A FLOSS License-selection Methodology for Cloud Computing Projects

Robert Viseur
CETIC, Rue des Frères Wright, 29/3, 6041 Charleroi, Belgium
UMONS, Faculty of Engineering, Rue de Houdain, 9, 7000 Mons, Belgium

Keywords: Cloud Computing, Saas, Paas, Iaas, PaaSage, Governance, License, FLOSS, Open Source.

Abstract: Cloud computing and open source are two disruptive innovations. Both deeply modify the way the computer resources are made available and monetized. They evolve between competition (e.g. open source software for desktop versus SaaS applications) and complementarity (e.g. cloud solutions based on open source components or cloud applications published under open source license). PaaSage is an open source integrated platform to support both design and deployment of cloud applications. The PaaSage consortium decided to publish the source code as open source. It needed a process for the open source license selection. Open source licensing scheme born before the development of cloud computing and evolved with the creation of new open source licenses suitable for SaaS applications. The license is a part of project governance and strongly influences the life of the project. In the context of the PaaSage European project, the issue of the open source license selection for cloud computing software has been addressed. The first section of the paper describes the state of the art about open source licenses including the known issues, a generic license-selection scheme and the automated source code analysis practices. The second section studies the common choices of licenses in cloud computing projects. The third section proposes a FLOSS license-selection process for cloud computing project following five steps: (1) inventoring software components, (2) selecting open source license, (3) approving license selection (vote), (4) spreading practical details and (5) monitoring source code. The fourth section describes the PaaSage use case. The last section consists in a discussion of the results.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

Licenses are thought to be selected at the beginning of the project with no posterior change (Fogel, 2005). They give the rules of the collaboration which everybody agrees to if participating in the project. To some extent, they provide a sort of "constitution" or legal agreement of how the project is developed and distributed. The selection of a license is particularly important in the context of projects involving many partners. The terms of free and open source licenses are widely considered as a part of the open source governance mechanisms (Markus, 2007).

Cloud computing and open source are two disruptive innovations (Marston *et al.*, 2011; Onetti and Capobianco, 2005; Ven *et al.*, 2007). Both deeply modify the way the computer resources (applications, storage,…) are made available and monetized. They evolve between competition (e.g. open source software for desktop versus SaaS applications) and complementarity (e.g. cloud solutions based on open source components, open source software demonstrating open standards for cloud computing, pooling of development resources between industrial actors through open source projects or cloud applications published under open source license by editors).

PaaSage is "*an open source integrated platform to support both design and deployment of Cloud applications, together with an accompanying methodology that allows model-based development, configuration, optimisation, and deployment of existing and new applications independently of the existing underlying Cloud infrastructures*" (paasage.org). The PaaSage is an ongoing project. It received research funding from the European Union's 7th Framework Programme.

A FLOSS license-selection methodology for cloud computing projects was created and applied to the selection of the FLOSS license for the PaaSage European project.

Viseur, R.
A FLOSS License-selection Methodology for Cloud Computing Projects.
In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 1*, pages 129-136
ISBN: 978-989-758-182-3
Copyright © 2016 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

The paper is organized in five sections. The first section describes the state of the art about open source licenses, including known issues, a generic license-selection method and automatic source code analysis practices. The second section studies the common choices of license in cloud computing projects. The third section proposes a FLOSS license-selection process for cloud computing project. The fourth section describes the PaaSage use case. The fifth section consists in a discussion of the results and a presentation of the perspectives.

