The Journal of Systems and Software 131 (2017) 386–401

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Automating the license compatibility process in open source software with SPDX

Georgia M. Kapitsaki[a,*], Frederik Kramer[b,1], Nikolaos D. Tselikas[c,2]

[a] Department of Computer Science, University of Cyprus, 75 Kallipoleos Street, P.O. Box 20537, CY-1678, Nicosia, Cyprus
[b] Otto von Guericke University, Universitaetsplatz 2, D-39106, Magdeburg, Germany
[c] Department of Informatics and Telecommunications, University of Peloponnese, End of Karaiskaki Street, 22 100, Tripolis, Greece

ARTICLE INFO

Article history:
Received 15 June 2015
Revised 11 January 2016
Accepted 23 June 2016
Available online 27 June 2016

Keywords:
Open Source Software
License compatibility
License violations
Software Package Data Exchange

ABSTRACT

Free and Open Source Software (FOSS) promotes software reuse and distribution at different levels for both creator and users, but at the same time imposes some challenges in terms of FOSS licenses that can be selected and combined. The main problem linked to this selection is the presence of a large set of licenses that define different rights and obligations in software use. The problem becomes more evident in cases of complex combinations of software that carries different – often conflicting – licenses. In this paper we are presenting our work on automating license compatibility by proposing a process that examines the structure of Software Package Data Exchange (SPDX) for license compatibility issues assisting in their correct use and combination. We are offering the possibility to detect license violations in existing software projects and make suggestions on appropriate combinations of different software packages. We are also elaborating on the complexity and ambiguity of licensing detection in software products through representative case studies. Our work constitutes a useful process towards automating the analysis of software systems in terms of license use and compatibilities.

1. Introduction

Free and Open Source Software (FOSS) is used in a variety of applications gaining popularity not only among the open source community, but also as part of commercial applications (Androutsellis-Theotokis et al., 2011). It cannot be doubted that FOSS provides new opportunities to software engineers and organizations to reuse existing knowledge and profit from the benefits of an open Information and Communications Technology (ICT) model. According to a survey performed by Sojer and Henkel (2010) with responses from 686 developers around 30% of functionality offered in software projects is based on reused code.

Licenses applied on FOSS products contain *the terms under which the software is made available and under which conditions it can be used, integrated, modified and redistributed* (Rosen, 2004). Licenses usually differentiate between user rights and obligations and are very diverse in these elements they define. Due to the nature of natural language text used in these licenses, their interpretation is not straightforward. The plethora of FOSS licenses renders this activity even more difficult (Di Penta et al., 2010). The problem that many software vendors often face is how to incorporate third party software in their implementations correctly without causing any license violations, remaining hence legally compliant. Compliance refers to conforming to a rule, such as a specification, policy, standard or law and is applicable for organizations that use open source software. License violations are a complex issue in FOSS due to the variety of licenses that state different and often contradicting conditions of use referring mainly to conditions for software modification and redistribution. When software libraries licensed under different terms are jointly used, this license diversity may lead to license incompatibilities. Licenses cover a range from very permissive ones, such as the MIT license and the Academic Free License (AFL), to highly restrictive, such as the GNU General Public License (GPL) and the Open Source License (OSL).

Main research questions that arise in the above area are 1) how we can detect license violations in existing software packages, 2) how we can make suggestions for appropriate license use, and 3) what is the level of complexity of license decisions for large software systems comprising of many libraries. Recognized organizations, such as Motorola, Eclipse and HP, participate in the process of defining a specification that assists in integrating license information into the distribution of software packages. In this framework, packages can be formatted according to the Software

* Corresponding author. Fax: + 357 2289 2701.
E-mail addresses: gkapi@cs.ucy.ac.cy (G.M. Kapitsaki), frederik.kramer@ovgu.de (F. Kramer), ntsel@uop.gr (N.D. Tselikas).
[1] Fax: +49 391 6754840
[2] Fax: +30 2710 372160

http://dx.doi.org/10.1016/j.jss.2016.06.064
0164-1212/

Package Data Exchange (SPDX) specification (Linux Foundation and its Contributors, 2015). In this work we are addressing license violation detection in the framework of the above research questions by using SPDX files. SPDX was chosen due to its gaining popularity among software organizations.

