10-2008

LICENSE COMPLIANCE ISSUES IN FREE AND OPEN SOURCE SOFTWARE

G.R. Gangadharan
Telematica Institute, Enschede, The Netherlands, gr@telin.nl

Stefano De Paoli
National University of Ireland, Maynooth, Ireland, Stefano.DePaoli@nuim.ie

Vincenzo D'Andrea
University of Trento, Trento, Italy, dandrea@disi.unitn.it

Michael Weiss
Carleton University, Ottawa, Canada, weiss@sce.carleton.ca

LICENSE COMPLIANCE ISSUES IN FREE AND OPEN SOURCE SOFTWARE

Gangadharan, G.R., Telematica Institute, Enschede, The Netherlands, gr@telin.nl

De Paoli, Stefano, National University of Ireland, Maynooth, Ireland, Stefano.DePaoli@nuim.ie

D'Andrea, Vincenzo, University of Trento, Trento, Italy, dandrea@disi.unitn.it

Weiss, Michael, Carleton University, Ottawa, Canada, weiss@sce.carleton.ca

Abstract

Today, Free and open source software (FOSS) is widely used by organizations and individuals and viewed as a new approach to developing software. New software can be developed by integrating FOSS components or incorporating source code fragments, thus adding value in terms of functionality and quality. The use of FOSS components in developing new software requires developers to comply with the terms of the licenses associated with those components. The issues related to this compliance scenario are of paramount importance, because the license of a FOSS component can impact the whole Information System or computer application being developed.

License compliance in FOSS is a significant issue today and organizations using FOSS are predominately focusing on this issue. The non-compliance to licenses in FOSS systems leads to the loss of reputation and the high costs of litigation for organizations. An automated approach is preferred to verifying license compliance of an FOSS being developed. Towards an automated approach, in this paper, we will argue for FOSS licenses in a machine interpretable form and for managing license compliance in a FOSS development process.

Keywords: Compliance, Free and Open Source Software, Software License.

1 INTRODUCTION

Compliance is referred as a state of being in accordance with certain established guidelines and/or legislation and/or internal policies. As an example for a set of guidelines, Control Objectives for Information and related Technology[1] (COBIT) is provided by the Information Systems Audit and Control Association (ISACA), and the IT Governance Institute (ITGI) for developing appropriate IT governance and control in an organization. If an organization practices these guidelines, according to the claims of COBIT, the organization can maximize the benefits derived through the use of information technology. As an example for internal policies, an organization can have a set of policies for supply chain management. The organization is expected to comply with these internal policies when purchasing materials. An example for legislation to which organizations (only for the United States of America) are required to be complaint is the Sarbanes-Oxley Act of 2002[2]. The regulations set forth in the Sarbanes-Oxley Act cover the issues of auditor independence, internal control assessment, and enhanced financial disclosure. All U.S. public company boards, management, and public accounting firms are required to be complaint with the Sarbanes-Oxley Act, failure of which leads to legal punishments. Providing a framework to automatically enforce the compliance to such guidelines, policies or legislations would be an ideal solution. However, this is highly difficult due to the ``rich texture'' of legal documents and guidelines.

Nowadays, free and open source software (FOSS) is widely used by organizations and individuals and viewed as a new approach to development of software (Rufn and Ebert, 2004). New software can be developed by integrating FOSS components or incorporating source code fragments, thus providing value addition in functionality and quality of the software. The use of FOSS components in developing new software requires to comply with the terms of the licenses associated with those components. Compliance to FOSS licenses can be complex due to the following reasons.

- The licenses vary in privileges and restrictions imposed for a licensor to follow in order to use, modify or redistribute the FOSS.
- The license clauses can be unacceptable to users or can be incompatible with other licenses.
- Opting different licenses during different phases of development of a software project can be quite confusing.

Today, organizations have adopted certain best practices to ensure the compliance. Managing compliance to licenses is essential to prevent the inadvertent dilution of authors' rights in FOSS development. An automated approach is preferred to verify the license compliance of a FOSS being developed. This requires FOSS licenses to be represented in a machine interpretable form.

The rest of the paper is organized as follows. Section 2 conceptualizes various compliance issues in FOSS development. In Section 3, we present some of the real scenarios of compliance issues faced in FOSS projects. Section 4 analyzes the best practices for compliance in FOSS development today, highlighting the insufficiencies of those approaches. In section 5, we describe a formalization of FOSS license clauses and present an automated way of managing compliance, followed by conclusions in Section 6.

2 LICENSE COMPLIANCE IN FOSS

In general, in a FOSS system, license conflicts arise in the following scenarios.

