EXPLORING THE OPEN SOURCE SOFTWARE DEVELOPMENT METHODOLOGY AS BARRIER TO ADOPTION

Joseph Nwankpa  
Dept. of Management & Information System,  
Kent State University,  
KENT, OHIO  
jnwankpa@kent.edu

Yaman Roumani  
Dept. of Management & Information System,  
Kent State University,  
KENT, OHIO,  
vroumani@kent.edu

ABSTRACT

The open source development methodology was seen by many as the development paradigm that would reshape the software industry as it boasted unique and unconventional characteristics that traditional development methodology could not match. However, despite all its potentials, the adoption of open source software at the end-user level remains moderate or at least failed to keep pace with the initial expectations. The aim of the paper is to identify and discuss major challenges for open source software, so as to facilitate end-users adoption. The paper argues that nature of OSS development methodology encourages the elimination of key activities during the design and development phase thus creating implicit adoption barriers.

Keywords: Open Source Software, Proprietary, Adoption

INTRODUCTION

A few years ago open source software (OSS) was seen as the panacea to proprietary dominance in the software industry. In fact, at a time the OSS model was thought by many as the model that would reshape the software industry as it boasted unique and unconventional characteristics that traditional software development model could not match. It was anticipated that open source software would genuinely challenge proprietary software market share. This new software development paradigm consisted of community of volunteers working together and collaborating to develop an end-product at no cost [5, 9]. The attention and interest in OSS lead to bold predictions at a time that OSS development paradigm would transform the software industry. OSS boasted unique characteristics such as reduced cost, rapid development cycles, high reusability and freedom to choose [7] compared to those of proprietary software. OSS seemingly
became the solution for companies seeking to optimize their IT budget while re-examining the return on investment. This new philosophy was heralded at a time as the gateway to adoption for companies looking to save huge licensing cost. In fact, earlier studies on OSS attempted to understand the motive and incentives behind community of developers who worked selflessly across geographical boundaries to develop free end-products [4] while some other studies examined the applicability of open source phenomenon outside the software domain [13].

However, despite all these potentials, the acceptance of open source software at the end-user level has remained moderate or at least failed to keep pace with initial expectations. At the user end, open source software continues to struggle to attract user-base. Indeed, some studies have examined this worrisome trend. On the one hand, proponents of open source model are quick to blame the big corporations and their proprietary walls as barriers to greater utilization of OSS. On the other hand, some argue that poor user interface, improper documentation, feature-centric development and programming for self are potential factors that hinder open source acceptance to the general public [12]. For instance, it has been noted that lack of apparent ongoing support for OSS project constitute significant barrier to adoption as users are increasingly apprehensive of the level of support for these products [9]. Nonetheless, with many companies such as Red hat providing support services for key OSS application, the level of adoption still remains slow. The severity of this is argument is brought to bear when we factor in the fact that OSS are suppose to be high quality, cheaper and more flexible when compared to proprietary applications.

This paper seeks to provide additional insight by examining the developmental methodology and processes of OSS. We argue that the developmental structure of OSS negates one of the fundamental processes in system development thus, leading to final products that may not represent the needs of mainstream end-users or greater user community. The central premise is that the duality role in OSS development where developers are also users contradicts the requirement analysis phase in system development. The resultant effect is developers who are technology savvy will rely on their software needs as a representation of mainstream user needs. Consequently, software developed from this process may struggle to meet the demands of users thus creating a huge barrier to adoption.

**OPEN SOURCE PROJECT**

Open source projects are typically initiated by individuals or small groups with an idea for something that is interesting to them. The common goal of an open source project is to create software that is useful or interesting to those who are working on it rather than to fill a commercial void [2]. Conversely, the open source development project may indeed be based on what developers find interesting rather than what is essential to wide-ranging users [14]. The project developers and designers are also the potential users of the software hence these projects are regularly driven by a need to use basis. Usually, the project initiators generally become the owners of the project. Open source projects typically engage in no active recruiting beyond simply posting their intended goals and access address on a general public website such as Sourceforge and Freshmeat. As a result, many open source projects have been created because some developers felt the existing project could not satisfy their software needs.
Due to the community approach to open source projects, it is inevitably that a gap will exist between community of developers who are typically technology savvy and non-developers who may not contribute codes, but are in need of software that provides a specific service [15]. If this is the case the question becomes to what degree can the software developed within the open source community accommodate the general users need? Many projects end up with software projects that satisfy the developers need, yet are so far from the need of programmers outside the project let alone the general users. This has created a scenario where some open source projects lack the depth to attract any meaningful user outside the domain of the project developers. Hence, creating a situation where open source projects continue to witness huge growth yet limited used and acceptance.

