Development of Open Source Software, a Qualitative View in a Knowledge Management Approach

Luã Marcelo Muriana[1], Cristiano Maciel[2] and Ana Cristina Bicharra Garcia[1]
[1]*Institute of Computing, Federal Fluminense University UFF, Voluntários da Pátria Street, Niterói, Brazil*
[2]*Department of Computing, Federal University of Mato Grosso – UFMT, Cuiabá, Brazil*

Keywords: Collective Intelligence, Open Source, Knowledge Management, Quality Assurance, Software Engineering, Community.

Abstract: Open Source Software (OSS) is software that users have freedom to modify and share it with no cost whatever their intentions. A major feature of this kind of software is its development in public, where the collective intelligence (CI) is applied and the knowledge is shared. The communication is a fundamental activity to these settings of development. To support the communication process, knowledge management (KM) stimulates the communication and the information sharing among people. This way, a good communication among users that are stimulated and coordinated addresses the final quality of the open source project. This work surveys how KM stimulates quality assurance in developing open source settings. It focuses on users, on the communication among them, and on the documentation they can help to write.

1 INTRODUCTION

In a traditional process of software development, people stay in the same place developing activities inherent to its process. But, in the last years Distributed Software Development (DSD) emerged, in which various organizations started to develop software with a team of people from different geographical areas.

In addition to this concept there is another one of CI, that for Malone et al. (2009) means that different groups work together on a way that seem intelligent.

CI has a new meaning in the last years, especially with the advance on the web application 2.0. The diffusion of these simple and easy technologies lets users interact. Currently, the users' contributions are treated as a valuable factor for CI. Users are also encouraged to contribute with content, interact with other users and exchange knowledge (Hwang et al., 2009). For DSD, the organizations follow standards established by themselves and traditional software engineering. In contradiction to that, CI in Information Technology area emerges in a scenario that OSS is highlighted.

OSS software is made available to the users to change and distribute the software for any purpose, and with no cost (GNU, 2013). Free software is the type of software that the author can attain a license for operation, copy, change and distribution. These licenses are called copyleft. Copyleft is less restrictive than copyright license, but various free licenses impose restrictions on free code; for example the joint use with closed code; or the imposition of obligations as “changes on distributed code should be available as source code”.

One of the characteristics of free software is the public development, using CI.

However, when people think of Distributed Development of OSS (DDOSS) some issues about the software engineering emerge: How is the process of software engineering executed? For Audy and Priklandnicki (2008) the methods of software engineering are structured approaches that give details of ‘how’ develop good software. For the authors, the methods involve a set of steps: planning and project estimation, requirements analysis, data structure design, architecture and program algorithm processing, coding, testing, maintenance, and others.

But in environments of DDOSS, according to Noll and Liu (2010) and Scacchi (2002), traditional software engineering is not applied.

If classic software engineering is not applied in open source development process, standards of quality in software would be rarely followed in these. For Shaikh and Ceron (2007), communication and effective management, programming language

Marcelo Muriana L., Maciel C. and Cristina Bicharra Garcia A..
Development of Open Source Software, a Qualitative View in a Knowledge Management Approach.
DOI: 10.5220/0004962903910399
In *Proceedings of the 16th International Conference on Enterprise Information Systems* (ICEIS-2014), pages 391-399
ISBN: 978-989-758-028-4

and the choice of test strategy are three factors which most affect the quality of OSS.

Communication, the focus of this research, can be regarded as a fundamental activity for development environments. But, as it does not happen in the physical presence, it should be encouraged and facilitated, once it is one of the problems of CI and, consequently, the understanding of information (Hwang et al., 2009). These problems affect the quality of the final product developed.

The management of knowledge, aiming to support this communication process, encourages the communication and information sharing. Good communication with users who are encouraged and coordinated, affects the final quality of the open source project.

This study therefore aims to analyze through literature research, how OSS can be developed throughout the process practices to assist the quality assurance of software and how to tailor it to the reality of these communities. The focus of this analysis is the study of what the authors consider the basis of open source communities, users and communication among them. Then, this analysis about the quality is based on KM, a subject which encourages the diffusion of knowledge.

The other part of this study is structured as follows: in Section 2 some studies related to these issues are presented; Section 3 considers open source communities; Section 4 discusses software engineering and the documentation on the development environment; Section 5 deals with social media tools and how it supports the communication among people; and Section 6 discusses quality assurance of software based on KM and members of open source communities.