We have designed and implemented a compatibility process that examines whether the information contained in a SPDX file regarding the license(s) applied on the software package is correct. This process is captured in our *SPDX Violation Tools (SPDX-VT*). SPDX-VT assists in: 1) verifying that the information of license(s) applied on the software package is correct, 2) identifying any license incompatibilities among the licenses of the software package, and 3) making suggestions for licenses that can be applied without causing violations when combining licenses from different software. Our contribution is threefold: examination of violations that exist in a specific software product, suggestions for appropriate combinations of software products, and deeper understanding of the relations found in the licenses of files employed in different products through three case studies. The above are complemented by the introduction of a graph that contains compatibilities between licenses; a side contribution of our work, that is evolving. This is the first work that approaches license violations in a specification setting that can offer a global solution for license compatibility enforcement. The preliminary version of this work has been presented in a previous work (Kapitsaki and Kramer, 2015). In the current paper, we are expanding our existing work by a more complete incompatibility analysis process that includes: 1) the introduction of a faster compatibility algorithm (modified Breadth First Search is introduced as contribution of the current work and is employed instead of the Floyd–Warshall algorithm adopted in the previous publication), 2) a wider evaluation on a larger number of representative FOSS projects (i.e., 20 projects), 3) more validation results on the software packages including a manual analysis of false positives in our approach, 4) analysis of the results through a number of case studies of popular open source software products (i.e., Apache Hadoop, Odoo and Shopware). The aim of this final contribution is to demonstrate the complexity of license handling, the importance of license identification tools and the issues that require special handling in the license compliance process.

The rest of this paper is structured as follows: Section 2 introduces the Software Package Data Exchange and license compatibility along with a description and discussion on modeling license compatibilities through a graph. Section 3 presents related work in the area of licensing modeling and violations, whereas the proposed compatibility process is presented in detail in Section 4. The evaluation of the work along with examined projects is presented in Section 5. Section 6 is devoted to specific case studies and, finally, Section 7 concludes the paper providing a short reference to future advancements.

2. License compatibility and the SPDX standard

The different terms that can be found in FOSS licenses affect their prospective combinations leading to license incompatibilities. In order to be able to examine such combinations, the licensing information of software packages is important. The latter is addressed in the Software Package Data Exchange employed in this work. The above are further analyzed in this section in order to introduce the area of the current research work.

2.1. The Software Package Data Exchange

The Software Package Data Exchange specification with 2.0 being its latest formal version recently released can be used in distributing information on the licenses of software packages (Mancinelli et al., 2006; Linux Foundation and its Contributors, 2015). According to the specification, *it is a standard format for communicating the components, licenses, and copyrights associated with a software package*. An SPDX file is associated with a particular software package and contains information about that package in the SPDX format. The main aim was to create a common data exchange format for information that is usually handled independently in different organizations, so that information about software packages and related content can be shared among those organizations and beyond with accurate data concerning the parts of the software.

SPDX has been initiated by the Linux foundation and its current contributors include commercial companies such as BlackDuck, Fujitsu, HP, Siemens, as well as non-profit organizations such as the Eclipse and the Mozilla foundations. SPDX covers over 200 licenses including the ones approved by the Open Source Initiative (OSI[3]). Licenses supported by SPDX can be used independently in a software package, i.e., as single license identifiers. These single license identifiers are complemented by the *SPDX License Expressions* that provide more accurate expressions for the licensing terms of a software product. Expressions allow the inclusion of license exceptions and the combination of license identifiers employing a number of operators offered by SPDX (e.g., AND, OR, WITH and +). For instance, the AND operator is used in license expressions, where the software product needs to comply with more than one licenses at the same time (e.g., GPL-2.0 AND MIT), whereas the WITH operator indicates license exceptions (e.g., GPL-2.0 WITH classpath-exception).

Software products in SPDX can have various formats in order to accommodate the need of readability by humans and by software tools: RDF (Resource Description Framework) file, a textual key-value pair format referred to as tag format and a spreadsheet format. The latter is provided for better readability and manual changes. With respect to license information that can be used for license compatibility and violation purposes, the following RDF fields are relevant:

- **Declared license - *licenseDeclared:***

The declared license indicates the license that has been asserted for the package as indicated by its authors (i.e., the creators of the software product). This is the license identified in text in one or more files in the source code package (e.g., in a LICENSE.txt file) and is not intended to capture license information obtained from an external source, such as the package website. This latter information is indicated in the Concluded license field instead. SPDX files may include multiple licenses in this field in cases where the software product carries more than one licenses at package level (and not in its independent files and libraries).

- **Concluded license - *licenseConcluded*:**

The concluded license indicates the license that the creator of the SPDX file concluded. This may differ from the declared license. Specifically, license concluded is used to indicate the license that was manually identified by the creator of the SPDX file taking into account the manual observation of the package structure (e.g., text files with license information) and the results from any scanning tools (for license identification). This may also indicate the license from the package website. The SPDX specification advises the users to provide detailed explanations for cases, where the Concluded license does not match the Declared license of the package, since the usual case would be that both licenses are identical.

- **Extracted license - *hasExtractedLicensingInfo***

This field includes an *ExtractedLicensingInfo* element that is used mainly to denote cases, where the SPDX file contains a license not

[3] http://opensource.org/

included in the SPDX license list or for cases, where the license text is provided. It is not included in version 2.0 of the specification, where only the element *ExtractedLicensingInfo* is used.