2 FLOSS LICENSES

2.1 Families of FLOSS Licenses

Software licenses can be roughly divided in three categories: the open source licenses, involving the source code sharing, the proprietary licenses, limiting the rights to the profit of software editors, and the hybrid licenses, providing protection for intellectual property but wide access to source code (de Laat, 2005; Muselli, 2007; West, 2003). Free and open source projects (also said FOSS or FLOSS) are covered by specific licenses that warrant the free and open source nature of software as defined by the Free Software Foundation (fsf.org) and the Open Source Initiative (opensource.org). The "free software" definition is based on four freedoms (use, study, modify and redistribute), while the "open source" definition is based on 10 criteria including the access to source code, the free redistribution of source code and the allowance to create derived works

The free and open source licenses can be divided in two main families (Cool *et al.*, 2005; de Laat, 2005; Honkasalo, 2009; St. Laurent, 2004; Viseur, 2013b). The first one includes the permissive licenses (also said : academic licenses). The second one includes the copyleft licenses (also said : reciprocal or restrictive licenses). The permissive licenses allow the use of the source code in proprietary software. The Apache, BSD and MIT licenses are famous examples. Contrariwise, copyleft licenses impose limitations on licensees of any derivative work, such as the conservation of the license or the availability of the source code when the software is made available. Their principle could be resumed by the sentence "*copyleft one day, copyleft always*". The AGPL, CDDL, CPL/EPL, GPL, LGPL, MPL and OSL licenses are famous examples of copyleft licenses.

The use of licenses evolves during time. Thus the study of Github repositories shows a rise of permissive licenses and the publication of source codes without license. The publication without license is problematic because it doesn't allow to know what are the rights and duties for the downloaded source codes, even if the source code is public (it stays covered by copyright). A move towards more permissive licenses can be also observed among some FLOSS editors (e.g. Talend or Alfresco) (Viseur and Robles, 2015).

2.2 Sub-families of Reciprocal Licenses

The reciprocal licenses family can be divided in three sub-families: the licenses with weak reciprocity, the licenses with strong reciprocity and the licenses with network reciprocity (Cool *et al.*, 2005; de Laat, 2005; Honkasalo, 2009; St. Laurent, 2004; Viseur, 2013b).

Table 1: Major Open Source Licenses by Type.

	Permi-ssive	Weak recipro-city (file-based)	Weak recipro-city	Strong recipro-city	Net-work recipro-city
BSD	x				
MIT	x				
Apache	x				
MPL		x			
EPL, CPL		x			
CDDL		x			
LGPL			x		
GPL				x	
AGPL					x
OSL					x

The licenses with strong reciprocity spread to all derivative works. Their detractors describe them as "*viral*" or "*contaminant*". The licenses with weak reciprocity only apply to the original work but don't automatically spread to derivative work. Those licenses can be file-based or not. The file-based licenses with weak reciprocity simplify the addition of functionalities under different compatible licenses. The licenses with network reciprocity are designed for hosted software. They are close to the licenses with strong reciprocity, but they force to

transfer the source code as soon as the user is in touch with the user interface of the application. The Table 1 resumes a set of popular licenses that were classified by category.

2.3 Issues around the License Selection

The choice of an open source license is an important step in the life of an open source project. It is not neutral on the life of the project. Moreover it conditions the way the software may be monetized by companies. The license impacts the business model, the compatibility between open source components, the easiness to change license and adapt to strategic changes as well as the project success.

2.3.1 Impact on Business Model

The business models are closely related to the selected licenses, because they allow to regulate the regime of appropriability (Viseur, 2012b).

Unlike the permissive licenses, the licenses with strong copyleft forbid the creation of a proprietary version of the software. However there is a common exception. The dual licensing schema often associates a version of the software under strong copyleft license (usually: GPL) and a commercial version (available for a fee). The commercial version may offer additional features. That schema is usually legal because the open source editor organizes the sharing of the ownership of contributed source codes, for example by using contributor agreement (Poo-Caamaño and German, 2015) or by accepting contributions only under a permissive license. The full ownership is possible by rewriting each contribution (Valimaki, 2003).

2.3.2 Impact of License Changes

Viseur and Robles (2015) studied the case of 24 open source projects that were impacted by a license change and identified problems caused by that change.

The main difficulty comes from the shared property of the source code. In consequence, the license change obliges to get an agreement from all the contributors. This process is problematic for large projects with a lot of contributors. An alternative consists in owning the copyright on the source code, by using contributor agreement or by rewriting each contribution (Poo-Caamaño and German, 2015; Valimaki, 2003). The first solution burdens the contribution process and may discourage the developers to participate. The second solution results in a lost of time.