2 RELATED STUDIES

This study had the aim to analyze open source communities and to know how to assure quality of software considering KM on communication process among users. For that purpose many studies have been carried out.

Zhao and Elbaum (2000) conducted a survey that aimed to: i) find out techniques of quality assurance used in open source development, ii) determine factors that affect quality assurance activities; iii) understand the perception of open source developers regarding quality assurance. The study was limited to the process activities as the whole process, instead of each step, being focused on software tests.

Tosi and Tahir (2013) analyzed 33 open source projects well known to understand how developers develop quality assurance on their open source projects. However, as with Zhao and Elbaum (2000) the focus was on Software Test.

Michlmayr et al. (2005) although, presented factors that contribute to quality assurance, as problems that interfere on its practice, the focus was on the process as itself. The communication among the users and the information sharing among them was not considered as an important factor for quality assurance.

Spinellis and Szyperski (2004) conducted a study concerning how the reuse of coding can contribute to quality assurance because this practice promotes more developers to see the same code, detecting problems and then correcting it.

Shaikh and Ceron (2007) investigated factors that have influence on open source quality and the relation among these factors. From this study, authors mentioned three main factors about basic characteristics of OSS quality: access quality, development quality and design quality. Although some criteria as availability and document updating have been mentioned, the authors did not report the importance of user participation and the communication process among them for this activity.

Aberdour (2007) presented an overview concerning the assurance quality process during whole open source development. However, KM was not mentioned.

Concerning the quality, no other study was found that considered the basis of open source community, users and communication, or KM as a fundamental theory for these communities.

3 OPEN SOURCE COMMUNITIES

OSS is a type of software which users can change and distribute with/without cost for any purpose (GNU, 2013). In a developer’s perspective, Spinellis and Szyperski (2004) says that OSS is the combination of two important properties: visibility of source code, and the right to derive the product from the original. OSI (2013) commented that open source does not mean that the user owns the access to the source code of one software. In accordance with GNU (2013), OSI (2013) says that a program is free software when users may execute it for any purpose and study how the program works and adapt it to their needs. Also, users may redistribute copies of the program.

Totally disagreeing to closed codes, and joining the criteria defined by OSI (2013), groups of people started to discuss the development of OSS in communities created exclusively for this purpose. Open Source Development is a revolutionary development model (Bayrak and Davis, 2003) because it allows people that are geographically distributed to work simultaneously, or if they wish, to work on the same project with common interests.

It can therefore be concluded that open source communities use the concept of CI as a tool to develop a joint work (Malone et al., 2009). For Porruvecchio et al. (2010), OSS can be seen as a result of sharing knowledge among people from a community. In many existing characteristics of communities that use CI as knowledge source, Hintikka (2008) and Hwang et al. (2009) highlight: the opinion diversity among people who join the group, existing tools that support knowledge sharing, participant independence and decentralization, which is linked to DDS.

However, open source communities are not focused on software documentation, and because of this the documents tend to be incomplete. The biggest focus of these communities is the universe around the code development (Porruvecchio et al. 2010). Therefore, the documentation needs to exist. It is not necessary to describe the whole system, but it needs to be able to clarify users doubts (Berglund and Priestley, 2001).

4 SOFTWARE ENGINEERING FOR OSS

Sommerville (2007) says that the process of software development consists of four basic steps: software specification, software development, software validation and the evolution of the software. Although, Noll and Liu (2010), Scacchi (2002) and Noll (2008) affirm that supporters of open source development do not use all the basic steps. Some studies have then been made to understand how the activities of software engineering are made on open source development, when the traditional process is not applied.

Studies carried out by Noll and Liu (2010) and Noll (2008) aimed to understand how the requirements are elicited, documented, accepted and validated in small open source projects. Through the analysis of elicitation in the web browser Firefox, it was found that the majority of the characteristics are elicited by developers based on their own experiences, or the knowledge of the users' needs. The authors highlighted that requirements are informally discussed and its validations happen through discussion between the developers, and rarely include users. The authors say also that the documentation consists of only discussion files.

Scacchi (2002) focused on how requirement engineering is applied on OSS development. The study analyzed four open source communities and in its results realized that many types of activities that are used on this kind of development are equivalent to traditional requirement engineering. However, to support these activities, the authors mentioned web applications such as email and boards are used as support tools. The authors highlight the use of informal language to describe the requirements. They allege that participants of the development community comprehend and easily condense the idea when the information is written in a succinct and informal manner. Lethbridge et al. (2003) confirmed this observation saying that as the more abstract the passages of the documents are, the more valuable and useful they are considered by the users.