- **All Licenses Information from Files - *licenseInfoFromFiles*:**

This field indicates the list of all licenses found in the files of the software package.

- **License Information in File - *licenseInfoInFile*:**

It contains the license information actually found in the respective file mentioned, if any. Licenses for each file of the package are mentioned (for instance using the license indicated in the header in a source file).

- **License List Version - *licenseListVersion*:**

This is an optional field that indicates the version of the SPDX License List used when the SPDX file was created, since the SPDX License List changes over time with new versions.

- **Comments on License - *licenseComments*:**

It indicates relevant background information or analysis for arriving at the Concluded license of the package. This is the field that should be used in order to describe the differences between the Declared and the Concluded license(s) of the file and explain the reasons for this difference.

Fig. 1 shows an example of the SPDX file of the software package py2exe with the presence of some of the above fields. For better handling and for integrating with SPDX files the SPDX consortium has introduced a number of tools including file converters and comparators (e.g., SPDX Viewer and SPDX Compare utilities). Although the list of tools is constantly expanding, more elaborated tools on the license use are missing. Novel tools could collaborate with SPDX files in order to assist software engineers, managers and legal departments in drawing useful conclusions on the use of software licenses in cases, where complicated software architectures prevail. The issue of license compatibilities is major for both small and large organizations, such as in the use case described by Debian[4], where licenses need to be identified and built into SPDX.

2.2. License compatibility

Licenses are categorized as either permissive or copyleft. 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). Permissive licensing allows the software to be part of a larger product under almost any other license. Copyleft licenses are posing more restrictions. Copyleft is further divided into weak and strong copyleft. If the software used is weak copyleft-licensed, the created work can be distributed under another license as long as no changes are made in this weak copyleft-licensed software used. Strong copyleft requires all derivative work that uses strong copyleft-licensed software to be distributed with the very same license. Public domain may also be applied on software products stating that the intellectual property rights of the work have expired and cannot be claimed anymore (Boyle, 2009). However, there is no equivalent for public domain in the European Union. The most popular licenses are found in MIT, Apache version 2.0 and BSD permissive licenses, GNU Lesser General Public License (LGPL) and Eclipse Public License (EPL) for weak-copyleft and the different versions of GNU General Public License (GPL) and Affero General Public License (AGPL) for strong copyleft licenses.

Despite the above main license categories, there are cases for which different organizations provide different interpretations of the license text placing a license in different categories. For instance, the Eclipse Public License 1.0 (EPL-1.0) is generally considered a weak copyleft one, but the German institute for legal issues regarding free and open source software IfrOSS places the license under the strong copyleft category. Such differences lead in ambiguities in license compatibility. Formal lists of licenses are provided by the Open Source Initiative and the Free Software Foundation (FSF[5]). OSI has approved 70 licenses including one that was deprecated by its author, whereas FSF currently lists 88 licenses. In this work 16 licenses approved both by OSI and FSF are covered.

License compatibility refers to the ability to combine different FOSS licenses into the same software product. Two FOSS licenses are considered compatible in the following sense:

Definition 1–License compatibility: license V_1 is "one-way compatible" with license V_2, if software that contains components from both licenses can be licensed under V_2. Alternatively, anyone who conforms to the rules of license V_1 would also conform to the ones defined by V_2.

According to Gangadharan et al. (2012) "*compatibility analysis is a process of matchmaking of candidate open source component licenses (at license clause level) in developing a new software*". This analysis is necessary due to the different rights and obligations that accompany each license and is a vital part of the license compliance process. License compliance describes the goal of organizations to ensure that they are aware of and take steps to comply with relevant laws and regulations in respect to the use of open source software. We shall therefore refer to a license compliance process as a well-defined procedure in the spirit of the process proposed and conducted within this work. License compliance is also linked with the compatibility process that can be followed for detecting license incompatibilities (Foukarakis et al., 2013).

Compatibility depends also on the ways software is connected and on the system architecture. A main distinction is made between static and dynamic linking. In the current work we do not make these distinctions. Although some licenses may make a distinction on the type of linking to the original software (e.g., LGPL), we do not consider the type of linking in the current state of the work, since the majority of programming languages follow a dynamic linking approach.