Conflicts by unacceptable license clauses. A license may contain certain clauses that are unacceptable to a software author. The cause for unacceptance is simply individual choice.

Following is a clause on distribution from the Lucent Public License Version 1.0.

---

[1] http://www.isaca.org/

[2] United States Pub. L. No. 107-204, 116 Stat. 745

While this license is intended to facilitate the commercial use of the Program, **the Distributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for Contributors**. *Therefore, if a Distributor includes the Program in a commercial product offering, such* **Distributor (``Commercial Distributor") hereby agrees to defend and indemnify** *every Contributor (``Indemnified Contributor") against any losses, damages and costs (collectively ``Losses") arising from claims, lawsuits and other legal actions ... with its distribution of the Program in a commercial product offering.*

The said terms require the distributor to defend each contributor. If a distributor does not wish to follow this clause, the license becomes unacceptable for her.

Conflicts by incompatible license clauses. Certain clauses of a license can directly prohibit integration of a FOSS component distributed with certain other license.

For example, the Apache License Version 2.0 is not compatible with the GPL Version 2.0 due to the following patent clauses on the Apache license.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted.

However, the Apache License Version 2.0 is considered as a free software license (based on the definition of the FSF) and is compatible with the GPL V3 (by the clause 11).

A contributor's ``essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, ``control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Conflicts by change of licenses over releases. The software organization should be careful in selecting a license for releasing a particular version of the software because the license of a particular release can have direct impact on future releases.

Consider a FOSS component S_A released under the GNU General Public License (GPL) license. At some point in the future, the licensor may decide to release a new version S_B under two different licenses say, GNU GPL[3] and Affero GPL[4]. However, the Affero GPL is incompatible with GNU GPL version 2 because of section 2(d) that covers the distribution of application programs via web services or computer networks.

If the Program as you received it is intended to interact with users through a computer network and if, in the version you received, any user interacting with the Program was given the opportunity to request transmission to that user of the Program's complete source code, you must not remove that facility from your modified version of the Program or work based on the Program, and must offer an equivalent opportunity for all users interacting with your Program through a computer network to request immediate transmission by HTTP of the complete source code of your modified version or other derivative work.

Thus, the release of S_B under Affero GPL conflicts with the license of the previous version S_A. However, the GPL version 3 and the GNU AGPL version 3[5] are compatible.

---

[3] http://www.gnu.org/copyleft/gpl.html
[4] http://www.affero.org/oagpl.html
[5] http://www.fsf.org/licensing/licenses/agpl-3.0.html

3 CASE LAWS ON LICENSE COMPLIANCE IN FOSS

In recent years, an increasing number of law suits have been filed involving several issues of compliances in FOSS. Following are some of the landmark cases in these areas.

BusyBox Versus Monsoon Multimedia Inc.

BusyBox[6] is a de facto standard for embedded Linux devices and Linux distribution installers, distributed under the terms of GPL. Monsoon Multimedia Inc.[7], developed and commercially distributed a firmware that contained the source code of BusyBox. Monsoon failed to refer the terms of GPL and hided the presence of BusyBox in the firmware. The case was settled with the release of the source code of firmware and with a levy of an undisclosed amount of penalty for the overuse of BusyBox by Monsoon.

Netfilter/iptables Versus Sitecom.

Sitecom inter alia (a German subsidiary of the Sitecom Europe B.V. [8], The Netherlands) developed and distributed a firmware for a specific kind of wireless network broadband router through its website. This firmware contained the Linux kernel including the software 'netfilter/iptables'[9] which is an open source project distributed under the GPL. Sitecom did not release the firmware under the GPL and did not mention that the firmware contained the GPL licensed software. Also, Sitecom neither mentioned the reference of the GPL nor to the source code of `netfilter/iptables'. A Munich District Court opined that the distribution of the firmware by Sitecom without complying the conditions of the GPL constitutes an infringement of copyright[10].

Fortinet Versus GPL.

Fortinet[11], a network security software firm, released an application FortiOS, an operating system as a part of some of Fortinet's products. The binary code of FortiOS contained the source code of some of the GPL licensed code including parts of the Linux kernel, in an encrypted way. However, Fortinet has not made the source code and license text available when distributing the code as is required by the GPL. Following an injunction from a Munich District Court, Fortinet released the source code of FortiOS under the terms of GPL.

The SCO Group Versus Linux.