Therefore, there is no process of open source development that is accepted worldwide (Acuna et al., 2012). Each community open source uses its peculiarity for the process, even though all of them have the user as an important source of knowledge.

4.1 Software Documentation

To obtain high quality software, even if there are documents to support the process, is not easy. However, the documentation can improve the quality of software code and the communication between the members of a community (Dagenais and Robillard, 2010).

It is highlighted that, in this study, when documentation is mentioned it can refer to traditional documentation of software engineering and to documents as manuals which describes how to use a specific product.

To understand in which circumstances the documentation of open source communities are created and kept, Dagenais and Robillard (2010) analyzed 19 documentations of ten open source projects and interviewed writers and readers of those documents. The first proviso is that creating and keeping these documents represents a big effort, because it is not known which factors are considered when documents are used and how these documents influence the project.

For Treude and Storey (2011), developers had a negative view of the documentation once written, it only occurred because of the necessary official requirements, and then it was almost always incomplete and not actualized. In addition, Parnin et al. (2005) said that when a document is written it quickly becomes old and is therefore distrusted by the users who do not use it.

To aggravate the situation, Lethbridge et al. (2003) regarded that sometimes the documents are not updated. The changes are registered only when the alterations present a big difference from the actual documentation

As an attempt to reduce the problems of an incomplete or delayed documentation, Berglund and Priestley (2001) said that even the documentation is of a specific activity, it can be opened to the community involved on that software through email lists and forums, where readers and writers can debate and dialogue questions. As many users are involved in this writing process, the discussions performed have more probability to be a source for new requirements to be elicited, letting the documentation evolve. Parnin et al. (2005) reported that people are capable to produce a great source of content and that this documentation is seen by many users.

Therefore, the key factor for documentation and for the whole process of OSS development is the communication. In open source communities the communication is fundamental to improve the group work (Porruvecchio et al., 2010).

5 SOCIAL MEDIA AND COMMUNICATION

In a study analyzed by Parnin et al. (2005), the authors highlighted that because of an incomplete documentation the users search for interactive media on the web to try to solve problems that arise during the development and/or use of some tool.

For Begel et al. (2010), social media has changed the way that people collaborate and share information. According to the authors, the traditional process of software engineering involves a big time spent at work for communication developers. In an attempt to decrease this problem, some web applications (email, file sharing, and communities) have the ability to improve workers' communication. With web 2.0, the use of social relations expanded the production and sharing information.

For the sharing information process, the communication between the members has a fundamental role, increasing and improving the work group, promoting a collaborative environment. The collaboration occurs in all levels of community participants and has as an advantage the diversity of skills, proposals and suggestions, and the development of higher quality *software* (Porruvecchio et al., 2010).

However, *open source* development requires a framework that allows the community to cooperate, develop and capture the qualities of this type of development (Berglund and Priestley, 2001). The communication in these groups can occur supported by various tools: chats, forums, wikis, email lists and others. These tools support idea exchange, helping request and information sharing, as an important indicator for the success of *open source* projects (Porruvecchio et al., 2010).

Through social media, a new way for knowledge exchange emerged (Treude et al., 2011). A diversity of studies analyzed various tools of web 2.0 which are highlighted under software engineering view, as follows:

- **Blog:** a website where structure is quickly updated from additions that are called articles. Many subjects can be discussed and readers can provide feedback. Feedback helps to improve software. Treude and Storey (2011) said that blogs are easily created and kept, but are not always enough.
- **Social Networks:** social structures composed by people or organizations, linked themselves by one or many kind of relations, who share common values and objectives.
- **Communities Q & A (*Questions and Answers*)**: Consists in environments focused on questions and answers. Websites Q & A became knowledge databases distributed among many people, differently skilled and specialized (Treude et al., 2011). Q&A environments are tools which help the discussion process for the development of software as a product. Parnin et al. (2005) highlighted that the speed to get answers on these environments is very quick. Regarding software development, this kind of website promotes the knowledge exchange among programmers via the internet, and according to the authors it can substitute the documentation when it is scarce or non-existent (Treude et al., 2011).
- **Wikis**: A specific kind of document collection in hypertext or collaborative software used to create it. Dagenais and Robillard (2010) said that among many advantages of this tool, it is an easy

way to create documentation which allows anyone to contribute to the documentation. However, through prior studies, authors affirm that wikis are abandoned by the users because they are not controlled environments compared to others and, consequently, allow SPAM and information inconsistency.

