Implementing Open Source Software Governance in Real Software Assurance Processes

Claudio A. Ardagna[1], Massimo Banzi[2], Ernesto Damiani[1], and Fulvio Frati[1]

[1] Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano
via Bramante, 65 - 26013 Crema (CR) - Italy
[2] TelecomItalia TILab - Vertical Platforms & VAS - Vertical Platforms Innovation
via V. Zambra, 1 38100 Trento - Italy

Abstract. Open Source is giving rise to new methodologies, competences and processes that organizations have to investigate both from the technical and the managerial point of view. Many organizations are studying the possibility to adopt open source products or migrate their systems to open frameworks, even for mission-critical application. In this paper we discuss a roadmap for organizations that want to establish a formalized governance methodology for the management of the open source products, taking into consideration issues such as software quality and community reliability. The governance framework is designed to be included in a more complete software assurance system for open source software.

Keywords: Software Assurance, Open Source, Governance.

1 Introduction

The concept of *Software Assurance* (SwA) [4] involves a number of different phases of software process and varies depending on the specific context where it is applied. For instance, the implementation of SwA actions can be targeted to the reaching of specific software quality, security or dependability requirements.

Generally speaking, the term *assurance* has been associated to the task of reducing general information risks while improving, at the same time, the overall quality of data reported to management and decision makers. In fact, the assurance concept is associated to different tasks, such as risk assessment, information systems security, internal audit, and customer satisfaction surveys, focused on specific aspects of the environment they are applied to [1].

As far as software engineering is concerned, the assurance process includes all the activities that an organization deems necessary to provide a correct level of confidence that a software effectively will satisfy the functional and non-functional requirements requested by its users. As depicted in Fig. 1, assurance activities, in a traditional waterfall-based development process, are process-oriented. Each phase of software lifecycle is covered by specific assurance activities, starting from the *Requirements* and *Use Cases* definition, where specific tests analyze the fulfillment of security requirements and the presence of possible abuse cases [13], and examining, for instance, the *Code* phase, where specific tools are exploited to calculate the sets of product metrics that are able to assess code quality, dependability, performance, and the like.

Of course, in the case of cooperative development processes, like the ones used for Open Source (OS) development, is not possible to identify a complete and rigorous development process and, consequently, associate all assurance tasks showed in Fig. 1. In this scenario, the definition of assurance mechanisms for OS solutions is a difficult task.

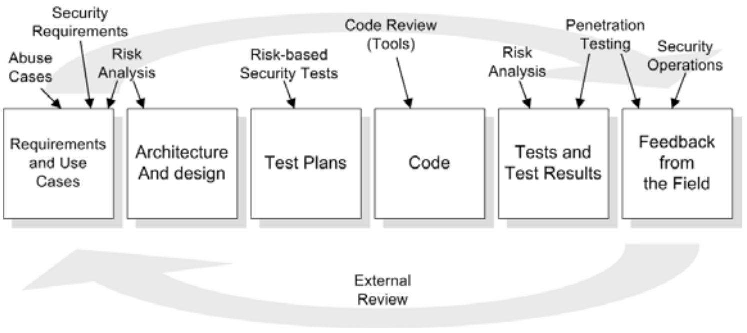

Fig. 1. Assurance tasks in a traditional, lifecycle-based development process

The problem of defining an assurance methodology for OS projects has been studied by Damiani *et al.* in [3], where the problem of establishing assurance policies and methodologies for OS software has been addressed, focusing on the specific security requirements of critical telco environments. In this paper we try to extend the concepts described in [3] focusing on issues related to OS software governance, providing a guide and a roadmap for managers and decision makers for driving them through the decision of adopting OS frameworks in their critical applications.

The paper is organized as follows. Section 2 gives an introduction of open source governance. Then, Section 3 describes OS governance methodology, introducing a new vision for the OS management, and Section 4 an outlook of the future work to implement the model. Finally, Section 5 gives our conclusions.

2 Open Source Governance

In the literature, the concept of "governance" has been mostly associated to the creation and the management of the community of users and developers that characterizes an OS product [12]. As for instance O'Mahony and Ferraro [14] define a model for OS community governance, while in [11] the author gives insights how the lessons learned from OS communities could be applied to other productive environments.

The concept of governance has to be extended in order to consider OS environments not only from the point of view of an inside-analyzer, but taking into consideration the needs of an organization or a group which wants to integrate OS products following precise and reliable adoption policies.

