

SELECTION PROCESS OF OPEN SOURCE SOFTWARE COMPONENT

Abstract : In 1985 Free software Foundation(FSF) was developed with the software being termed as free however some individuals felt that the term free software would not go well with the corporate world and as such proposed the term “Open Software”.This saw the formation of Open Source initiative(OSI) in 1998 which had an intend to enable software developers to be able to publish their software with an open source license, to allow others to access the source code and also develop further the same software[1]. The two movements in the software development age disagree on basic principles but do agree on the practical recommendations of the software. FSF meant software that gives one freedom to do what they want and when it comes to OSI it is “software compliant with one of the licenses of Free Software Foundation or Open Source Initiative.[2, 3]. The two terms OSS or FOSS(Free OSS)are used interchangeably but i will focus on OSS and in this paper i will discuss the various methods that are used in the evaluation and selection of an OSS component. To be able to tackle this i have structured the discussion as such that i discuss an in depth definition of the two terms OS and FS, OSS component, evaluation framework and the selection process and finally my conclusion.

1 Introduction

1.1 Open Source Software

1.1.1 OSS/FS definition

To be able to tackle the definition of Open source software i talk about the two movements in its development. Free Software Foundation (FSF) started in 1985 is a social movement that promotes user of computer programs with rights to use, study, copy, modify, and redistribute these computer programs [2]. OSI on the other hand promotes open source software and certifies open source license as well as advocating the benefits of open source software. In contrast, free software movement emphasizes the freedom of use. Although the two terms have been defined differently in practice, people treat them the same and use Free/Open Software (FOSS), Free/Libre/Open Source Software (FLOSS) or open source software/free software (OSS/FS) as the general term.

FSF defines free software as software with four freedoms,Free software” is a matter of liberty, not price [2]. These freedoms include:

- the freedom to run the program, for any purpose (freedom 0) to any type of computer system;
- the freedom to study how the program works, and change it to make it do what you wish (freedom 1). Access to the source code is a precondition to this.
- the freedom to redistribute copies so you can help your neighbor (freedom 2)
- the freedom to distribute copies of your modified versions to others (freedom 3). By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this

We see that the software freedom is good and according to Wheeler “OSS/FS programs are programs whose licenses permit users the freedom to run the program for any purpose, to study and modify the program, and to freely redistribute copies of the original or modified program”[4]. This brings us to the OSI definition where we have to think of not only the freedom granted with the free software but also the distribution

terms of the software[6]. OSI definition lists the following as the requirements that the distribution must conform to:

1. Free Redistribution. There is no restriction in the license from selling or giving away the software as a component of an aggregate software.
2. The software should include source code and must allow distribution in source code as well as compiled form.
3. Derived works. License should allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
4. Maintain integrity of the author's source code.
5. No discrimination against persons or groups.
6. No discrimination against fields of endeavor. Should be free to be used in any field.
7. Any users whom the software is distributed to have all the rights defined in the license.
8. License must not be specific to a product.
9. License must not restrict other software.
10. License must be technology-neutral.

As such open source software is free software as it gives the freedom required and can be differentiated with the proprietary software which is restrictive in its use and one can not modify it. For such kind of modifications, redistribution or usage of proprietary software one requires permission to be granted. Some distinction with freeware and shareware with OSS is that freeware are software packages that can be used freely for unlimited time and redistributed but the source code is not made available. Shareware on the other hand allows redistribution of the software copy but it is not free if continued usage then the user has to pay license fee for it or discontinue usage, the source code is unavailable as well.[7]. From the two definitions of “free software” and “Open source” we see some differences but when it comes to the implementation and usage they agree on some practical recommendations and work hand in hand and complement each other.