Beside the applications mentioned before, Berglund and Priestley (2001) mentioned email lists that as Q&A environments help the discussion process for the development of software. Begel et al. (2010) also highlighted the use of microblogs that have reduced the number of characters in each interaction, and because of that it is a tool used by the participants to share links, make appointments, keep the developers aware of the activities, etc

Social media present various advantages on software engineering (Begel et al., 2010):

- Social networks normally provide a complete environment for communication.
- On these social media, work teams of software engineers expose their goals and ideas.
- Users who join social networks would know how to communicate and coordinate to develop a product successfully, requiring tools that assist the management and transfer of knowledge, and let mutual collaboration occur. The idea is that the community increases because the knowledge distribution becomes more efficient and quick, minimizing misleading information among coworkers who cannot meet in person.

The authors said that a condition for social media to bring benefits is the planned use of those tools.

In *open source* communities, KM is knowledge sharing and free access to information. For the sharing step, the communication among the members has a fundamental role, increasing and improving the work group and then promoting a collaborative environment (Porruvecchio et al, 2010).

6 SOFTWARE QUALITY IN OPEN SOURCE DEVELOPMENT

Each open source project is different from the others and has its own particulars (Porruvecchio et al, 2010). However, the basis of these projects is the same: users and communication among them. By the way, the authos affirmed that users should be competent to understand and contribute with the highest level of details on these projects. Then, KM aims sharing of knowledge and mutual help.

6.1 Knowledge Management

KM is an activity supported by learning processes by capturing and reusing past experiences. It is a unique activity because of the focus on each person and his/her ability, which are systematically shared in the organization (Parnin et al., 2005).

In the context of OSS development, sharing is the way that more people can assist the development process of software (Rus and Lindvall, 2002).

The knowledge can be captured in many ways: traditional manuals, videos, wikis, blogs etc. (Treude and Storey, 2011).There are many kinds of knowledge, and then a variety of tools should be used to deal with this great source of knowledge.

In a study carried out by Porruvecchio et al. (2010), an email list of developers of 70 Open source projects that were hosted on SourceForge.net was analyzed. The authors said that the understanding of communication of members of this group can help to improve efficiency and quality of projects. The results showed that each member communicates to at least one other user, that there are one or two developers who assume the main roles on the project and that there is one user who communicated to the whole community. The last user has the role of managing the knowledge among all community members once the virtual environment became a learning space through requests for explanations and/or others members' assistance.

Therefore, the authors checked the importance of *peer support* on these communities. *Peer support* consists on a mutual helpful relation between two members. Endres et al. (2007) affirmed that *peer support* is fundamental for *open source* communities and report its essentiality to increase knowledge sharing.

Rus and Lindvall (2002) said that organizations should use knowledge learned from past projects to decrease time and cost in the development process of new products. Although the authors noted this affirmation for organizations of software development, it can be applied to open source communities once individual experience of users can be converted to knowledge for the development of new projects.

Therefore, KM has a fundamental role in practices of open source communities (Endres et al., 2007), improving quality of performance, since they promote an important contribution to build common knowledge basis (Porruvecchio et al, 2010).

6.2 Users

A basic element for open source communities is the user. It is through the users and their interests that these communities develop their projects. Porruvecchio et al. (2010) said that certain groups of people are always part of a community and providing the bases around a growing project.

Although the authors stated that participation in open source community is open to everyone who wishes, they highlight that control it is necessary. One method pointed by them is the distribution of levels of participation in the community: some users have more permissions than others, but anyone who wants to join the community is allowed. Then, a social structure is pointed as a tool to accomplish this control (Berglund and Priestley, 2001).

Porruvecchio et al. (2010) separated user group into five levels: Users, Advanded Users, Errors Repairers, Developers, and Manager.

Along the same creating thoughts of a social structure in an *open source* community Spinellis and Szyperski (2004) reports the Onion Model, which divides users into four levels: common users, error reporters, developers and core team?

Regardless of the social structure adopted in an *open source* community, it is important to realize the role of these users in those groups. Porruvecchio et al. (2010) said that the developers discuss problems they find during the development of a particular feature or during a *bug* repair, or while users request help to solve difficulties of using the software, or warn about errors and *bugs*. Therefore, dealing with such a wide variety of contributors, there is a great sharing of knowledge and, consequently, the project tends to be more powerful (Khanjani and Sulaiman, 2011).