The quality assurance component of open source is predicated on developers working together as a community and making positive contribution. In his paper “The Cathedral and the Bazaar” Raymond argues that high level of quality demonstrated by open source software is partly due to the high degree of peer review and user involvement. Open source software claims methodological superiority over proprietary model because of its ability to attract programmer from across the globe [8]. Yet arguably quality can also be view from product usability and how the product fits the task that it is built to perform. The user centered design of OSS makes it difficult to articulate the quality dimension or at least to compare OSS to proprietary software that is design with main stream users in mind.

**Technology Adoption**

Technology adoption has been widely investigated within the Information System literature. For instance, the technology acceptance model (TAM) introduced by Davis has been widely applied to understand the attitude one has about the use of technology and subsequently, used to predict the adoption and use of information technology [1]. TAM suggests that two particular beliefs, perceived usefulness and perceived ease of use influences user’s computer acceptance behavior. Perceived usefulness is defined as the prospective user’s probability that using a particular system would enhance his or her job performance. Perceived ease of use refers to the degree to a prospective user believes that using a particular system would be free from effort [1].

Perceived usefulness of a project within the open source community can be measured by the size and duration of the project. This is usually the case as more interesting projects with higher potential will attract more developers and are usually more sustainable. Moreover, a study by Krishnamurthy found that the number of developers associated with a project was positively correlated to the age of the project [6]. The study also found that projects with more developers were viewed and downloaded more often. However, perceived ease of use is more complicated and difficult to attain using the open source model. One major shortcomings of open source software is its inability to create user friendly platform. Arguably, this is because open source model was designed to serve users who are computer savvy hence did not required an elaborate user friendly design. However, as open source software to moves toward mainstream users; these concerns become evident limiting open source ability to compete favorability with proprietary software. To mitigate these concerns, some companies such as Red hat, provide support services that makes it easier to install, use and maintain these software licensed under the GPL. However, this business model is only possible if critical mass and some level of user threshold can be achieved.
Open Source Development Approach

Open source development processes are fundamentally different from traditional software development in a variety of ways. However, one that clearly stands out is the requirement analysis aspect of the system needed to be developed. Conventional system development methodology outlines and encourages active interaction and feedback between users and system analysts in a bid to ensure that system meets end-users needs. In fact, new system development methodologies such as agile software development, rapid application development are aimed at generating feedbacks and integrating end-users with the designers and developers in an attempt to ensure that final product is consistent with users needs. It goes without saying that software companies go through series of prototype testing to elicit feedback from users in order to develop software that are consistent with end-users.

Nevertheless, for OSS software, the requirement analysis begins and ends with the developers who initiated the project and subsequent developers that will join the project mid way into development stage [15]. Hence, by so doing OSS development process omits requirement analysis a key activity in any software development. Most projects make limited use of software engineering approaches and practices such formal design procedures, specification and system prototyping [9]. The resultant effect of this process structure can be seen in two folds. On the one hand, developers are encapsulated by their pre-defined views of the software requirements which may not be able to capture the overarching requirements of member of the open source community who may not be developers but are in need of the software. On the other hand, the process ensures that developers’ ability to get positive feedback from non-developer members of the community is indeed eliminated. Software application by their nature can incorporate multiple functionalities, hence developing software capable of meeting multiple users requirement is not an unattainable goal. In fact, this is why proprietary software continues to have significant advantage over OSS. While proprietary software are able to serve the need of wide variety of users thus gaining the necessary critical mass and network externality needed to gain dominance, OSS which by its nature is tailored to meet the needs of few developers working on the project find it very difficult to gain wider use and acceptance. One the most critical factors cited as a barrier to OSS adoption is lack of support. Users are quick to argue that OSS lack reliable support services that accompanies commercial software. Such concern begs the issue of who was the intended user of the software and what nature of support is need. The duality role in OSS where developers and project initiators are indeed users ensures that support is as good as the system developed. These developers understand the system configuration and may not see the need for trivial support services either embedded in the system or outside the system. In fact, the project intended users based on the OSS perspective are technology savvy thus view support services as waste of manpower time and effort.

Furthermore, the dual role of developers in the OSS paradigm does create multiple project spread with the open source community. The number of open source projects keeps multiplying as developers continue to create project that will meet their specific needs and not necessarily the needs of wider open source community. An examination of projects within software categories in open source hosting sites reveals as many as a dozen projects with little variations. For example, Sourceforge.net which is the dominant open source project hosting website lists more than 150,000 projects and more than one million registered users. These projects span the open source
community with little variable creating a confusing situation where similar projects with limited
capabilities and functionalities are developed and maintained within the open source community.
Hence not only does duality role affect comprehensive understanding of software requirement
and feedback, but the process indirectly encourages a constellation of user requirements leading
to a plethora of projects. Giving these conditions, it is difficult to see how OSS can compete with
commercial software or for that matter gain wider acceptance and adoption.