Compatibilities among licenses can be modeled via a graph that shows connections between compatible licenses (Fig. 2). In the current work a set of popular licenses among the ones mostly indicated in FOSS repositories, such as SourceForge and the KnowledgeBase of BlackDuck, are used. The complete set of licenses can be found in Kapitsaki and Kramer, 2015. The hard task of compatibility analysis for the construction of the graph was based on various sources. These include Wikipedia article entries on licenses, information from the license texts and forums, as well as the GNU hosted page on license compatibilities with GPL that contains valuable information on the rights and obligations of many FOSS licenses. The initial version of the graph can be found in an existing publication (Kapitsaki and Kramer, 2015), whereas the latest version is presented in detail also in a previous work containing a smaller set of main licenses (Kapitsaki et al., 2015). In this work, we use licenses covered in both existing versions of the graph of Fig. 2. Note that not all edges and their explanations are presented here. The reader may refer to the previous work (Kapitsaki et al., 2015). The justification for new nodes and edges appearing in Fig. 2 in comparison to this previous work are explained in the Appendix. The plus (+) sign used in some licenses

[4] http://wiki.spdx.org/view/Technical_Team/Use_Cases/2.0/Debian_has_an_interest_in_only_building_things_that_are_linking_license_compatible

[5] http://www.fsf.org/

```
<rdf:RDF …>
   <referencesFile>
      <File rdf:nodeID="A12">
         <copyrightText>copyright (c) 2000, 2001 thomas heller copyright (c) 2003 mark Hammond
         </copyrightText>
         <licenseConcluded>
            <ExtractedLicensingInfo rdf:nodeID="A13">
               <licenseName>MIT-style</licenseName>
               <extractedText>According to MIT license, add some modifications</extractedText>
               <licenseId>LicenseRef-2</licenseId>
            </ExtractedLicensingInfo>
         </licenseConcluded>
         <licenseComments></licenseComments>
         …
         <licenseInfoInFile rdf:nodeID="A13"/>
         <fileType rdf:resource="http://spdx.org/rdf/terms#fileType_source"/>
         <fileName>./py2exe 0.6.9/py2exe-0.6.9/source/run_ctypes_dll.c</fileName>
      </File>
   </referencesFile> …
    <hasExtractedLicensingInfo>
      <ExtractedLicensingInfo rdf:nodeID="A29">
         <licenseName>LGPL</licenseName>
         <extractedText>LGPL is referenced without a version number. Please look up LGPL in the
License Admin to view the different versions.</extractedText>
         <licenseId>LicenseRef-5</licenseId>
      </ExtractedLicensingInfo>
   </hasExtractedLicensingInfo>
    …
         <licenseDeclared>
          <DisjunctiveLicenseSet>
               <member rdf:resource="http://spdx.org/licenses/MIT" />
               <member rdf:resource="http://spdx.org/licenses/MPL-1.1" />
          </DisjunctiveLicenseSet>
         </licenseDeclared>
    …
    </SpdxDocument>
    …
</rdf:RDF>
```

Fig. 1. Extracts of the SPDX file of the py2exe software package.

means that we are referring to the indicated version of the license or later versions of that.

The rationale behind the use of the graph is the same as in Wheeler's slide that provides the only alternative available license graph (Wheeler, 2007), i.e., it can be used to see if different software products licensed under two or more distinct licenses can be combined in a new application under another license. This can be performed by following the edges coming out of these licenses to see whether they reach the same license at some point or if one of them is reachable from all other licenses of the distinct set. Please note that the above graph can be used by anyone in the research or software engineering community as a basis for an understanding on the license compatibility issues, bearing in mind that many organizations use different interpretations of the license text and consider different compatibilities.

It should be noted that, although license text is legal, it is closer to natural language, as each license creator can create the text without following specific guidelines. This lack of structure makes the application of Natural Language Processing (NLP) techniques for the purpose of license term extraction and the automatic graph creation insufficient. The purpose of the graph is to provide a simple notation and provide one and only conclusion about the compatibility between two licenses. Thus, at this level of granularity, where the license terms, i.e., license rights and obligations, are not depicted in the graph, NLP techniques on license texts that in most cases span over multiple pages would not assist the final accuracy of the graph. The study of NLP on a finer level of granularity on license terms is a useful – yet unsolved – research issue.