According to Porruvecchio et al. (2010) and Khanjani and Sulaiman (2011), the participation of all kinds of users is encouraged as a practice that should be encouraged on development environments, regardless of the reasons that lead the user to interact in these communities.

In a study carried out by Parnin et al. (2005), they proposed a model of *crowd documentation* in a large group of contributors which collaborate to the documentation of API, the authors showed that the documentation can emerge from questions and answers. This also happens because the proposed model encourages the participation of users through the idea of awarding the best answer. Therefore, it supports the process of quality assurance.

According to Dagenais and Robillar (2010), in a study about how OSS documentation is created and kept, the community is encouraged to join the written process through questioning. These questions help to repair bugs and then update the documentation. The authors stated that community feedback is essential because it helps to localize which part of the document needs to be clarified.

However, Khanjani and Sulaiman (2011) highlighted that to have many volunteers in *open source* development requires a centralized organization to coordinate activities and do maintenance on the product. The author also stressed that users help to improve the quality with more correctness, completeness, safety, and quality requirements, which justify the use of a social structure proposed by Porruvecchio et al. (2010) or by (Spinellis and Szyperski, 2004).

Therefore, for users to share knowledge, and together, support the quality of the final product, the communication is the starting point to develop an efficient team (Porruvecchio et al., 2010).

6.3 Quality Assurance

Currently, software has been one of the most requested products on the market. “The concern about quality has become an essential requirement. This is a basic idea to ensure software functionality with minimal errors, defects and greater satisfaction on quality expectations” (Maia, 2003).

Pressman (2000) said that the quality of software is defined as the conformity to explicit functional requirements and specified performance, following standards for development of documents and follow good practice of *software* engineering.

For open source development, developing with quality depends on two factors: code revision and testing data (Khanjani and Sulaiman, 2011). For Shaikh and Ceron (2007), the access to the code is fundamental for open source development, as it allows the developers to have a high quality contribution and makes the code available for anyone to analyze it and detect bugs. To support quality the seeking of various tools as emails’ list and tools of management settings can be used (Khanjani and Sulaiman, 2011). However, the authors say that when we think about quality, some aspects should be considered: level of service to be improved, productivity and satisfaction of final user. If these aspects are considered in software development, the system efficiency for users and developers will increase, and productivity also, once users and developers are motivated to develop better products and to find problems on the code developed.

However, for Khanjani and Sulaiman (2011), seeking process for quality assurance on this kind of development has some problems:

- There is no formal design for OSS development and poor designs have poor codes and therefore poor quality. The authors state that the communication process and an appropriate structure are important, and it attracts developers to cooperate, but it is necessary to be attractive;
- The lack of knowledge of community participants to repair bugs;
- The quality can be affected by the lack of documentation. The documentation is focused on the programming style desired, which will assist new users to know the system and understand the modifications and evolution on the code.

Michlmayr et al. (2005) added problems of open source development as lack of volunteers, whicj consist in a problem that some projects need to deal, mainly the unpopular ones. The majority of volunteers are only aimed at the code development. There are not many people who want to help with tests. Besides, the communication is another problem which brings negative impacts to quality, for example when bugs are not correctly reported.

Regarding documentation on knowledge of community members and communication among users tend to help with solving previously reported problems.

Berglund and Priestley (2001) said that the written process of documentation of many projects can be elaborated using discussion topics of users. These topics guide the documentation providing information and then the documentation obtained in the final process is focused on the user and its quality tends to increase. However, care must be taken to not transform the documents on a repository, hindering the user to find the desired information.

As a development perspective that states free code, the communication among the members of these communities needs to be facilitated (Porruvecchio et al., 2010).

The connection among the members creates a network in which sharing information facilitates goal achievement and problem resolution (Porruvecchio et al., 2010). In this way, KM becomes the basis to promote sharing information as common practice in these environments. The participants have a variety of roles in open source communities, and their activities are complementary: they trust each other to improve the final product. It is important to maintain the contact, share information and give/get feedback. Raymond (1999) said that a high quality level must be attributed to the level of relation between the members of a community.

Management and knowledge transference are challenging activities that are essential to integrate new collaborators in a project (Treude and Storey, 2011). However the authors report a study which says that the documentation is not always useful and is almost always outdated, making knowledge transference difficult.