Moreover the license changes may lead to some problems. The new license may suffer from incompatibilities (i.e. incompatibilty between licenses or undesirable side effect such as license propagation) with previously linked projects (e.g. MySQL library vs PHP). For example, according to the Free Software Foundation, the popular GPL v2 license is not compatible with GPL v3 and Apache v2. Thus a license change implies to check the compatibility of the new license with the components whose the software depends. Secondly, the license change may irritate the community and lead to a fork. Forking is a mechanism of splitting in a community that usually results in the cohabitation of two competing projects. Fifteen percent of the forks would be a consequence of the reaction to a license change (Viseur, 2012a).

It results that license changes should be avoided as far as possible. However it may be required by the evolution of the environnement (e.g evolution of license use, license change of subcomponents or maket change).

2.3.3 Impact on Project Success

The license is highlighted in literature as a success factor for open source project. However the impact is always discussed (Viseur, 2013b). There is a trend to attribute a negative impact for copyleft or restrictive licenses.

In practice, the negative impact of restrictive licenses would depend on the status of the project (Midha and Palvia, 2012). If we consider the criterion of market success, the negative impact of restrictive licenses only takes place for the first version of the software. It tends to disappear with time. If we consider the criterion of technical success, the negative impact of restrictive licenses does not occur in the early stages of the project but in the following stages. That finding would be explained by the fact that the license is one of the only pieces of information available to users when the software appears and that the first developers see the restrictive licenses as a protection against the risks of privative ownership.

2.4 Methodologies for Choosing Open Source Licenses

We rely on the generic methodology (license selection based on the valuation schema) presented

in Viseur (2013a). That valuation scheme is divided into three steps.

Step 1: choosing the type of license (proprietary, hybrid, open source).
Step 2: choosing an open source license (if an open source license is chosen in step 1).
Step 3: checking general constraints (e.g. compatibility issues or organization policy).

As we discuss the selection for open source license, we are mainly interested in the step 2. This second step is divided into four sub-steps.

Sub-step 2.1: It checks the interest or the willingness to join an existing community or ecosystem. If yes, the license of the ecosystem is retained (e.g. Apache license for Apache Foundation or Eclipse license for Eclipse Foundation).

Sub-step 2.2: It tests if the priority is given to the maximum distribution of the software. If yes, a permissive license is chosen (e.g. MIT or BSD).

Sub-step 2.3: The priority is given to the sharing of developments. This sub-step verifies if the license should facilitate the integration of software into third-party software that are potentially under other licenses.

Sub-step 2.3.1: If the integration with third-party software is not preferred, the willingness to cover the use of the software as SaaS is checked. If yes, a license with network reciprocity is chosen (e.g. AGPL or OSL). Otherwise, a license with strong reciprocity is chosen (e.g. GPL).

Sub-step 2.3.2: If the integration with third-party software is preferred, the degree of permissiveness accepted in case of new features addition is checked. In case of high permissivity, a license with based-file weak copyleft is used (e.g. MPL). Otherwise, the LGPL is used.

Sub-step 2.4: It verifies the willingness to share the source code ownership. If yes, a contributor agreement is imposed to the contributors.

Once the open source license is selected, the compliance with organization policy is checked (e.g. black / white list of open source licenses, patent policy or compatibility with other open source licenses used in previous projects). In case of incompatibility, the method is applied again; a new license is selected or the source code is covered by multiple licenses (if the source code is covered by multiple open source licenses, the user can select the license that is suitable to his context of use).

2.5 Beyond License Selection: Legal Analysis of Source Code

Several tools allow to analyze the source code in order to simplify the detection of legal problems (e.g. unexpected licenses or incompatibility between licenses).