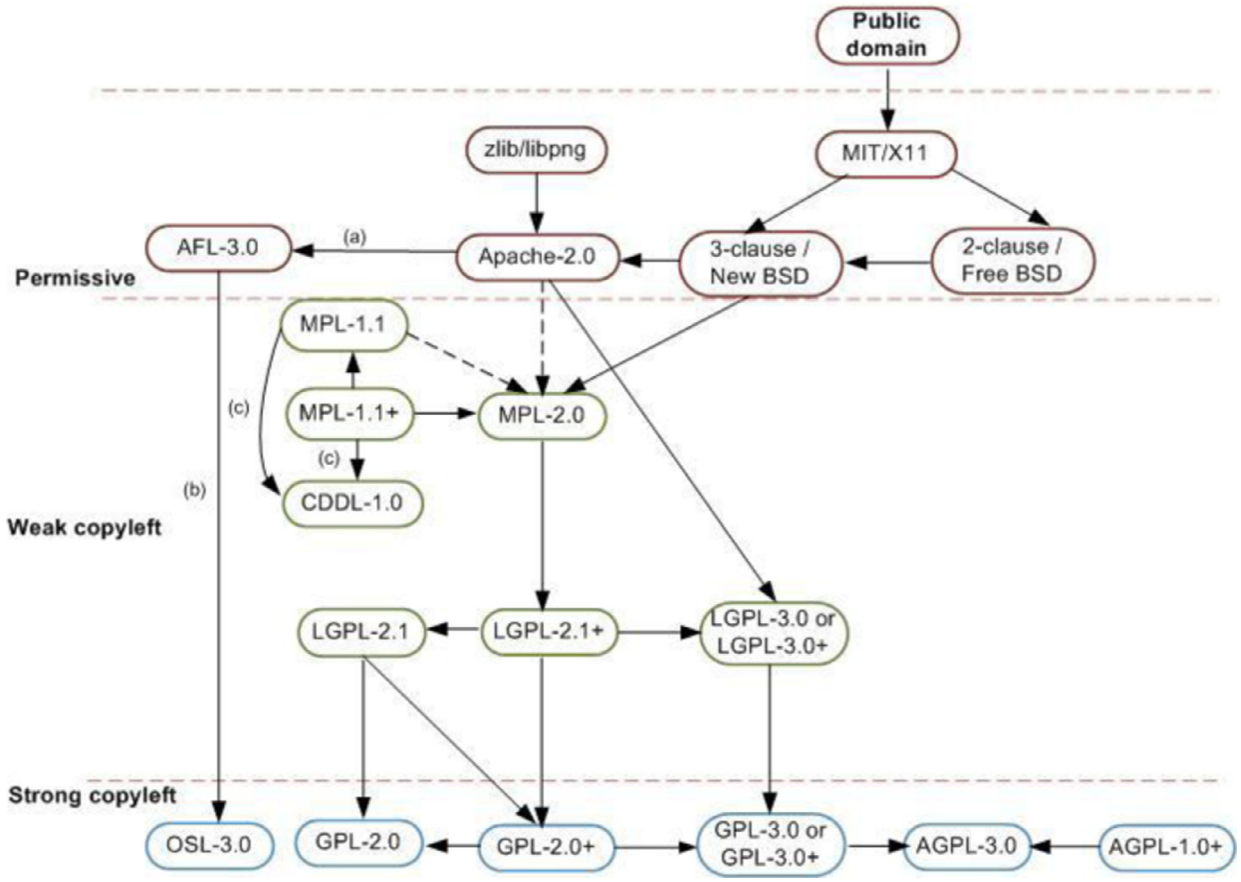


Fig. 2. Modeling license compatibility via a graph for popular licenses.

3. Related work

Open source software licensing has been addressed from different perspectives in the existing literature including licensing evolution and license extraction from software projects. In this work we focus on the presentation of previous works on license modeling and license violations that are both directly linked to the contributions of our work.

3.1. License modeling and standardization

License compliance has been addressed in many previous works with some focusing on modeling aspects of licenses along with reasoning on license rights and obligations. Graph-based approaches, such as the one presented in the current work, can be found in existing works that consider either abstract licenses or a smaller number of existing licenses (Wheeler, 2007). (Foukarakis et al., 2012). Such graphs are based on manual interpretation of each license. Wheeler (2007) proposed a first compatibility graph that pulls together various information sources. In comparison to this earlier slide, our work covers a wider range of licenses and recently contributed additional information sources including licenses that were potentially less popular or did not exist in 2007 (e.g., MPL-2.0), when Wheeler's slide was created.

Other approaches model license structure in a more detailed manner as rights and obligations of the licensee following similarly a graph representation, as in the metamodel of Alspaugh et al. (2010). The metamodel is based on an empirical analysis of the license texts in order to extract meaningful terms leading to an informed classification of natural language patterns. As open source licenses usually do not contain extensive compatibility rules, such as "*... this license is not compatible with...*", and since the metamodel was based on empirical analysis of the existing body of licenses it does not contain compatibility modeling components. In Qualipso (Gordon, 2010) an ontology for the modeling of software systems in Web Ontology Language (OWL) is introduced. The ontology contains a set of 8 FOSS licenses and their rights. For a detailed overview of license assistive tools the reader is referred to the work of Kapitsaki et al. (2015).

Apart from the modeling problem a standardization problem exists. License texts may either form part of the source file or may be missing completely. Even in cases where license information is put at the beginning of a source file, it usually does not follow proper standards or conventions. As already mentioned the SPDX format addresses this standardization problem and is the only approach with strong and continuing industry sponsorship and development effort. There are various tools that perform license information extraction from a given source file or repository to address this challenge, some of which are shown in Table 1. Available tools are the Automated Software License Analyzer (ASLA) (Tuunamen, 2009), the Open Source Software Licensing (OSSLI) (Lokhman et al. 2012), the Open Source License Checker (OSLC) (Xu et al., 2010), FOSSology (Gobeille, 2008), Ninka (German et al., 2010) and various tools provided by the open source community (i.e., SLIC, Licensee, ScanCode, LicenseFinder).