The common policies used in the case of adoption of commercial products could not be applied, since they are mostly focused on the purchase cost variable that could not be directly applied to OS products. Otherwise, such OS governance should take into account aspects that are typical of OS development, like community management, the number of developers, the quality of the available source code, the type of license, and so forth. If not managed, the “free” world of OS software can rapidly lead to inconsistent situations from both technical and licensing perspectives. Therefore, governance policies are becoming increasingly important as a means of ensuring the long-term viability and acceptability of OS projects inside and across enterprises.

The problem has been debated from many point of view, trying to give high-level recommendations on how to implement an existing governance framework to drive specific education and enforcement actions [7], or how to define a governance structure suited to the symbiotic relationship of actors involved in the OS process [10].

An enforceable program of OS governance is of paramount importance for gaining more value from OS products, protecting, at the same time, the interest and assets of organizations. In this paper we describe the governance model that an important telco player (Telecom Italia) studied and developed by exploiting researches on OS environments undertaken by the academic research group at Università degli Studi di Milano. The model is aimed at understanding the context in which the software will be adopted and having a full knowledge of the impact that the adoption will have in organization common activities. In fact, in the telco context the exploiting of OS can range from simple deployment of selected applications, to the embedding of OS packages in the development of critical systems. The model provides a set of metrics, recommendations, and procedures to analyze the OS software to be embedded in legacy solutions.

The implementation of a reliable governance framework will be integrated in the overall SwA strategy that could be implemented by organizations to raise their awareness and control of the adopted software packages.

3 Open Source Governance Model

The OS governance methodology implemented by TelecomItalia is composed of two distinct phases: *i)* the selection of the suitable OS tool and *ii)* the management of the OS environment.

3.1 Open Source Tools Selection Methodology

The need for TelecomItalia of having a model for OS governance comes from the results of an analysis of installed OS packages across all the company’s business areas. The results was reported in [3] and shows that, while OSS adoption was pervasive, it was carried out without any kind of adoption strategy: a number of different applications were installed to perform the same activity, resulting in multiple outcomes when performing the same action.

The typical business process for telco companies is based on eTOM model (enhanced Telecom Operations Map) [19], released by the TeleManagement Forum. The model includes a hierarchy of process definitions, offering a structure to represent and develop areas of the process. Furthermore, it is important in the management of licenses and Intellectual Properties Rights (IPRs). In fact, different process areas imply different levels of visibility and customization of the adopted OS package. This issue leads to a careful analysis of the license types and the area where they are going to be integrated, ensuring that the license of the adopted package is consistent with its specific license and business area. According to the specific service in which the OSS component is embedded, special care has to be taken in verifying the suitable licenses.

The methodology answers three specific questions: *i)* **Identify reliable communities**, identifying a set of parameters to be collected and evaluated periodically to rate the communities themselves, *ii)* **Verify the type of licenses used**, comparing them with the business area they are going to be included, and *iii)* **Verify the quality of the adopted solution**, evaluating it in terms of adherence to coding and security standards.

Community Reliability Metrics. The evaluation of the reliability of an OS community requires the definition and evaluation of a set of metrics that have to be applied to public data released by the communities themselves, or extracted analyzing forums, download logs, and releases frequency. Currently, there is any standard framework of metrics, but a number of independent projects (e.g., as described in [16]) provide their own metrics, each one focused on specific aspects.

Even though well-known portals like SourceForge, Freshmeat, and the OW2 consortium, release information about the projects they host and distribute, those data are semantically heterogeneous and do not allow cross-portal evaluations. For instance, the management of issue tracking system is handled very differently at different portals, and the definition of "closed" for an issue is subject to different interpretations. Also, the importance given to the number of downloads varies when the community considers more valuable the quality of a product rather than its diffusion.

Currently, the most-established assessment frameworks are the BRR Model [15], which provides a complete set of metrics to evaluate if a project is enough mature to be adopted but does not define rules on collecting data, and FLOSSmole [9], that collects and distributes data from several communities. However, these two sets of metrics are not homogeneous and are difficult to compare.