The SCO Group[12] currently has a lot of disputes with various Linux vendors and users. SCO initiated a series of lawsuits and claims that Linux was an unauthorized derivative of UNIX produced by SCO. Furthermore, the claims of SCO states that Linux infringes upon their copyrights. The SCO Group also claimed the ownership of System V Release 4.0 (SVR4) Unix copyrights. In a case of IBM versus SCO, the case is seemed to be in favor of SCO. The judgement states as follows[13]: "SCO has not offered any competent evidence to create a disputed fact regarding whether IBM has infringed SCO's alleged copyrights through IBM's Linux activities." However, in the SCO versus Novell case[14], the court clearly wrote that Novell is the owner of the UNIX and UnixWare Copyrights. Novell was awarded summary judgments on a number of claims, and a number of SCO claims were denied. SCO was instructed to account for and pass to Novell an appropriate portion of income relating to SCO Source licences to Sun Microsystems and Microsoft. A number of matters are not disposed of by this ruling, and the outcome of these are still pending.

Lessons learned.

---

6 http://www.busybox.net
7 http://www.monsoonmultimedia.com
8 http://www.sitecom.com
9 http://www.netfilter.org
10 http://www.jbb.de/judgment_dc_munich_gpl.pdf
11 http://www.fortinet.com
12 http://www.sco.com
13 http://www.groklaw.net/pdf/IBM-718.pdf
14 http://sco.tuxrocks.com/Docs/Novell/Novell-377.pdf

Analyzing these cases reveal the need for:

- a better interpretation and enforceability of FOSS licenses that highlight the significance of compliance.
- sanctions in case of failure to comply.

These cases also pointed to the high costs of litigation for non-compliance. Many organizations are, therefore, trying to establish policies on the inclusion and verification of the presence of use and that allow them to verify the presence of third-party components in a proprietary code base.

4 LICENSE COMPLIANCE: AN ANALYSIS OF CURRENT PRACTICES

Nowadays, organizations developed certain metrics for managing compliance in FOSS development. Following are some of the best practices for compliance to licenses currently in use (Fan et al., 2004).

Contributors Agreement. A FOSS project may require a way of confirmation from its contributors through agreements that the author of the source code ensures the cleanliness of the code. With this effort, the project can be expected to produce a codebase that has clear IP provenance and protects the IP rights of others.

The contributors of Apache Harmony Project (supported by the Apache Software Foundation) are required to sign a contribution checklist[15] not only to ensure the cleanliness of the code but also to encourage the contributors to carefully examine their contributions before bringing to the project.

At Eclipse foundation[16], two levels of legal documentation are currently in use to cover all contributions of source code made by developers (Campbell, 2007). The Eclipse foundation requires that all contributions are made by the rightful copyright holder and under the Eclipse Public License[17] (EPL). A committer agreement[18] is signed by each committers to stipulate their contributions as their original work. If a committer is sponsored to work on an Eclipse project by a Member organization, then that organization is asked to sign a Member Committer Agreement[19] to ensure the intellectual property rights of the organization are contributed under the EPL. Furthermore, Eclipse ensures that the submissions through Eclipse web page are licensed to others under the terms of the Eclipse foundation.

Internal Review.

At every release and build of FOSS, organizations should verify whether any contaminated code is used in the software. A set of team members can verify the cleanliness of the source code in a project. The team can also verify that no unapproved modifications were made to external software components.

Compliance Tools.

Companies such as BlackDuck[20] and Palamida[21] offer products for ensuring IP compliance. These products compare the inputted source code against a knowledge base built from an assortment of OS projects and report matches between the inputted code and code in the knowledge base. However, we cannot evaluate these products as ideal solutions because these solutions fail to address formalization of licenses and license conflicts. Furthermore, it is highly difficult to verify the other kinds of IP infringements (such as patent and trademarks violations) made by the code.

---

15 http://harmony.apache.org/bulk\contribution\checklist.txt
16 http://www.eclipse.org/org
17 http://www.eclipse.org/legal/epl-v10.html
18 http://www.eclipse.org/legal/committer_process/EclipseIndividualCommitterAgreementFinal.pdf
19 http://www.eclipse.org/legal/EclipseMemberCommitterAgreementFinal.pdf
20 http://www.blackducksoftware.com
21 http://www.palamida.com

In academia, to the best of our knowledge, there is an obvious paucity of research on the topic of IP compliance associated with FOSS.

A compliance tool described in (Nordquist et al., 2003) gives an automated way to help software developers in detecting license conflicts. However, the scope of this tool is very limited and immature. Some informal and unstructured discussions about the concerns of IP and FOSS are explicated in the forums of Open Business Readiness Rating[22]. As there are no standards today for verification of compliance today, the perspicacity of present best practices is subject to individual organizations.

5 TOWARDS FORMALIZED LICENSE COMPLIANCE

Compatibility analysis is a process of matchmaking of candidate open source component licenses (at license clause level) in developing new software. The matchmaking algorithm performs the compatibility analysis between any two given licenses to decide whether they are compatible. A license is compatible with another license if all license clauses are compatible. The given candidate components can be combined, if their license are found compatible by the algorithm.