The number of messages exchanged among the members in a community is an indicator of success for the project as it shows interest of the community on its development. In this way, an effective communication tool is fundamental in these environments (Porruvecchio et al., 2010).

In a study carried out by Parnin and Treude (2011), it was found that blogs are the most common means of communication covering almost 90% of the subjects of this theme. The authors analyzed the types of posts and found that the majority were regarding tutorials followed by experience reports. The analysis of the posts also showed that 81% of the posts contained comments, building interrelationships between authors and users. This interaction resulted in improvements on code and documentation.

Therefore, the importance of KM in development environments was shown because it encourages the sharing of information and then the good communication among the members. Rus and Lindvall, (2002) affirmed that sharing knowledge is a risk prevention strategy to that is generally ignored.

7 CONCLUSIONS

This study aimed to present a discussion about free software communities and KM in these communities, concerning the process of *open source* software and its peculiarities, to ensure the quality of the product developed. The study was based on the purpose of CI and KM, to input quality throughout the process of software development on factors that are considered primordial for this study, users and communication among them.

Users are the main reason to open source development exists. If they do not fell themselves motivated and encouraged to make part of the open source community, as mentioned, the final quality can be affected. As many studies reported, it is important to encourage users' participation, and for that, to create a hierarchy among the users to determine their participation and efficient ways to promote good communication among the community participants.

The communication among the community members are fundamental to the DDS and thus, it needs some ways to provide the interaction among users. Through several social midias, users exchange knowledge, and it can improve the quality of the software. So, KM, besides just promoting the communication among member of the community, must support the knowledge exchange, which can be used for diverse purposes, but it is worth highlighting that do not mind its purpose, its existence is a factor that assists the process of quality assurance.

Quality Assurance is an activity that must be considered throughout the development process, in free or proprietary software. In this way, mechanisms that promote the interaction among users should be able to promote the exchange knowledge.

REFERENCES

Aberdour, M. 2007. Achieving Quality in Open-Source Software. *In Software, IEEE*, vol.24, no.1, pp.58,64.

Acuna, S. T., Castro, J. W. and Dieste, O. 2012. Juristo, N., A systematic mapping study on the open source software development process, Evaluation & Assessment in Software Engineering *(EASE 2012), 16th International Conference on* , vol., no., pp.42,46, 14-15.

Audy, J. and Priklandnicki, R. 2008. Desenvolvimento Distribuído de software. Rio de Janeiro, Elsevier. <http://books.google.com.br/books?id=znis1KYslRAC&printsec=frontcover#v=onepage&q&f=false> Accessed in 29/03/2013.

Bayrak, C. and Davis, C. 2003. The relationship between distributed systems and open software development. Commun. ACM 46, 12 (December 2003),99-102.

Begel, A., DeLine, R. and Thomas Zimmermann. 2010. Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*. ACM, New York, NY, USA, 33-38.

Berglund. E. and Priestley, A. 2001. Open-source documentation: in search of user-driven, just-in-time writing. In *Proceedings of the 19th annual international conference on Computer documentation (SIGDOC '01)*. ACM, New York, NY, USA, 132-141.

Dagenais, B. and Robillard, M. P. 2010. Creating and evolving developer documentation: understanding the decisions of open source contributors. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering (FSE '10)*. ACM, New York, NY, USA, 127-136.

Endres, M. L., Endres, S. P., Chowdhury, S. K. and Intakhab Alam 2007. Tacit knowledge sharing, self-efficacy theory, and application to the Open Source community. In *Journal of Knowledge Management*, Vol. 11 Iss: 3, pp.92 - 103.

GNU, 2013. O que é Software Livre? < http://www.gnu.org/philosophy/free-sw.html> Accessed in 14/06/2013.

Hintikka, K. A. 2008. Web 2.0 and the collective intelligence. In *Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era (MindTrek '08)*. ACM, New York, NY, USA, 163-166.

Hwang, Y. C., Yuan, S. T. and Weng, J. H., 2009. A study of the impacts of positive/negative feedback on collective wisdom– case study on social bookmarking sites. In *Journal Information Systems Frontiers, Springer*, Volume 13, Issue 2 , pp 265-279.

Khanjani, A. and Sulaiman, R. 2011. The process of quality assurance under open source software development, Computers & Informatics (ISCI). In *IEEE Symposium on*, vol., no., pp.548,552, 20-23 March 2011.

