systems Methodology, Conceptual Model Building and its Contribution, Wiley, West Sussex, UK.