OSSLI uses a model-centric approach and is implemented as an extension to the Papyrus graphical modeling framework of the Eclipse platform. It focuses on direct support at development and compile time. OSLC, ASLA and FOSSology are standalone tools, whereas FOSSology is not only a standalone Java application but a multi-user web application. ASLA was proposed as a result of a feature and performance analysis performed on Nomos, which is developed as part of FOSSology, and OSLC (Tuunamen, 2009). Only OSSLI and FOSSology are considering SPDX. In FOSSology this is

Table 1
Main license identification tools.

	ASLA (Tuunamen, 2009)	OSSLI (Lokhman et al. 2012)	OSLC (Xu et al., 2010)	FOSSology (Gobeille, 2008)
Type	License checker	License checker	Licence checker	License management
License	GPL-2.0	MIT	GPL-2.0	GPL-2.0
Latest version	0.14	0.1.0	2.0	2.6
Date of latest version	07.03.08	05.03.2012	25.02.2007	22.12.15
Programming language	Java	Java	Java	PHP, Perl, C and Python
Origin	research	research	research	industry
SPDX consideration	×	√ (theoretical)	×	√
License dependency matrix	√ (with GUI support)	√	√	×

done through its integration with the Nomos and Monk / Ninka agents (used as alternative license scanners). Furthermore, FOSSology is the only tool that enables the regular scan of repositories and source code folders (using schedulers) and integrates a web-based management dashboard. Due to its integrated nature, its continuing industry support, its strong relation with the SPDX standardization body and its extensible web application nature, we use FOSSology as a basis for our work on license identification checking.

3.2. License violations

The aforementioned graph-based modeling approaches do not provide information about distinct rights and obligations that are defined in the detailed text of the license, but help in defining general rules for reasoning on compatibility among licenses. In another approach license rules are expressed in a legal rule language in XML (Extensible Markup Language) format and are applied to the model of a software system in order to detect possible licenses automatically through the Carneades argumentation system[6] (Gordon, 2011). The process is performed through a reasoner that searches for licenses that provide the required rights and/or belong to a specific class of the hierarchy as expressed in an argument map.

Some license identification tools contain license compatibility capabilities. However, in contrast to our work all aforementioned tools do not use a graph-based approach but a simple compatibility matrix (license dependency matrix in Table 1). Only ASLA supports the update of this matrix using its graphical user interface. More specifically, ASLA uses license rules that were defined based on dynamic and static linking of software components, whereas it includes user-definable compatibility rules. This license compatibility map and the ability to maintain those licenses within a graphical user interface are doubtlessly strengths of ASLA. OSSLI performs compatibility analysis on architectural level. It can be better seen as an extension of the Unified Modeling Language (UML) package model type and an assistant in constructing license compatible architectures than a compatibility analysis tool for a given source code repository. Violation detection is available on a limited basis in OSLC through the detection of incompatible imports in a specific source file. Apart from the above solutions, the Kenen tool of German and Di Penta (2012) identifies compliance to open source licenses for Java applications. Some commercial tools provide support, such as the KnowledgeBase of BlackDuck and Artifactory by JFrog), an internal repository that acts as a proxy between a build tool and the outside world. Artifactory is able to detect license conflicts based on license management rules.

As aforementioned another aspect relevant to license compatibility is the software system's architecture (Alspaugh et al., 2010). This linking is also reflected in a UML profile for the addition of license and intellectual property-relevant information in the UML model of a software system used in the framework of the aforementioned OSSLI (Lokhman et al., 2012). It performs also conflict detection on licenses investigating whether license terms of different licenses conflict when linked or interconnected, but its implementation is still at a premature state.

To the best of our knowledge this is the first work that proposes automating the license compatibility process in the framework of SPDX. In contrast to existing tools that focus mainly on license information extraction the current work provides the opportunity to standardize license compatibility-relevant activities in a global specifications setting. Regarding the modeling of license compatibilities our approach uses a graph representation in contrast to existing tools and this is performed for several reasons. A graph representation allows to visualize compatibility paths that in turn assists human decision makers to take decisions more easily and accurately. A graph allows also through its expansion to visualize the dependency map for different licenses focusing either on the whole license set or on a specific subset of FOSS licenses. Therefore, the use of a dependency graph can improve human decision interaction and facilitate the collaborative maintenance of license compatibilities.

4. Examining license compatibilities in SPDX

The main contribution of this work is the SPDX violation check that examines and detects potential violations in SPDX files. SPDX-VT understands the structure of SPDX files and suggests actions for license compliance based on the findings in the files utilizing the license graph and the compatibility algorithm described below.