In order to fulfill the task of evaluating more the community aspect than the software itself, evaluating at the same time the reliability and the reactivity of the community to newly discovered bugs and vulnerabilities, we consider FOCSE (Framework for OS Critical Systems Evaluation) [5], a framework, developed by the Università degli Studi di Milano, based on a set of general purpose security-related metrics that perfectly fit the needs of the analysis. In fact, FOCSE includes some specific metrics expressing the capability of the

Table 1. First level metrics

Metric	Description
Date of Search	Date when current data were collected
First Release Date	Date of the first documented stable release of the product
Last Stable Release Date	Date of the last product stable release
#ofBugsDetected	No. bugs detected from bug reports
#ofBugsFixed	No. bugs fixed from bug reports
#ofForumReplies	No. community answers to forum messages
#ofForumThreads	No. groups of specific user questions
#ofReleases	No. releases since project start up
#ofPatches	No. patches since project start up
#ofDownloads	No. downloads since project start up
Project Core Group existence	Evaluate the existence of a group of core developers (1 if exist, 0 if do not exist)
Number of core Developers	No. core developers contributing the project
Forum and Mailing List Support	Check forum and mailing list availability (0 if no mail no forum, 1 if mail but no forum, 2 both mail and forum)
RSS Support Check	RSS availability (0 no, 1 yes)
Documentation fulfillment	1 to 5 check for the presence of the following type of documents: admin, user, config, release delta notes,white paper guides

community to adapt to continuous changes in security threats and vulnerabilities. It provides a set of metrics focused on multiple areas: *i)* Generic Application, *ii)* Developers Community, *iii)* Users Community, *iv)* Software Quality, *v)* Documentation and Interaction support, and *vi)* Integration and Adaptability with new and existing technologies.

The original formalization of FOCSE provides two different sets of metrics: the first one includes all metrics that can be readily computed from available information on the OSS projects, while the second one requires parameters whose measures involve privileged access to the developers' community. Such metrics has been improved and adapted for the particular needs of the Telco context where they are going to be applied to; Tab. 1 and Tab. 2 summarize the metrics, giving a short description of them.

To generate a single estimate, it is necessary to aggregate the value of metrics in Tab. 2. This way, two or more projects, each one described by its set of attributes, can be ranked. To this aim, an aggregation algorithm has to be

Table 2. Second level metrics

Metric	Description
Age	Age of the project calculated from first stable release to *Date of Search* **Date of Search - First Release Date**
Last Update Age	Age of the last stable project update to *Date of Search* **Date of Search - Last Stable Release Date**
Project Core Group	Evaluate the existence of a group of core developers [**1 exist 0 do not exist**]
Number of core developers	Number of core developers contributing the project
Replies/Threads (Community Vitality)	Vitality of the community in terms of no. answers in response to specific user questions **(#ofForumReplies/#ofForumThreads)**
Number of Releases	Number of releases since project start up
Update Average Time (time/release)	Vitality of developers group,i.e. mean number of days to wait for a new update **Age/(#ofPatches+#ofReleases)**
Average of downloads per release	**#ofDownloads/#ofRelease rate**
Bug Fixing Rate	Rate of bugs fixed, weighted over the total number of bugs detected **(#ofBugsFixed+1/#ofBugsDetected+1)* (1-exp(-#ofBugsDetected))**
Problems per release	**(#ofBugsDetected - #ofBugsFixed) /#ofReleases**
Mean Time Between Failures (MTBF)	**(#ofBugsDetected - #ofBugsFixed)/Age**
Forum and Mailing List Support	Check forum and mailing list availability (0 no mail no forum, 1 mail no forum, 2 mail e forum)
RSS Support Check	RSS availability [**1 exists, 0 do not exist**]
Documentation Fulfillment	Check for the presence of the following type of documents: admin, user, config, release delta notes, white paper guides [**1..5**]

provided to compute a weighted sum of all metrics value that will return an indicator to establish if a product is acceptable or not.

A solution of this problem has been found in [5] using the *Ordered Weighted Average* (OWA) operator. Introduced by Ronald Yager in [20], the OWA operator provides a parameterized aggregation method, characterized by a reordering step that makes it a nonlinear operator. The OWA operator is different from the classical weighted average in that coefficients are not associated directly with a particular attribute but rather to an ordered position. The operator is defined as follows:[1]

Definition 1. *Let* $w = [\omega_1, \omega_2, \ldots, \omega_n]$ *a weight vector of dimension* n*, such that* $\omega_1 \in [0, 1]$ *and* $\sum_i \omega_i = 1$*. A mapping* $f_{OWA} : R^n \rightarrow R$ *is an OWA operator of dimension* n *if*

$$f_{OWA}(a_1, a_2, \ldots, a_n) = \textstyle\sum_i \omega_i a_\sigma(i)$$

where $\{\sigma(1), \sigma(2), \ldots, \sigma(n)\}$ *is a permutation of* $\{1, \ldots, n\}$ *such that* $a_\sigma(i) \leq a_\sigma(i-1)$ *for* $i = 2, \ldots, n$

All decision processes that involve multiple selection criteria (i.e. metrics), like OS selection methodology described in the article, require some compensatory trade-offs. They occurs in the presence of conflicting goals, when compensation between the corresponding compatibilities is allowed. The OWA operator can realize trade-offs between objectives, by allowing a positive compensation between ratings, i.e. a higher degree of satisfaction of one of the listed metric will compensate for a lower degree of satisfaction of other criteria to a certain extent.

The metrics framework has been tested on a number of open source products to show its potential and effectiveness. Tab. 3 compare three open source SSH clients using FOSLET metrics: PuTTy, WinSCP, and ClusterSSH. Finally, Tab. 4 shows the aggregated values for the three applications, calculated using the OWA operator. A precise and suitable calibration of OWA weights, and the application of the aggregation operator, will lead to the creation of ranking of the examined OS products, allowing to choose for the best one with respect to organization policy.

License Type Evaluation. The intersection between the requirements of each business area defined in the eTOM model (i.e. visibility of the services to common users, customizations of existing code), and the specific characteristics of the different types of available OS licenses, make their analysis an important issue of the overall governance framework.

The definition of a standard procedure for license analysis starts from the review of the literature in terms of OS licensing. The great success of OS paradigm has lead to the proliferation of a great number of OS license types. Every license shares the principle that everyone should be able to use, copy, modify, and distribute the code, but each one has particular ways to manage such actions.

[1] A more complete formalization of the OWA operator is out of the scope of this paper. A more detailed explanation could be found in [5][20].

Table 3. Comparison of proposed SSH implementations [5]

Metrics	**Putty**	**WinSCP**	**ClusterSSH**
Age	2911 days	1470 days	1582 days
Last Update Age	636 days	238 days	159 days
Project Core Group	Yes	Yes	Yes
Number of Core Developers	4	2	2
Number of Releases	15	32	15
Bug Fixing Rate	0.67	N/A	0.85
Update Average Time	194 days	46 days	105 days
Forum and Mailing List Support	N/A	Forum Only	Yes
RSS Support Check	No	Yes	Yes
Reply/Threads(Community Vitality)	N/A	3.73	5.72

Table 4. OWA-based SSH applications comparison [5]

	Putty	**WinSCP**	**ClusterSSH**
f_{OWA}	0.23	0.51	0.47

Several works categorize OS license in term of Intellectual Property Rights [17] and try to track the transformation of the OS licensing with respect to the continuous changes of OS environment [8]. A complete listing of available license has been published by the Open Source Initiative (OSI, www.opensource.org) and by the Free Software Foundation (FSF, www.fsf.org), reporting more than one hundred license types that differs in some parts and that could affect the adoption and use in production environments.

Discussing how to manage all these license types is outside the scope of our work. Instead, we categorize licenses in three overall categories. An understanding of these licensing categories lays the groundwork over which the approach of making decisions about OS licensing is based. In our three-category model, licenses are categorized basing on the extent to which it requires a derivative work to share same license in the produced code [18].

Category A: Unrestricted Licenses. The licenses in that category are wholly unrestricted with regard to the scope of license use in derivative works. A developer operating under a Cat. A licenses may therefore use the commons to create virtually any work, in any way, and then use any kind of licensing he or she desires for the derivative work. The developer may, for example, use a free or OS license for the derivative work, or even use a commercial closed license for the work. Examples are the BSD-style licenses.

Category B: File-based Community-fostering Licenses. Like Cat. A, includes licenses that allow developers to use the open code to create virtually any work, in any way. They assume, however, that these derivative works will be made up of source files that will ultimately become a single binary file, and that any source file that contains such code must be licensed with the same license. They do not, however, impose this requirement on files that do not contain code from the original code base; those files can be licensed in any way the developer wishes. Examples are Mozilla-style licenses and the Sun's Common Development and Distribution License.

Category C: Strong Copyleft Licenses. Like Cat. A and B, it contains licenses that allow developers to use the open code in any way. Like Cat. B licenses, they require any file that contains derived code derived to be licensed under the original license. Furthermore, they also require that any file, regardless of code origin, which is combined under certain circumstances with the common files must be licensed under the original license. Examples are the GNU General Public License.