Black Duck Software is a famous proprietary software that includes license analyze features. Fortunately some open source software exist. The open source code analysis tools are usually based on the analysis of the source code files headers and on the open source licenses footprints. In consequence, they are not able to detect files under proprietary license or duplicated source codes.

The more famous open source tool is probably FOSSology. It is supported by Hewlett Packard (www.fossology.org). Its main drawback is probably the slowness. An alternative software is edited by Ohloh (now Open Hub after Black Duck bought the service) and called Ohcount (www.ohloh.net/p/ohcount). The Ohcount main function is source code line counter. However, the "*-l, --license*" option allows to display the detected licensing information contained in each source code file. A basic formatting process allows to create reports that simplify the detection of problems. The process can be supplemented by focused searches (for example, with Find and Grep open source tools) in order to detect issues with patents or copyrights notices. An example of implementation is described in (Viseur, 2012b).

Those tools allow to process an automated analyze of the source code hosted in repositories in order to anticipate the issues (mainly: wrong copyright notices and unexpected licenses headers). Alerts could be directly and automatically reported from Git directory (git-scm.com).

3 COMMON CHOICES IN OPEN SOURCE CLOUD PROJECTS

3.1 Typology of Cloud Computing

From the point of view of a user, cloud computing comes in three distinct service models (Marston *et al.*, 2011; Mell and Grance, 2011). The Software as a Service model (SaaS) provides the user with an application hosted in the cloud (e.g. Google Mail or

Google Documents). The Platform as a Service (PaaS) model provides an environment to develop and deploy applications (e.g. Microsoft Azure or Google App Engine). The Infrastructure as a Service (IaaS) model provides storage and computation capacities (e.g. Amazon S3 or Amazon EC2). These models can be deployed in a corporate network or an external platform. The first case is known as private cloud, the second, as public cloud.

3.2 Open Source SaaS Projects

The open source offers numerous alternatives to SaaS products (e.g. Owncloud, Cozycloud or Odoo). The providers may build services upon open components under license with weak or strong reciprocity and keep a product advantage by comparison to competitors. Indeed the use of open source software as SaaS doesn't oblige the provider to furnish the source code as there is no software distribution ("*convey*").

The FLOSS editors' behavior facing to SaaS was studied by Viseur (2012c) in the field of ebusiness software. The paper reveals the rise of licenses with network effect (e.g. AGPL and OSL). Indeed several of the studied projects (e.g. Magento, OpenERP and SugarCRM) adopted such licenses at their creation or later evolved towards that type of license. That trend has been observed since 2008 and is not without consequences for editors who were sometimes obliged to negotiate license change with community.

3.3 Open Source PaaS and IaaS Projects

The situation for IaaS and PaaS open source software is less well-known. We selected a set of open source cloud projects and detailed some characteristics, including the license and the responsible (private company, foundation, self-organized community,...). The projects are famous open source IaaS or PaaS software. They are: Openstack, Eucalyptus, OpenNebula, Cloudstack, Deltacloud, Openshift, Appascale, Stratos and Tsuru.

The open source cloud computing projects landscape is characterized by the weight of initiatives hosted by Apache Foundation (www.apache.org). We also point to the use of Apache license by other organizations such as Appscale Systems (www.appscale.com) or Red Hat (www.redhat.com).

Table 2: Characteristics of open source cloud projects.

Name	Description	License	Responsible
OpenStack	IaaS (described as cloud operating system).	Apache 2 license.	By OpenStack Foundation. Strong HP engagement.
Eucalyptus	IaaS.	Various with GNU GPL (+ commercial release).	By Eucalyptus company.
Open-Nebula	IaaS (described as management tool for virtualized datacenters).	Apache 2 license.	Founded by European project, now supported by C12G Labs.
Cloud-Stack	IaaS (management of large networks of virtual machines).	Apache 2 license.	By Apache Software Foundation.
DeltaCloud	Application Programming Interface (API) that abstracts the differences between cloud computing implemen-tations.	Apache 2 license.	By Apache Software Foundation.
OpenShift	PaaS.	Apache 2 license.	By Red Hat.
AppScale	PaaS (open source implementation of Google App Engine).	Apache 2 license.	By AppScale Systems.
Stratos	PaaS (framework based on Apache Tomcat, PHP and MySQL.	Apache 2 license.	By Apache Software Foundation.
Tsuru	PaaS.	BSD 3-clauses license.	By Globo.com company.

As a consequence, the Apache license is often use in IaaS and PaaS projects (see Table 2). The Apache license is a permissive one and offer latitude to choose business model. Moreover it obliges to document intellectual property issues (e.g. copyrights, trademarks, patents or other licenses covering sub-components) in NOTICE file (that practice can be reassuring in business context).

4 FLOSS LICENSE-SELECTION PROCESS FOR CLOUD COMPUTING PROJECTS

We suggest a FLOSS license-selection process in five steps.

Inventoring Software Components – The first step consists in an inventory of the components that have to be reused in the new development. That inventory allows to check the open source definition conformity and the compliance of the licenses attached to the components. It also allows to detect copyrights issues and conflicts with common valuation practices among the partners (e.g. TTO in the universities).

Selecting Open Source License – The second step consists in the license selection. It relies on the process that was explained in the section "*Methodologies for choosing open source licenses*". The result should take into account the common practices in cloud computing industry.

Approving License Selection – The third step consists in an approval process. The license that was selected at the second step is subject to a vote.

Spreading Practical Details – The fifth step consists in spreading the practical details among the partners and the developers, in particular for labeling the license in the source code of the software.

Monitoring Source Code – The sixth step consists in monitoring each release of the software in order to detect violation to the license policy.

5 USE CASE: PASSAGE PROJECT

PaaSage would like to "*deliver a development and deployment platform, with an accompanying methodology, with which developers of enterprise systems can access services of cloud platforms in a technology neutral approach that abstracts the technical details while guiding them to configure their applications for best performance*" (www.paasage.eu).

The PaaSage project has its own specific features. Firstly, the PaaSage project receives research funding from the European Union's 7th Framework Programme. In consequence, it must respect some dissemination obligations. Open source appears as a suitable solution to offer common technological foundations and disseminate results in industry. Secondly, the project gathers 19 partners (i.e. cloud technology providers, application developers, researchers and technology transfer experts). In consequence, the partners are heterogeneous, with possible differences between intellectual property policies among the different partners. For example, the strong copyleft was quickly considered as inappropriate for most of companies in cloud industry. Thirdly, some partners brought existing software components. In consequence, some open source or proprietary licenses were already used and must be taken into account in the view to integrate the software components in a common open source software package.

The five steps license-selection methodology was applied as follows.

1) Step 1: Inventoring Software Components.

An inventory of the existing component was processed by appealing to partners and validated during a meeting. It highlights the use of some open source licenses and a copyright issue with one of the partner (privative source code).

2) Step 2: Selecting Open Source License.

The PaaSage members wished to favour the creation of source code commons but encourage industry and avoid the problems of virality that cause incompatibility problems between software components. Considering those requirements we suggested to choose a license with weak reciprocity. Although it is widely used in open source cloud project, the permissive Apache license didn't satisfy the constraints expressed for PaaSage project, event if Apache license is widely used in open source PaaS / IaaS industrial ecosystem.

If we restrict the choice of licences at the list of recommended licenses that is published by Open Source Initiative at http://opensource.org/licenses we had to select between 4 licenses: the LGPL (2.1 or 3), the CPL/EPL, the CDDL and the MPL (1.1 or 2.0).

The CPL/EPL and CDDL were eliminated because they is incompatible with widely used GPL licence. The LGPL 3.0 license is stronger in terms of mutualization and responsibility to contribute to the development. However some companies could be afraid by the "GNU" label. The MPL 2.0 license is easier for the creation of combined works that contain files with various licenses. The PaaSage project gathers several partners and needs the integration of different components. The MPL license was designed to simplify the aggregation of open source code and third party components under various licenses. Moreover, the MPL license allows a wide variety of business models, in particular dual licensing model. Indeed it offers the possibility to add extensions under proprietary license and commercialize derivative software with technological differenciation. In consequence the

MPL 2.0 license was suggested. No contributor agreement was required.