4.1. Graph structure

As aforementioned license compatibility is modeled by a directed graph (Fig. 2). Graph directionality is apparent due to the "one-way compatibility" between licenses. The graph is also acyclic because usually compatibility stems from a less restrictive to a more restrictive license. For instance, although the MIT and the BSD licenses seem equivalent, software that contains both licenses should be licensed entirely under the more restrictive license. Using the less restrictive license would cause violations due to the lack of conformance to obligations present in the more restrictive license. In this case this would be the BSD license. Hence, it is not feasible to return to the less restrictive license again via edge reachability.

The compatibility evaluation of FOSS licenses can be really complex. For example, the MPL-1.1 and all versions of the GPL are incompatible (Rosen, 2014) and cannot be shipped jointly in one single software artifact. The problem can be solved by proceeding from MPL-1.1 to MPL-2.0 and subsequent shipping under GPL-2.0 terms for instance. In order to express the above incompatibilities and discourage the above combinations, i.e., MPL-1.1 with versions of LGPL, GPL and AGPL as well as Apache-2.0 with LGPL 2.1 and GPL-2.0, a special edge type that deactivates transitivity in the graph is used denoted with a dashed line in the graph of Fig. 2.

[6] http://www.qualipso.org/licenses-champion

Therefore, two types of edges both referring to one-way compatibility between the adjacent licenses are used: 1) solid edges and 2) non-transitive edges. The transitive relation in the compatibility between connected licenses is applicable in the graph, unless a path starts with or contains a non-transitive edge. Transitivity is also applicable in the graph, if a path ends with a non-transitive edge. The direction of non-transitive edges denotes the last possible vertex of a transitive (compatibility) path. For example, zlib license is compatible with MPL-2.0 (the non-transitive edge is the last one in the path), but it is not compatible with GPL-2.0 (this time the corresponding path contains a non-transitive edge that is not the last one in the path). In summary, license compatibility does not apply for licenses that reach the same node, if the path contains a non-transitive edge that is not the last edge of the path. License compatibility is, therefore, defined as:

Definition 2–Compatibility path (in the license graph): given two vertices V_1 and V_2 in the license graph G(V, E) a valid compatibility path between V_1 and V_2 exists, when license V_1 reaches V_2 via a number of vertices on a path that either does not contain any non-transitive edges or, when it does, there is only one non-transitive edge as last edge on the path, or vice versa (from V_2 to V_1).

Using the introduction of the two different edge types, graph compatibility is defined as:

Definition 3–License compatibility graph: a directed graph G(V, E) with a) each vertex V representing a specific license with its version, and b) each edge E connecting two compatible licenses V_1 and V_2 (with direction from V_1 to V_2) either as a solid or non-transitive edge.

The problem of detecting license incompatibilities can be expressed by examining whether a number of FOSS licenses can be combined in a new software product under one single or more alternative licenses. When examining compatibilities between two licenses this basically resolves into the following graph problem considering the license graph:

Definition 4–Problem of compatibility between two FOSS licenses: given two vertices V_1 and V_2 in the license graph G(V, E) find whether compatibility paths p_1 from V_1 to V_X and p_2 from V_2 to V_X both exist with V_X being the vertex with the smaller number of hops from both V_1 and V_2 in comparison to other vertices in the graph reached from V_1 and V_2 (i.e., V_X is the closest vertex).

In the above case V_X would be the less restrictive applicable license closest to both V_1 and V_2, although more licenses (more restrictive), with which both V_1 and V_2 are compatible, would also be eligible as applicable licenses. V_X may also be the same vertex as V_1 or V_2. The above rationale is generalized for the case of more vertices:

Definition 5-Problem of compatibility between n FOSS licenses: given n vertices $V_1..V_n$ in the license graph G(V, E) find whether for each vertex V_i paths to a common vertex V_X all exist with V_X being the node with the smallest number of hops from $V_1..V_n$ in comparison to other vertices in the graph.

4.2. Algorithm for violation detection

The aforementioned problem of compatibility between *n* FOSS licenses falls into graph multiple-source reachability that refers to determining where we can get to in a directed graph. Several algorithms for graph reachability problems in linear time exist, such as: Breadth First Search (BFS) (Lee, 1961), iterative deepening Depth First Search (DFS) as used for instance by Ausiello et al. (2015), and the computation of the transitive closure of the graph once – for instance by the Floyd–Warshall algorithm (Floyd, 1962) – and the subsequent storing of this graph and use for queries on license compatibility. Although the transitive closure is the best choice for cases where a large number of queries need to be performed on the graph indicated also in the initial version of SPDX-VT (Kapitsaki and Kramer, 2015), for efficiency purposes (keeping the execution time as short as possible) we have opted for BFS that is more appropriate for cases, when only one or few queries are performed.