This classification has been used to create a check list table to help decision makers on deciding how to behave in front of the different tools according to their usage within an organization. Together with such categorization, that will help to restrict the number of open tools we can include in our products, a deeper check could be executed exploiting free products, like the Koders tool (www.koders.com), that will examine the product code base ensuring that it is developed free of concerns raised by OS licenses, known security vulnerabilities or corporate policies regarding OS code usage.

Software Quality. The quality of OS software is expected to be good enough. Anyway, some quality metrics, taken by literature, are applied to deeply evaluate it checking its intrinsic quality, the accordance to coding standards, and the solution of known security issues. In particular, software quality is evaluated using the well-established metrics for Coupling, Cohesion, Cyclomatic complexity, and the Change Risk Analysis and Prediction (CRAP) metrics. At the same time, the accordance to specific coding standards is assessed using free verifier tools specific of the programming language used for the implementation, such as the J2EE Verifier Tool (java.sun.com/j2ee/avk/), the High Integrity C++ Conformity (www.codingstandard.com), or the MISRA-C and MISRA-C++ tools (www.misra-cpp.org).

Finally, security is taken into account. Security aspects are usually handled by OS developers by exploiting community feedbacks and issues tracking environments. The need of some form of security certification based on a rigorous and in-depth analysis has been raised, and the formalization of a certification framework that will allow, on the one side, suppliers to certify the security properties of their software and, on the other side, users to evaluate the level of suitability of different OS security solutions, is strongly required [1].

3.2 Open Source Environment Management

The final, and may be the most critical, aspect of the OS governance is the definition of the correct way the adoption has to be managed. The investment in OS is also an investment in the IT department, since the adoption of established, good-quality products will increase the internal skill level and know-how.

Starting from this point, it is possible to operate in five distinct ways, differentiated by the approach of the organization versus the OS environment.

Train people on the product. Developers are trained about the product, giving them the knowledge for critical bug-fixing or possible customizations.

Train people to community work. Developers are trained to work in community, with a win-win approach, taking into consideration also the needs and requirements of other members.

Ignite the community with developers. In case the OS product is considered important enough to be able to operate on the product, it can be wise to exploit internal developers to contribute to the community either for small enhancements or for bug-fixing. This allows to acquire competence on the code, but understanding the behaviour and feelings of the community, on how the community operates and on the timing it can react to requests.

Join the community board. If the organization wants to have the highest Return of Investment, once ensured that problems with the licenses and internal skills are overcome, the choice will be of operating directly on the OS community board in implementing the necessary functionality, driving it, and defining future tasks and functionalities. The result is approaching the OS as a resource, where developers are either internal and external, with savings not just on the maintenance of the component, but also in its development.

Create a community. If a suitable community does not exist, it can result profitable to share and open an internally developed solution to extend the test and code base. Such an approach has been followed by TelecomItalia for WADE, where a community has been created and driven to extend the OS platform JADE [6]. A Support Group, that clearly owns the major skills on the solution and will represent the core developers group, and a Technical Manager, responsible for deciding on the roadmap of the product within the arising community, have to be identified.

To protect the investment, where possible, the organization has to negotiate privileges with the community, such as implementing certain features in a protected development environment, where the commit control is delegated to the organization itself. Furthermore, a major involvement of organization in the communities core group, will ensure a major stability of project requirements and achievements.

Such a vision, applied to the last two scenarios, paves the way for a new perspective within the OS paradigm. The strict collaboration with OS communities gives to commercial organization the opportunity of consistent savings since they can enter and work in a working environment complete with all the communication, bug tracking, and versioning tools typical of OS projects. In addition to that, the possibility to negotiate privileges with the board of the community, gives the opportunity to work in a protected context.

For instance, organizations can ask to the board to open a new development branch exploring solutions that are of interest for them; at the end of the implementation, such code will be released, as usual, under the adopted OS license, but, at the same time, organizations will see their investment in OS protected and profitable.

4 Future Work

At present, a major problem of OS adoption is the complete lack of standards that force to normalize available data on the quality/confidence of the communities. This often results in a huge analysis effort that, due to the continuous evolution of OS communities, has to be periodically repeated. The adoption rate of OS components by large companies suffers from the lack of standards. Furthermore, the FOCSE framework, which has been used as the basis for the metrics framework, has to be improved to include more sophisticated and community-based concepts, like for instance the alignment between project strategy and development objectives.