Duality role and technology adoption

Technology adoption has always been based on the ability of the technology to meet the needs of
potential users [3, 11]. Software providers try to understand existing void in technology usage
and then design software capable of matching or to some degree filling the gaps. Hence,
requirement analysis is a critical process in software development methodology as organizations
and software providers attempt to ensure that product is consistent with the need. Requirement
analysis in system engineering involves those process that go into determining the needs of
users. It represents a description of how a system should function and a description of the system
properties and attributes. The software requirement analysis process thus covers the task of
modeling, analyzing these requirements as well documenting them as a basis for system design.
Studies have shown that improper attention to software requirement analysis during a software
development is a leading cause of software that fails to deliver the needs for which they were
designed. However, the OSS development methodology creates a unique structure where the
developers and end-users are indeed one and the same. This unusual situation means that
developers may not require requirement analysis as a way of understanding the end-users needs
since they are also the intending users. Thus the critical aspect of requirement analysis in a
software development is accelerated almost to the point of total elimination in the OSS
development methodology. Hence we propose:

Proposition 1: The dual role in OSS development methodology where developers are also end-
users will likely reduce the amount of requirement analysis in a project.

The perceived usefulness of a technology has been identified as one of the primary antecedents
to intention to adopt a technology [1]. Hence, systems and software applications are typically
tailored to meet the requirement of potential users. In the case of OSS, developers’ needs are
represent the needs of the overall potential users since the developers are indeed motivated to
design and develop a software application that will meet their current software need. However,
once the software is designed and developed and licensed under open source licensing scheme, it
is made available to the public who are free to download, use at limited or no cost. But the extent
to which such software application will meet the demands of the wider open source community
as well as users outside the open source community depends largely on the degree to which the
initial developers requirement conform to those of potential users outside the project. More often
than not, the needs and requirement these sophisticated technology savvy programmers are
usually inconsistent with those of end-users who typically lack such level of expertise in
software design and development. Therein lies the problem with OSS as potential users outside
the boundaries of the project struggle to grasp how such projects can indeed be useful to them. In
fact, these potential users struggle to understand these projects goals, usefulness and fit with their
current system needs. This argument becomes more evident when one examines many projects
spread across open source hosting websites, which were designed to meet the need and
requirement of a couple of developers or project owners. Such projects are indeed dead today
because first, they do not reflect or represent the needs and requirement of users outside the boundaries of the project and secondly, they scope of the project was too narrow to support any potential user base. Hence we propose:

*Proposition 2: The dual role in OSS development methodology where developers are also end-users will likely increase the number open source projects inconsistent with wider user requirement.*

The very nature of software products enables multiple functions to be embedded in a single software application. In fact, commercial software providers are able to leverage this multi facet ability of software to develop applications that can accommodate the different user requirements. Thus proprietary software providers are able to widen the software scope and requirements while capturing the needs of multiple users. OSS are also rich in functionality a study by Nichols and Twidale found that usability of OSS was a key barrier to adoption and not functionality of the software [10]. The study noted that OSS development methods followed a process of continuous improvement with emphasis on functionality and not usability. However, proper usability design can only be achieved if potential end-users requirements are taking into account when developing and designing the software. Hence we propose:

*Proposition 3: Having procedures in OSS development methodology that encourage the understanding of wider user requirements will increase usability and enhance adoption.*

**DISCUSSION AND IMPLICATION**

This paper offers a new perspective on how OSS development methodology creates adoption barriers for users outside the community. We argue that the reason OSS continues to struggle with end users adoption has more to do with the development process than the usability of the software. Hence if OSS are to exploit their full potentials in term of wide spread acceptance, they need to address development procedures and processes. By ignoring requirement analysis which is a critical aspect in software engineering, open source developers and designers alienate users outside the project boundaries and community. Thus, while many OSS projects may meet the needs of the few developers and contributors of the project, the same cannot be said of potential external end-users who will grapple with the software relevance and fit with their current software needs. In addition, inadequate requirement analysis does create issues of project proliferation within the open source community. The limited requirement analysis during the design phase of the software development means that many OSS projects are just tailored to meet few developers need creating scenarios were projects are started in a bid to meet those needs not included in the existing OSS projects. In fact there are lots of projects that are not fundamentally dissimilar yet continue to exist alongside each other. The chances of these projects attracting enough developers and contributors remain questionable at best. Indeed with multiple projects, ability of the open source community to pull talents and contributors across geographical boundaries may not be sustained thus leading to abandoned projects, low quality software, reliability issues and end-users rejection of OSS.
In order to gain the full potentials of OSS in terms of widespread adoption, the open source community need encourage processes that are supportive of requirement analysis. The initial design should be more involving than it is currently as this will ensure that existing projects meet the demands of wider end-users and hence encourage adoption.
References:


