The Journal of Systems and Software 102 (2015) 72–87

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

An insight into license tools for open source software systems

Georgia M. Kapitsaki [a], Nikolaos D. Tselikas [b,*], Ioannis E. Foukarakis [b]

[a] Department of Computer Science, University of Cyprus, 75 Kallipoleos Street, P.O. Box 20537, CY-1678 Nicosia, Cyprus
[b] Communication Networks and Applications Laboratory, Department of Informatics and Telecommunications, University of Peloponnese, End of Karaiskaki Street, 22 100 Tripolis, Greece

ARTICLE INFO

Article history:
Received 11 December 2013
Revised 21 November 2014
Accepted 28 December 2014
Available online 30 December 2014

Keywords:
Free/Libre/Open Source Software
License identification
License compatibility

ABSTRACT

Free/Libre/Open Source Software (FLOSS) has gained a lot of attention lately allowing organizations to incorporate third party source code into their implementations. When open source software libraries are used, software resources may be linked directly or indirectly with multiple open source licenses giving rise to potential license incompatibilities. Adequate support in license use is vital in order to avoid such violations and address how diverse licenses should be handled. In the current work we investigate software licensing giving a critical and comparative overview of existing assistive approaches and tools. These approaches are centered on three main categories: license information identification from source code and binaries, software metadata stored in code repositories, and license modeling and associated reasoning actions. We also give a formalization of the license compatibility problem and demonstrate the role of existing approaches in license use decisions.

1. Introduction

Free/Libre/Open Source Software (FLOSS) or free and open-source software (FOSS) (Androutsellis-Theotokis et al., 2011) has assisted in the spread of emerging technologies, allowing users to utilize freely publicly available software and developers to incorporate third party source code into their implementations. Individual and already tested libraries are often used as building blocks for larger software systems, offering reusable functionality and providing the means for faster time-to-release. Various open source communities consisting of active developers and bug fixers for specific projects can be encountered ranging from small to very large groups depending on the popularity of the software system. The terms under which the software has become available and is provided for use are depicted in the corresponding licenses (Lawrence, 2004). Licensing is a legal issue, since software is highly linked with intellectual property. In general, licenses "*provide access rules that allow other people to go through the legal firewall and use the intellectual property*" (Lindberg, 2008). In open source software, licenses express how the software can be used further by the potential users differentiating between user rights and obligations. The strong importance of license use is also reflected in the research community that has shown a rising interest in open source software licensing in the last years (Alspaugh et al., 2009; Hemel et al., 2011; Sojer and Henkel, 2011).

As the number of components in software systems increases, so does the complexity of deciding which license(s) can be applied on the final system, or of checking if there are any incompatibilities among the terms defined in the licenses adopted in the different software components. Especially during the development phase, it is usual that software engineers include additional – and often redundant – dependencies in their code light-hearted, without checking possible licensing violations (Sojer and Henkel, 2011). In an enterprise world, where commercial software is often distributed against high prices, such an issue cannot be treated light-hearted (Douglas, 2011). Lots of different licenses have appeared containing various bounds and conditions on the software use: GNU General Public License (GPL), Apache License, MIT License, to name a few. Things are getting even more complex because each license may have multiple versions and each version is independent from a legal point of view, and if we consider other kind of licenses that are critical to understanding collaborations in FLOSS projects, i.e., individual Contributor License Agreements (CLAs). Without copyright assignments or CLAs, changing a software license requires the consent of every contributor to that system (Jensen and Scacchi, 2011). An important parameter, but out of the scope of this article, is that the legal interpretation of FLOSS licenses may differ among countries and especially between the United States and European courts as can be seen from several cases on FLOSS software copyright and/or licensing issues (Peeters, 2007; Hassin, 2007).

* Corresponding author. Tel.: +30 2710 372216; fax: +30 2710 372160.
E-mail addresses: gkapi@cs.ucy.ac.cy (G.M. Kapitsaki), ntsel@uop.gr (N.D. Tselikas), ifouk@uop.gr (I.E. Foukarakis).

http://dx.doi.org/10.1016/j.jss.2014.12.050
0164-1212/

Table 1
Open source license types.