1.2 OSS component

1.2.1 What is an OSS component.

A software component is a software technology for encapsulating software functionality according to some specification . As in Object oriented programming it is in the form of objects or collections of objects displayed in binary or textual form, adhering to some interface description language (IDL) thus ensuring its existence is independent of other components in a computer [9]. According to Szyperski and Messerschmitt they highlight the following characteristics that a software component should have to be termed one: [8][10]

- Multiple-use- re usability of the software component
- Non-context-specific
- Composable with other components
- Encapsulated (i.e., non-investigable through its interfaces)
- A unit of independent deployment and versioning

As such from Szyperski[10] definition : “A software component is an executable unit of independent production, acquisition, and deployment that can be composed into a functioning system”. When it comes to OSS component then it means that software component that is easily accessible and is free. The software component source code can be accessed and reused over and over independently of the other components in the computer.Components can be built in-house, acquired from third-parties, or just downloaded from the websites of the OSS communities. OSS components always have the source code open while COTS(Commercial Off The Shelf) components may have the source code open but mostly it is not the case;both of these components are acquired from third parties and thus users have no control of the provided functionality and evolution of them [11].

When OSS component is discussed it reflects the Component-Based Software Development (CBSD) which is used alot in developing software systems nowadays it involves usage of reusable parts to develop software.Successful reuse of these components highly depends on being able to search through the OSS marketplace to identify and evaluate which component fits the requirements of the organization. The main reason of CBSD is the ability to reuse software components which in turn allows easy adoption of technology and innovation in organizations[16]. The other reasons that drive component based software development include:[18]

- Reduction in the time needed to develop and maintain systems as a result of reusing existing components this allows the development and maintenance of

computer systems to follow the real needs of a quickly evolving business application.

- Improved interoperability, user friendliness and quality through the reuse of functions and interfaces.
- Controlled and improved quality.

1.2.2 OSS Component requirements

When it comes to selection of an OSS component the requirements that would influence their selection are similar to those of OSS program. Organizations chose to use OSS products for certain reasons; the main reasons for usage being cost, variety of available components, and the access to the source code hence a possibility of making modifications. These components could be used in multiple projects that are co-related in that share similar functionality and/or architecture. In that case certain project types are based on a set of ready made components and the final solution is built on top of that base. There can be variations on the number of such components used, however, from the organization point of view they can be regarded similarly as components used in Software Product Lines [12] [13].

As an organization decides to use an OSS component it is all dependent on the requirements that the organization would like to fulfill . These requirements are organization specific and range from functional, technical, economical or even political and as such these should be elaborated to be able to support selection of any component. Required features of a particular software product in use should be prioritized in order to see whether they are being met or not and to what extend can it be acceptable. These requirements would as well guide in the selection of an OSS component, each of them covers different features as explained below.[14]

1. **Functional** refers to the features that the software product has to cover and it all depends on the purpose the program or OSS component is intended for. The crucial functionality required for the product or component to work should be covered and there should also be a clear roadmap of any changes that would occur in future.
2. **Technical**: The environment to run the software product should be considered as it also affects the selection of the software components and libraries that would work with the software. The target platform to run the software should be considered that is the operating system, the libraries and virtual environment. Reliability issue as well as maintainability of the product and its required software components is part of the technical requirements.

3. **Organizational:** The community that develops OSS product and component should be in existence, have good reputation, able to offer sufficient technical support to the users of the product and the development process should be compatible with the organization's. Community existence is not all that matters if it does not show that it would exist for long so longevity is also important.
4. **Legal:** Different legal implications govern the different F/OSS licenses therefore it is important that a company is certain about its own expectations and thus guide whether to use OSS/FS. These legal requirements cover:
 - No Copyleft Effect for Add-ons or Combinations: The license does not oblige the company to license own add-ons under the same license as the F/OSS.
 - No liability to third party code: Given that the company selling the product did not write parts of the code, it may not want to be liable for the foreign code parts. This requirement is country specific as some do impose these law yet others dont.
 - No patent infringements: Usage of the F/OSS in question does not collide with any patent restrictions
5. **Economical:** The organization financial constraints should be put into consideration when selecting any product whether OSS or proprietary software. Economic requirement covers issues of Sustainability of the product, Protection of investment, Increase of productivity, Flexible maintenance according to individual needs, Quick availability, Increasing know-how, Cost reduction and Division of development costs.
6. **Political:** It covers aspects such as Possibility for influencing further development with respect to individual needs, Decrease of proprietary dependencies, Transparency over safety and security and Publicity and marketing effects. These influence the selection process as an organization cannot forget about the IT world and how fast its changing and thus the products have to be aligned to fit.

The above requirements are evaluated according to some criteria and whether they are fulfilled then the organization would decide to select a particular product or component accordingly. Once several software component has been identified then they have to be evaluated to see that it really does fit for the particular project or software development. The evaluation should not only focus on the requirements explained but also on the risk associated with the software components and to be able to evaluate the risk the quality attributes should be examined.[15]. OSS component for commercial projects should be evaluated in the confines of the context of the project, this context is normally a combination of many factors including project risks and actual role the component will play in the solution to be used. Every project is unique having its own internal

and external factors that affect its risk tolerance so component selection is project dependent. The selection aims to try and eliminate spending too much time on the evaluation of the OSS component and to reduce risks associated with the given project.

The project could be risky in nature and the role of the component to be used in the final product is insignificant this will involve less risk in component selection. On the other hand the component's role could be very crucial for the final product and the project less risky. In both cases the project contexts differ and this should be included in the evaluation criteria of the OSS components. The different project context would be:

- low risk context -Simple project with high tolerance for possibly malfunctioning component;
- high risk context -Project on tight budget, mission critical, high demand for performance and reliability. Component is core element or its reliability and ease of integration is very important.
- Ordinary context -Typical project with all components expected to work correctly and their integration to the project should not pose problems; or the component is one of the many important components in the product.

Being on the information and digital age most organization engage in selecting a particular software component through use of research on the internet and we see that this has become an OSS market place but at times the information is not enough and thus selecting an OSS component becomes challenging and risky if not carefully evaluated. OSS market place [17] is a virtual self organised site on the internet that allows the interactions between reusers and providers of OSS components as well as the actions of other actors that facilitate or promote such transactions. *Providers* offer OSS components through their own websites. *Reusers* use a search mechanism or Intermediary services to find and select components, whilst *Promoters* foster the OSS movement. Selecting the correct open source module is not an easy task it's difficult to search through all the libraries, identify and extract the best modules, and correctly estimate the customization and integration cost but mostly people tend to use Google. Successful reuse of OSS components will depend on being able to go through OSS marketplace to identify and evaluate which component fit the requirements which are organization specific.

2 OSS evaluation

2.1 OSS Program/ Component evaluation framework

There exists several frameworks that have been used in the evaluation of OSS component. These include Qualification and Selection of Open Source software (QSOS), Open Source Maturity Model (OSMM) of Navica and CapGemini and the Business Readiness Rating (BRR). All of them however do allow quantitative evaluation of number of criteria based on their assigned weights and thus help to easily compare results of evaluation when similar pieces of software are evaluated for one particular use. An overview of these methods is explained below:

“**QSOS** is a method, designed to qualify, select and compare free and open source software in an objective, traceable and argued way. It is publicly available under the terms of the GNU Free Documentation License”[20][24]. The QSOS process is made up of four interdependent steps:

1. Definition-creation of frames of reference used in the following steps.
2. Evaluation- Based on : i) functional coverage; ii) risks for the user; and iii) risks for the service provider independent of any particular user or customer context.
3. Qualification-weighting of the criteria split up on the three axes and modeling the context, user requirements, and/or strategy set by the service provider.
4. Selection-process the data provided in steps one and two through the filter set up in step three in order to proceed to queries, comparisons, and selections of products.

The software is selected according to the weightings, representing the user’s requirements specified in step three. This model helps in selecting the best open source solution in a given context.

OSMM developed by Navica [23] help organizations in determining the maturity of OSS components. The product’s maturity is assessed on criteria such as product

software, support, documentation, training, integration and professional services. It uses 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. To help with facilitating the evaluation process OSMM provides a set of document templates for use, these are product requirements template, product software template and professional support template.

CapGemini Open Source Maturity Model[22] works in a similar way as the one by Navica calculating the maturity level it does involve the interaction between consultants and the customers of the open source software. 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. It also has a feedback phase where the effectiveness of the utilized indicators could be evaluated and proposes that the continuous evaluation of indicators would provide the required calibration within a changing environment – particularly important for F/OSS.

Business Readiness Rating (BRR) [21] is a recent initiative that aims 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. BRR is comprised of four phases: Quick Assessment Filter, Target Usage Assessment, Data Collection & Processing and Data Translation with rating ranging from 1, “Unacceptable”, to 5, “Excellent”. 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). Normally BRR [25] proposes twelve criteria that could be used to evaluate software once an initial shortlist of products has been established however it suggests usage of six or seven most important criteria in the assessment. The evaluator decides the most important ones for the software to be successful in the environment to be used and for the purpose it is to fulfill, weighting the criteria accordingly. Initial analysis suggests that this effort provides more detailed evaluation data and scoring than the previously mentioned models.

Another model in use is **Karin van den Berg's** Open Source Evaluation Model [26]. As described in the master thesis he uses important aspects of open source software: community, release activity, longevity, license, support, documentation, security, functionality, integration, modularity, standards, collaboration with other software and software requirements to be able to evaluate the software. In the model he proposes four criteria to be used in the selection of the candidate these are functionality, longevity, release activity and the community. In the evaluation process two processes are followed

the selection method called Elimination by Aspects model and the second one is the Linear Weighted Attribute Model. Elimination by aspects tries to eliminate products that do not conform to the minimum requirements according to the functionality and longevity criteria. The longevity criteria states that the last stable release should not be not older than 2years while functionality states that the product should be able to meet the required functionality to work. After the elimination the results gotten are ranked and weighted using the Linear Weighted Attribute model.

QualiPSo Open Source Maturity Model aims to help in building trust in development processes of companies using or producing FLOSS products focusing on the quality aspects. [<http://www.qualipso.org/>]

Although all the models use slightly different methods and techniques, they all attempt to evaluate the quality or maturity of F/OSS artifacts using information that is already available.

2.2 Phases of OSS Selection Process

Several methods of OSS evaluation have been discussed by David Wheeler [27] but he suggests use of four steps in evaluation which he describes them as general process. This process is based on four steps: identify candidates, read existing reviews, compare the leading programs' basic attributes to your needs, and analyze the top candidates in more depth. These steps can be followed when evaluating both OSS programs or proprietary programs. When using the wheeler method it implies that the organization already know what product they would like to use to fulfill their basic needs. The steps are explained as follows:

2.2.1 Identify candidates

The process proposes to first look at lists of OSS programs, lists of “generally recognized as mature” or “generally recognized as safe” OSS programs. To be able to get these programs Wheeler recommends searching on specialized sites to track OSS/FS programs, these sites are:

- Enterprise open source directory: <http://www.eosdirectory.com/>- tracks key projects and hosts commentary on them
- Freshmeat: <http://freshmeat.net/> - Specialised list including ranking
- Icewalkers: <http://www.icewalkers.com/>- Maintains list that only tracks programs that run on Unix/Linux

- Free software directory: <http://directory.fsf.org/>
- Open source for windows: <http://osswin.sourceforge.net/>- tracks programs running on windows
- OpenDisk : <http://www.theopencd.org/>-Gives away a CD of respected OSS/FS programs
- Sourceforge and Savannah: <http://sourceforge.net/index.php> and <http://savannah.gnu.org/>-host or include many OSS/FS project
- Koders: <http://www.koders.com/> - Searching for particular reusable components
- Use of good search engines- Google, Teoma, Alltheweb, and AltaVista

Another way is to search using Google specialized searches for Linux and BSD(Unix) as well as through Linux distribution e.g debian(<http://www.debian.org/>) and Kruger(<http://www.krugle.com/>) which help in searching for source code.

2.2.2 Read existing reviews

It is much more efficient first to learn about a program's strengths and weaknesses from a few reviews than to try to get that information just from project websites. The easiest way is to use a search engine and search for an article containing the names of all the identified candidates in the process you can discover issues that were not known earlier. Apart from search engines like Google it is also useful to search web sites that cover that market or functional area and see if there are any published reviews. Market share is an indirect review of a product and must be considered because it is important. Products having large market share are likely to be sufficient for many needs and are easier to support and inter operate. Market share though is difficult to measure due to the usage patterns of FS/ OSS where users might just download and use the product without registering.

2.2.3 Briefly compare the leading programs' basic attributes to your needs First

It is necessary to read the project web site to get information about the project and the required functionalities of the product. The next step is to evaluate the project on a number of important attributes: functionality, cost, market share, support, maintenance, reliability, performance, scalability, usability, security, flexibility/customizability, interoperability, and legal/license issues. Through examining the different attributes benefits, drawbacks, and risks of using a program can be determined. [28]The Software

Release Practice HOWTO has guidance to developers on how to create a project web site while the Free Software Project Management HOWTO provides guidance to those who manage such projects. Apart from examining the important attributes pertaining a given software product, other issues like local policies or technical requirements should be considered.

2.2.4 Perform an in-depth analysis of the top candidates

The attributes mentioned in the previous step are investigated, but it is advisable to actually try things out instead of quick reading the available literature. For issues such as performance and scalability it is best to set up a representative situation dummy data and see how well the program performs. In all this analysis the OSS/FS product are tried on non critical situations. Always try to get the best in the market and consider the version numbers of the program as one version could be different from the other one with either added features or some discarded. Services of a professional software examiner can be employed to examine a program's design documentation, source code, and other related material.

In Wheeler[27] the selection process is not done in a linear progressive manner but in an iterative way until the best component that suits the requirements is met. Moreover the selection cannot be dependent on the functionality requirements only but also usability and cost issues should be addressed. Annick and Jean [29] propose use of an operational approach to selecting OSS component regarding quality as part of the things considered in the selection. The approach assigns appropriate responsibilities to obtain an appropriate balance of power between the client and the development team. In the model two quality teams are established client quality team and development quality team with both of them have a role to ensure that the FLOSS component selected is in accordance with the quality criteria selected. The context of the client and software development team must be taken into account. Due to growing trend of software component reuse and and component based software development careful selection of these components is important. In Claudia et al [17] they talk about 2key ways of evaluating an OSS component ; experience-based Evaluation where the developer or a colleague has previous experience with the component and recommends it and the other way is through Searching for information to Evaluate components mostly through use of community portal.

3 Conclusion

From the different evaluation frameworks discussed and the selection process it is evident that more research has to be carried out on the selection of OSS components as a very clear roadmap is not distinct. These models do not focus much on the evaluation of the code different aspects have to be understood to help researchers in improving the selection of OSS components. In addition it is evident that most of the time the requirements that are to be met revolve around functionality, legal ,technical and many more as discussed but the quality of the OSS component is not mentioned so much. Availability of source code is not enough to guarantee quality of software to users other than developers.

OSS components selection should be improved as the emergence of the OSS market place poses challenges that need to be tackled. The market place is rapidly growing with new components and technologies continuously offered. What is relevant and new today could be irrelevant tomorrow also there should be proper standards put in place for describing OSS. Besides personal experience in selection one should be careful so that the information is not misleading.[17].

Mostly open source components were typically being used where developers were seeking quick solution but this is not the case nowadays. Therefore care should be taken in selecting the appropriate component to eliminate having problems in the future which may result through not following the company's roadmap and the product that requires the component. The technology is changing rapidly so the environment for use of the component need to be explored fully and the existing component library. In comparison with when selecting proprietary software component the process is different as the resources are easily gotten in a sense that they are open to everyone and there is greater freedom and access to broad range of views.

Bibliography

- [1] Eric S. Raymond <http://www.catb.org/~esr/open-source.html> Goodbye, "free software"; hello, "open source". Accessed 8th October 2010
- [2] Free Software Foundation :<http://www.fsf.org> and . Accessed 8th October 2010
- [3] Open Source: <http://www.opensource.org>. Accessed 8th October 2010
- [4] Open source/Free software definition: http://www.dwheeler.com/oss_fs_refs.html. Accessed 8th october 2010.
- [5] S. Hissam, C. B. Weinstock, Daniel Plakosh and Jayatirtha Asundi, “Perspectives on Open Source Software, Software Engineering Institute”, technical report, CMU/SEI-2001-TR-019, Carnegie Mellon University , Nov 2001.
- [6] Open Source definition: <http://www.opensource.org/docs/osd>. Accessed 8th October 2010
- [7] Why “Free Software” is better than “Open Source: <http://www.gnu.org/philosophy/free-software-for-freedom.html>. Accessed 8th October 2010
- [8] Greg Olsen,Coghead, “From COM to Common”Component software’s 10-year journey toward ubiquity : <http://queue.acm.org/detail.cfm?id=1142043>. Accessed 12th October 2010.
- [9] Software component :http://en.wikipedia.org/wiki/Component-based_software_engineering. Accessed 12th October 2010.
- [10] Szyperski.C,Dominik.Gand Stephan Murer:Component Software – Beyond Object-Oriented Programming. Addison-Wesley, 2002.
- [11] Li, J. “Process Improvement and Risk Management in Off-The-Shelf Component Based Development” PhD theses at NTNU, 2006:84, pp 29. Accessed 12th October 2010.
- [12] J. Bosch, Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000.

- [13] K. Pohl, G. Böckle, and F. J. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005
- [14] D. Cruz, T. Wieland, and A. Ziegler, "Evaluation Criteria for Free/Open Source Software Products Based on Project Analysis", *Software Process: Improvement and Practice*, 11(2): 107-122, 2006
- [15] Rudzki.J,Kiviluoma.K, Poikonen.T and I.Hammouda, Evaluating Quality of Open Source Components for Reuse-Intensive Commercial Solutions ; *Software Engineering and Advanced Applications*, SEAA '09. 2009 35th Euromicro Conference on Software Engineering and Advanced Applications.
- [16] Mohagheghi P, Conradi R (2008) An Empirical Investigation of Software Reuse Benefits in a Large Telecom Product. *ACM Transactions of Software Engineering and Methodology*, Vol. 17, No. 3, Article 13, 30 pages.
- [17] Challenges of the Open Source Component Marketplace in the Industry Claudia Ayala, Øyvind Hauge, Reidar Conradi, Xavier Franch and Jingyue Li, et al. *IFIP Advances in Information and Communication Technology*, 2009, Volume 299, *Open Source Ecosystems: Diverse Communities Interacting*, Pages 213-224
- [18] Brereton, P., Linkman, S., Thomas, N., Bøegh, J., De Panfilis, S.: *Software Components - Enabling a Mass Market*. In: *STEP 2002: Proceedings of the 10th International Workshop on Software Technology and Engineering Practice*, pp. 169–176. IEEE Computer Society, Los Alamitos (2002)
- [19] D. A. Wheeler, "How to evaluate open source software / free software (oss/fs) programs," [http://www.dwheeler.com/oss fs eval.html](http://www.dwheeler.com/ossfs/eval.html), Accessed 16th October 2010
- [20] *Method for Qualification and Selection of Open Source software (QSOS) version 1.6* © Atos Origin (April 2006), <http://qsos.org/> Accessed 16th October 2010.
- [21] *Business Readiness Rating for Open Source* © OpenBRR.org, BRR 2005. <http://www.openbrr.org>. Accessed 16th October 2010.
- [22] http://pascal.case.unibz.it/retrieve/1097/GB_Expert_Letter_Open_Source_Maturity_Model Accessed 16th October 2010.
- [23] Navica, "Open Source Maturity Model," <http://www.navicasoft.com/pages/osmm.htm>, 2004. Accessed 16th October 2010.
- [24] QSOS : <http://www.osbr.ca/ojs/index.php/osbr/article/view/583/540>, May 2008. Accessed 16th October 2010
- [25] OSS Watch- Business Readiness Rating, <http://www.oss-watch.ac.uk/resources/brr.xml>. Accessed 16th October 2010.

- [26] K. v. d. Berg, “Finding Open options, An Open Source Software Evaluation Model with a Case Study on Course Management Systems”, Master thesis, Tilburg University, August 2005.
- [27] How to Evaluate Open Source Software / Free Software (OSS/FS) Programs David A. Wheeler dwheeler @ dwheeler.com, http://www.dwheeler.com/oss_fs_eval.html#identify. Accessed 17th October 2010
- [28] Software release practice: <http://www.tldp.org/HOWTO/Software-Release-Practice-HOWTO/> and Free software management : <http://www.tldp.org/HOWTO/Software-Proj-Mgmt-HOWTO/index.html> . Accessed 17th october 2010
- [29] Annick Majchrowski and Jean-Christophe Deprez (2008) “An Operational Approach for Selecting Open Source Components in a Software Development Project”,<http://www.springerlink.com/content/h88j529275876h64/fulltext.pdf>.