3) Step 3: Approving License Selection.

A vote for MPL 2.0 license approval was processed. The following question was asked : "*does your organisation approve the adoption of the MPL2.0 licence as the common licence for PaaSage [Y/N] ?*". MPL 2.0 has been adopted unanimously as the common license for PaaSage.

4) Step 4: Spreading Practical Details.

The chosen open source license must be referenced in the source code. The developers have to indicate the license in the source code of the software, i.e.:

- Put the text of the license in the root of the source code.
  Typically, the original text is written in a file that is named LICENSE or LICENSE.txt. The text of the open source license can be found on www.opensource.org or on the websites of organizations that published the license.
- Put the short description of the license in the header of each file.
  The websites of Open Source Initiative and other organizations that publish the licenses suggest standard headers for each license. Tools allow to automate the process of header addition (e.g. customized bash scripts or Copyright Wizard plugin for Eclipse).

The MPL is well documented. In particular, the FAQ give examples of reusable headers and copyright notice (refer to http://www.mozilla.org/MPL/2.0/FAQ.html).

5) Step 4: Monitoring Source Code.

We plan to use tools allowing to detect lacks of conformity with the PaaSage license policy in the source code stored in the official Git repository.

Thanks to the PaaSage project, we discovered that Apache offered fast and reliable tool for license analysis: Apache Rat. "*Apache Rat is a release audit tool, focused on licenses*" (http://creadur.apache.org/rat/). The hosting as Apache Software Foundation (ASF) incubated project is a sign of the growing sustainability of the project (the Apache incubator provides an ASF entry point for new projects and allows the development of emerging communities). Moreover, Apache Rat may work as a plugin of Maven (http://mojo.codehaus.org/rat-maven-plugin/). "*Apache Maven is a software project management and comprehension tool*" (maven.apache.org). Maven and Git are used for PaaSage project. So it would be possible to use Apache Rat with Maven in order to generate alerts in case of abnormal license footprints. In addition Maven should allow to get a view of the structure of dependencies and to more efficiently filter those alerts.

6 DISCUSSION AND PERSPECTIVES

Results – PaaSage is an open source integrated platform to support both design and deployment of cloud applications. The PaaSage consortium decided to publish the source code as open source. It needed a process for the open source license selection. Indeed open source licensing scheme was born before the development of cloud computing (the applications were usually installed on local desktops or servers) and evolved with the creation of new licenses suitable for SaaS applications. The study results in a FLOSS license-selection process in five steps: (1) inventoring software components, (2) selecting open source license, (3) approving license selection (vote), (4) spreading practical details and (5) monitoring source code.

Limitations – We point to the following limitations to our approach. Firstly, our simplified license-selection methodology doesn't distinguish the different versions of a license. However the successive versions of a license can strongly differ and can even be incompatible (e.g. GPL 2 and GPL 3). The selection method must be used with the hypothesis the last version of the license is used. Secondly, the open source code analysis tools suffer from some limitations. Thus they doesn't detect "copy and paste" behaviors. Indeed they are based solely on open source license footprints and are able to detect rightly added open source source code files. In addition, they don't detect proprietary source codes or proprieray artefacts in source code (e.g. trademarks or proprietary source codes). Additional processing is needed to address those issues.

Improvements – The upstream work could still be improved by the use of design tools (e.g. OSSLI) allowing to detect legal issues (e.g. unwanted license side-effects or incompatibility issues) as soon as the software design step. That precaution would allow to anticipate problems as early as possible and raise developers awareness by doing. Moreover it would complement the use of source code monitoring tools for detecting a posteriori errors.

ACKNOWLEDGEMENTS

The work presented in this paper has been partially funded by the EU FP7-ICT project PaaSage (grant no. 317715).