TelecomItalia, is trying to create consensus on some sort of standards - the Open Source Governance Model as described in Section 3 - by operating within TeleManagement Forum: the world's leading industry association focused on improving business effectiveness for service providers and their suppliers. The idea is to export the work performed in collaboration with Università degli Studi di Milano and integrate it with the experiences of the largest telco service providers in the world to define *de-facto* standards to be proposed to interested OS service supplier companies and indirectly to communities.

5 Conclusions

In that paper we analyzed the state of the art in OS governance, highlighting known issues and proposing a new model for OS governance that could be included in a overall OS software assurance strategy. Our model takes into consideration the evaluation of communities reliability, the type of used license, and the quality of the released software. Furthermore, we give a schema for OS environment management, introducing a new vision for the OS paradigm that considers commercial organizations as active elements of the OS communities, reserving to them some sort of privileges that can protect their investment.

Acknowledgment. This work was partly founded by the European Commission under the project SecureSCM (contract n. FP7-213531) and by the Italian Ministry of Research under FIRB TEKNE (contract n. RBNE05FKZ2_004).

References

1. Ardagna, C.A., Damiani, E., El Ioini, N.: Open Source Systems Security Certification. Springer, Berlin (2008)
2. Ardagna, C.A., Damiani, E., Frati, F., Madravio, M.: Open Source solution to secure e-government services. In: Encyclopedia of Digital Government. Idea Group Inc., USA (2006)

3. Ardagna, C.A., Banzi, M., Damiani, E., Frati, F., El Ioini, N.: An Assurance Model for OSS Adoption in Next-Generation Telco Environments. In: Proc. of 3rd IEEE Int. Conf. on Digital Ecosystems and Technologies (DEST 2009), pp. 619–624. IEEE Press, New York (2009)
4. Evans, M.W., Marciniak, J.J.: Software quality assurance & management. Wiley-Interscience, New York (1987)
5. Ardagna, C.A., Damiani, E., Frati, F.: FOCSE: An OWA-based Evaluation Framework for OS Adoption in Critical Environments. In: Open Source Development, Adoption and Innovation, pp. 3–16. Springer, Boston (2007)
6. Banzi, M., Bruno, G., Caire, G.: To What Extent Does It Pay to Approach Open Source Software for a Big Telco Player? Open Source Development. In: Communities and Quality, pp. 307–315. Springer, Boston (2008)
7. Best Practices in Open source Governance (2010), `https://fossbazaar.org/content/best-practices-open-source-governance`
8. Fitzgerald, B.: The transformation of Open Source Code. Management Information Systems Quarterly 30(3) (2006)
9. FLOSSmole: Collaborative collection and analysis of Free/Libre/open Source Project Data (2010), `http://flossmole.org`
10. Franck, E., Jungwirth, C.: Reconciling Investors and Donators - The Governance Structure of Open Source. Lehrstuhl für Unternehmensführung und politik Universität Zürich, Working Paper No. 8 (2002)
11. De Laat, P.B.: Governance of Open Source Software: State of the Art. J. Manage Governance 11, 165–177 (2007)
12. Lattemann, S., Stieglitz, C.: Framework for Governance in Open Source Communities. In: Proc. of the 38th Hawaii IEEE International Conference on System Sciences. IEEE Press, New York (2005)
13. McDermott, J., Fox, C.: Using Abuse Case Models for Security Requirements Analysis. In: Proc. of the 15th Annual Computer Security Applications Conference (ACSAC 1999), Washington, DC, USA (1999)
14. O'Mahony, S., Ferraro, F.: The Emergence of Governance in an Open Source Community. Academy of Management Journal 50(5), 1079–1106 (2007)
15. OpenBRR: A Framework for Evaluating Open Source Software (2010), `http://www.openbrr.org`
16. Qualipso Project: Trust and Quality in Open Source systems (2010), `http://www.qualipso.org`
17. Rosen, L.: Open Source Licensing. Prentice-Hall, Englewood Cliffs (2005)
18. Sun Whitepapers: Free and Open Source Licensing (2010), `http://mediacast.sun.com/users/sunmink/media/sunlicensingwhitepaper042006.pdf`
19. TeleManagement Forum: Business Process Framework (eTOM) (2010), `http://www.tmforum.org/browse.aspx?catID=1647`
20. Yager, R.R.: On ordered weighted averaging aggregation operators in multi-criteria decision making. IEEE Transaction Systems, Man, Cybernetics 18(1), 183–190 (1988)