Open Digital Rights Language (ODRL) (Iannella, 2002) is an open standard language for the expressions of terms and conditions over assets, in open and trusted environments. Although the ODRL is a right expression language for specifying rights over digital assets, we can use it for expressing a software license in machine interpretable way. A machine interpretable representation of a FOSS license presented in this paper is not a substitute for a legal license but has as its goal the construction of tools that can assist with license compliance checking.

The representation of FOSS licenses and the matchmaking algorithm are based on well-established results presented in our earlier work (Gangadharan et al., 2007a). A license L_S in ODRL for a software S consists of a finite set of models (generally referred as license clauses), each of which further consists of a set of elements. Elements can be specified with value or without value (empty element having the element type only). Elements can contain other elements that can give rise to an arbitrarily deep hierarchy of elements within elements.

Two licenses are compatible, if all their respective license clauses are compatible. Two license clauses are compatible, if their elements are compatible. Elements are compatible:

- if they are of the same type and have the same values, or
- if one of the elements is unspecified, and the other element is compatible with all elements of this type (e.g. attribution is compatible with non-attribution)
- if they one element can be redefined as another (e.g. the right to derive from an asset implies that you can adapt it, or compose it with other assets)

A partial representation of a BSD style license in ORDL is as follows.

```
1. <o-ex:offer>
...
2.   <o-ex:requirement>
3.      <o-cc:attribution/>
4.   </o-ex:requirement>
...
5. </o-ex:offer>
```

A partial representation of a GPL license in ODRL is given as follows. We represent the Copyleft clause of GPL as sharealike in ODRL Creative Commons Profile and the indemnity clauses in the ODRL Service Licensing Profile (Gangadharan et al., 2007b).

[22] http://www.openbrr.org/forums/viewtopic.php?t=104

```
1. <o-ex:offer>
...
2.    <o-ex:requirement>
3.       <o-cc:sharealike/>
4.    </o-ex:requirement>
5.    <sl:indemnity>
6.       <sl:thirdpartyinfringementsclaims/>
7.    </sl:indemnity>
...
8. </o-ex:offer>
```

Following the matchmaking algorithm, we compare licenses at the license clause level. Line 2 of both licenses are *<o-ex:requirement>* models. The element in line 3 of GPL (*<o-cc:sharealike>)* is unspecified in BSD. If there is a sharealike in one license, but not the other, the two licenses are deemed incompatible, and matchmaking ends.

6 CONCLUDING REMARKS

License compliance in FOSS systems is a significant issue today and many organizations using FOSS are focusing on this issue. In this paper, we have analyzed the causes of compliance issues and we have reviewed a set of organizational best practices currently in practice and their limitations. Towards an automated license compliance management, we have proposed a novel algorithm that analyzes compatibility between FOSS licenses expressed in ODRL. Although the given ODRL representation of FOSS licenses and the algorithm for matchmaking licenses is incomplete, this work is a humble beginning for an ambitious representation of FOSS licenses in machine interpretable form and the analysis of license compatibility.

Acknowledgements

This work was partially supported by funding from the IST Programme under the 7th Research Framework Programme of the European Union, within the research project COMPAS (Compliance-driven Models, Languages, and Architectures for Services).

References

Rufn, M. and Ebert, C. (2004). Using Open Source Software in Product Development: A Primer. IEEE Software, 21(1).

Fan, B., Aitken, A., and Koenig, J. (2004). Open Source Intellectual Property and Licensing Compliance: A Survey and Analysis of Industry Best Practices. http://olliancegroup.com/opensource/compliance_best_practices.php.

Campbell, J. (2007). Open Source Software - Clarifying the IP Trail. http://www.talentfirstnetwork.com.

Nordquist, P., Petersen, A., and Todorova, A. (2003). License Tracing in Free, Open, and Proprietary Software. In Proceedings of the Northwestern Conference by the Consortium for Computing Sciences in Colleges.

Iannella, R. (2002). Open Digital Rights Language (ODRL) Version 1.1. http://odrl.net/1.1/ODRL-11.pdf.

Gangadharan, G.R., Weiss, M., D'Andrea, V., Iannella, R. (2007a). Service Licensing Composition and Compatibility Analysis. In Proceedings of the Fifth International Conference on Service Oriented Computing (ICSOC).

Gangadharan, G.R., D'Andrea, V., Iannella, R., and Weiss, M. (2007b). ODRL Service Licensing Profile (ODRL-S). In Proceedings of the 5th International Workshop for Technical, Economic and Legal Aspects of Business Models for Virtual Goods.