REFERENCES

Cool Y., De Patoul F., De Roy D., Haouideg H., Laurent P., Montero E. (2005), *"Les logiciels libres face au droit"*, Cahier du CRID, n°25, Bruylant.

de Laat, P.B. (2005), *"Copyright or copyleft ? An analysis of property regimes for software development"*, Research Policy, vol. 34, pp. 1511-1532, 2005.

Fogel, K. (2005), *Producing open source software: How to run a successful free software project*, O'Reilly Media, Inc.

Honkasalo, P. (2009), *"Reciprocity under the GNU General Public Licenses"*, Nordic Journal of Commercial Law, no 1.

Markus, M. L. (2007), *"The governance of free/open source software projects: monolithic, multidimensional, or configurational?"*. Journal of Management & Governance, 11(2), pp. 151-163.

Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A. (2011), *"Cloud computing—The business perspective"*. Decision Support Systems, 51(1), pp. 176-189.

Mell, P., Grance, T. (2011), "*The NIST Definition of Cloud Computing"*, US Nat'l Inst. of Science and Technology; http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

Midha, V., Palvia, P. (2012), *"Factors affecting the success of Open Source Software"*. Journal of Systems and Software, 85(4), pp. 895-905.

Muselli, L. (2007), *"Les licences informatiques : un outil de modulation du régime d'appropriabilité dans les stratégies d'ouverture. Une interprétation de la licence SCSL de Sun Microsystems."*, 12ème Conférence de l'Association Information et Management, Lausanne, juin 2007.

Onetti, A., Capobianco, F. (2005), *"Open source and business model innovation. the funambol case"*. In Proceedings of the first International Conference on Open source Systems.

Poo-Caamaño, G., German, D. M. (2015), *"The Right to a Contribution: An Exploratory Survey on How Organizations Address It"*. In Open Source Systems: Adoption and Impact (pp. 157-167). Springer International Publishing.

St. Laurent, A.M. (2004), *"Understanding Open Source and Free Software Licensing"*, O'Reilly Media, 2004.

Valimaki, M. (2003), *"Dual licensing in open source software industry"*. Systemes dInformation et Management, 2003, vol. 8, no 1, pp. 63-75.

Ven, K., Verelst, J., Mannaert, H. (2007), *"On the Relationship between Commoditization and Open Source Software"*, 12e conférence de l'Association Information et Management (AIM), Lausanne (Suisse), 18-19 juin 2007.

Viseur, R. (2012a), *"Forks impacts and motivations in free and open source projects"*, International Journal of Advanced Computer Science and Applications (IJACSA), Volume 3 Issue 2 February 2012; http://dx.doi.org/10.14569/IJACSA.2012.030221.

Viseur R. (2012b), *"Gérer la propriété intellectuelle dans les projets à base de logiciels libres"*, 17ème conférence de l'Association Information et Management (AIM), Bordeaux (France), 21-23 mai 2012.

Viseur, R. (2012c), *"Evolution des stratégies et modèles d'affaires des éditeurs Open Source face au Cloud computing,"* in Terminal : Technologie de l'Information, Culture, Société, n°113-114, 2013, pp. 173-193.

Viseur, R. (2013a), *"Élaboration d'un schéma de valorisation pour l'édition de logiciels open source »,* 18ème conférence de l'Association Information et Management (AIM), Lyon (France), 22-24 mai 2012.

Viseur, R. (2013b), *"Identifying Success Factors for the Mozilla Project"*, Proceedings of the Ninth International Conference on Open Source Systems (OSS 2013), Capodistria (Slovenia), June 25-28, 2013.

Viseur, R., Robles, G. (2015), *"First Results About Motivation and Impact of License Changes in Open Source Projects"*. In Open Source Systems: Adoption and Impact (pp. 137-145). Springer International Publishing.

West, J. (2003), *"How open is open enough?: Melding proprietary and open source platform strategies"*. Research policy, 32(7), pp. 1259-1285.