	License type	Examples
	*Permissive:* *software may be distributed under any license*	MIT BSD (Berkeley Software Distribution) Apache v2.0
	*Weak copyleft:* *if original software using this license is modified derivative work needs to carry the same license; otherwise derivative work may be distributed under any license*	GNU LGPL (Lesser General Public License) MPL (Mozilla Public License)
	Strong copyleft: *any modification to software using this license needs to be distributed under the same license*	GNU GPL (General Public License) v2, v3 OSL (Open Software License)

Finding the right license(s) associated with software source code, keeping these licenses up to date, choosing the appropriate license for each new software product, updating this choice, when and if necessary, are important issues for any open source user and for software engineers employing open source software for various purposes. In the current work we address these issues by giving an overview of available licensing tools. These are mainly found in license classification, identification and compatibility approaches. Additionally, a critical comparison on the existing tools is performed based on a set of metrics that we introduce. The contribution of our work lies in the introduction of these metrics for license assistive tools as well as in the overview and comparison of existing tools along with an approach toward the formalization of the license compatibilities problem. The main aim of this work is to provide a useful reference on FLOSS compatibilities for software researchers and practitioners, formalize the available areas and approaches and point out existing gaps and needs for further research.

The rest of this paper is structured as follows: Section 2 introduces license compatibility and performs a classification of the existing approaches and tools. Each category is presented in detail with representative approaches in Sections 3 through 5 containing methods for license information identification, license information presented in online repositories of source code and license modeling along with techniques for reasoning actions on license compatibility. A comparison of the features offered in each tool category is given in Section 7 based on the metrics introduced in Section 6. Finally, Section 8 concludes the paper providing a short reference to future advancements.

2. License compatibility and tools

2.1. License classification and violations

Open source licenses can be characterized either as permissive (called also academic) or copyleft (called also reciprocal). The latter category can be further divided into weakly protective and strongly protective. This main distinction is depicted in Table 1 along with some representative examples. Permissive licenses have minimal requirements. Open source software with a permissive license can be distributed as part of a larger product under almost any other license, while the only requirement is the attribution to the original authors. The distinction between strong and weak copyleft lies in the permissions given in a derivative work, i.e., *work adapted from the originally copyrighted item* (Lindberg, 2008). Any derivative work of strong copyleft-licensed software needs to be distributed with the same license. On the other hand, if the software used carries a weak copyleft license, the derivative work can be distributed under another license as long as it has not modified the weak copyleft-licensed software used. Some licenses may make a distinction on the type of linking to the original software, although generally weak copyleft permits both dynamic and static linking of the derivative work to the original software. Further information on the characteristics of each license category can be found in a relevant publication (Kechagia et al., 2010). Strong and weak copyleft should not be confused with full or partial copyleft, which refers to the part of the copyleft-licensed software that must be made available under the same copyleft license, when modified or distributed: in *full* copyleft this applies to all parts of the software, whereas in *partial* some modifications may be distributed under a different license.

Formal lists of licenses can be found in the Open Source Initiative (OSI) and the Free Software Foundation (FSF). OSI has approved till this point 70 licenses including one that was deprecated by its author, whereas FSF lists 88 licenses. Placing a license into a specific category is not straightforward. Especially the distinction between permissive or weak copyleft, and between weak or strong copyleft, is not always clear. This depends on the specific terms in the license text, but also on whether there is a differentiation between static and dynamic use of the code in the license. Hence, different organizations, such as GNU[2] or the institute for legal issues regarding free and open source software ifrOSS,[3] may place the same license into different categories or even disagree on whether a license can be regarded a free license. Table 2 includes a categorization of most OSI-approved licenses excluding licenses with no obvious consensus on their category and licenses that are rarely used (e.g., Naumen Public License, Open Group Test Suite License).

As indicated, there are cases where the license classification is not straightforward. For instance, EPLv1.0 (Eclipse Public License) allows using the object form of an original work under EPL in a product to be licensed under a commercial license provided that the EPL portion conforms to the terms of the EPL. However, the license is considered as strong copyleft by ifrOSS, since the above is not allowed when the source code of the EPL licensed work is used directly. Another case is the CPAL license (Common Public Attribution License) derived from MPL (Mozilla Public License). Although its connection with MPL gives an indication for a weak copyleft character, the license term that states that the copyleft comes into effect when software with even unmodified version of the CPAL-licensed software is distributed over a network brings this license closer to the strong copyleft AGPL license (Affero General Public License).

The variety of open source licenses makes it difficult for organizations to cope with incompatibilities that might exist due to the use of software libraries based on different licenses. License *A* is considered one-way 'compatible' with license *B*, if software that contains components from both licenses can be licensed under license *B*. The term 'one-way' is used to highlight that license *A* is compatible with license *B*, but the reverse case (i.e., license *B* is compatible with license *A*) is not assured. As shown later in Section 5, the reverse case could be feasible only if both licenses *A* and *B* belong to the same category, while it is never true if license *A* is a permissive license and license *B* is a copyleft one. In other words we could say that license *A* is univocally

[2] http://gnu.ist.utl.pt/licenses/license-list.html.
[3] http://www.ifross.org/en/license-center#term-222.