Lethbridge, T.C., Singer, J. and Forward, A. 2003. How software engineers use documentation: the state of the practice. *In Software, IEEE*, vol.20, no.6, pp.35,39.

Maia, J. R. C. 2003. Garantia a Qualidade de Projeto Orientado a Objeto. Project Management Institute. Santa Catarina.<http://www.euax.com.br/system/attachments/4/original/2006.013Metricas_software.pdf?1265047553> Accessed in 10/09/2011.

Malone, T. W., Laubacher, R. and Dellarocas, C. 2009. Harnessing Crowd: Mapping the Genome of Collective Intelligence. Working Paper no. 2009-001, MIT Center for Collective Intelligence.

Michlmayr, M., Hunt, F., Probert, D. 2005. Quality Practives and Problems in Free Software Projects. In *Proceedings of the 1st International Conference on Open Source Systems*. Genova, Italy, 24-28. < http://oss2005.case.unibz.it/Papers/47.pdf:> Accessed in 08/06/2013.

Noll, J. 2008. Requirements Acquisition in Open Source Development: Firefox 2.0 In IFIP International Federation for Information Processing, Volume 275; Open Source Development, Communities and Quality; Barbara Russo, Ernesto Damiani, Scott Hissam, Björn Lundell, Giancarlo Succi; (Boston: Springer), pp. 69–79.

Noll, J. and Liu, W. 2010. Requirements elicitation in open source software development: a case study. In *Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open source Software Research and Development (FLOSS '10)*. ACM, New York, NY, USA, 35-40.

OSI, 2013. The Open Source Definition. <http://opensource.org/docs/osd> Accessed in 14/06/2013.

Parnin, C. and Treude, C. 2011. Measuring API documentation on the web. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software*

Engineering (Web2SE '11). ACM, New York, NY, USA, 25-30.

Parnin, C., Treude, C. Grammel, L. and Storey, M. 2005. Crowd Documentation: Exploring the Coverage and the Dynamics of API Discussions on Stack Overflow <http://www.cc.gatech.edu/~vector/papers/CrowdDoc-GIT-CS-12-05.pdf> Accessed in 15/05/2013.

Porruvecchio, G., Uras, S. and Concas, G. 2010. Knowledge management aspects in open source communities. In *Proceedings of the 9th WSEAS international conference on Telecommunications and informatics (TELE-INFO'10)*, V. Niola, J. Quartieri, F. Neri, A. A. Caballero, F. Rivas-Echeverria, and N. Mastorakis (Eds.). World Scientific and Engineering Academy and Society, Stevens Point, Wisconsin, USA, 52-60.

Pressman, R. S., 2000. Software Engineering – A Practitioner's Approach, 5º ed. *McGraw-Hill International*, London.

Raymond, E. S. 1999. The Cathedral and the Bazaar. Sebastopol, CA: O'Reilly & Associates.

Rus, I. and Lindvall, M. 2002. Knowledge management in software engineering, Software. *In IEEE*, vol.19, no.3, pp.26,38.

Scacchi, W. 2002. Understanding the requirements for developing open source software systems. Software. In *IEE Proceedings* - , vol.149, no.1, pp.24,39.

Shaikh, S. A. and Ceron, A. 2007. Towards a quality model for Open source Software (OSS). < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.1973&rep=rep1&type=pdf.> Accessed in 01/06/2013.

Sommerville, 2007. "Engenharia de Software São Paulo". Pearson Adisson-Wesley. Brazil, 7th edition.

Spinellis, D. and Szyperski, C. 2004. How is open source affecting software development?. *In Software, IEEE*, vol.21, no.1, pp.28,33. DOI: 10.1109/ MS.2004.1259204.

Tosi, D. and Tahir, A. 2013. A Survey on How well-know Open Source Software Projects are Tested. In Communications in Computer and Information Science. *Springer*, Volume 170, 42-57.

Treude, C. and Storey, M. 2011. Effective communication of software development knowledge through community portals. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering (ESEC/FSE '11)*. ACM, New York, NY, USA, 91-101.

Treude, C., Barzilay, O. and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web? (NIER track). In *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. ACM, New York, NY, USA, 804-807.

Zhao, L. and Elbaum, S. 2000. A survey on quality related activities in open source. *In SIGSOFT Softw. Eng. Notes 25*, 3 (May 2000), 54-57.