In order to be compliant with the different types of edges on the license graph, we have modified BFS so that the algorithm does not consider as reachable edges without transitivity:

```
1   algorithm LICENSE_BFS(G, v)
2   begin
3     let Q be a queue
4     Q.enqueue(v)
5     label v as visited
6     while Q is not empty
7     v ← Q.dequeue()
8     process(v)
9     // if v is reached from a non-transitive edge
10      if incoming edge of v is non-transitive
11        label v as visited from non-transitive
12      for each edge from v to w in G.adjacentEdges(v)
13    begin
14      if w is not labeled as visited OR (w is labeled as visited
        AND w was visited from non-transitive)
15      begin
16        label w as visited
17        // if w is visited from a non-transitive edge do not visit its children vertices

18        if w is not visited from non-transitive in this iteration
19          Q.enqueue(w)
20      end
21    end
22  end
```

When performing BFS traversal initiating from a specific license vertex *v* visiting all its adjacent vertices (line 12), if an adjacent vertex *w* is visited via a non-transitive edge, that vertex is added in the list of reachable vertices, but its children (i.e., vertices reached by its outgoing edges) are not (lines 17 to 19). This modification ensures that incompatible licenses are not reached (please refer to Definition 2). However, licenses may be reached in the graph via multiple paths, some of which may contain non-transitive edges but not all of them (e.g., MPL-2.0 license is reached from MIT/X11 via three different paths, one of which contains a non-transitive edge). In this case the path that contains only transitive (i.e., solid) edges is regarded valid and, hence, its children are considered in the set of reachable vertices to be examined (second part of the if statement of line 14 of the algorithm).

Based on the above we apply the following steps to determine the applicable license(s) among a specific set of licenses (info licenses in the SPDX terminology) encountered in a software package:

```
1   applicable_licenses = {}
2   for each vertex v_i in info_license_set
3   begin
4     // modified BFS traversal
5   reachable_licenses := {rv_i ∈ nodes of LICENSE_BFS(G,v_i)}
6     applicable_licenses = {rv_i} ∩ applicable_licenses
7   end
```

The worst case performance for modified BFS is $O(|E|)$ for each examined vertex, which approximates to $O(|V|)$ for the presented license graph, where the number of edges is comparable to the number of vertices. If BFS needs to be executed for all vertices in the license graph (i.e., a software package contains files with all possible licenses of the license graph), time complexity reaches $O(|V^2|)$.

4.3. Violation detection process

The SPDX violation check process depicted in Fig. 3 comprises of the following steps of the tool:

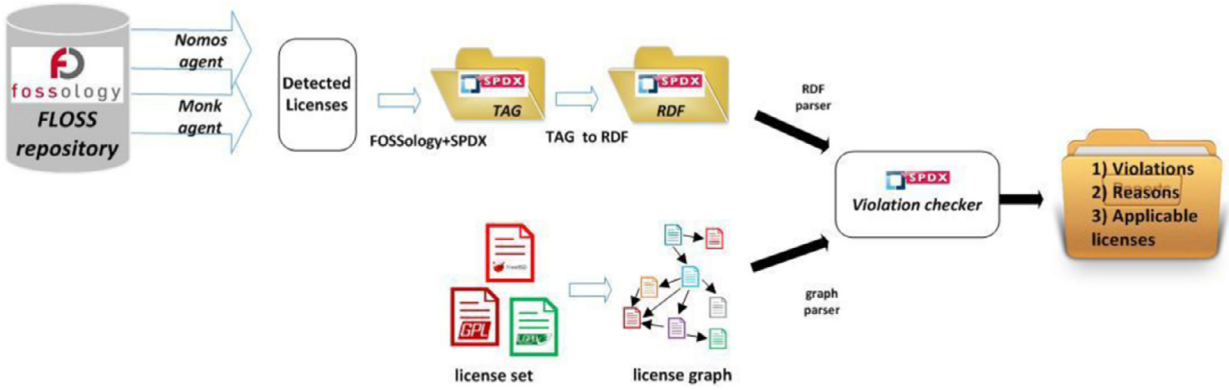


Fig. 3. Steps of the incompatibility detection process.

Table 2
Open source software projects in the testing set.

#	Software package	Version	Size	# Files	Programming language(s)	Official license
1	Anomos	0.9.5	824.2 KB	134	Python	GPL-3.0
2	AresGalaxy	2.3.0	1.06 MB	147	Delphi	GPL-2.0
3	CKEditor	4.4.7	1021.3 KB	275	JavaScript	GPL-3.0, LGPL-3.0, MPL-2.0
4	CuteFlow	2.11.2	3.81 MB	850	PHP	BSD 3-Clause
5	FileZilla (client)	3.11.0.2	4.22 MB	1275	C++, wxWidgets	GPL-2.0
6	Fraqtive	0.4.8	393.52 KB	199	C++	GPL-3.0
7	Hadoop (Apache)	2.7.0	17.26 MB	10,699	Java	Apache-2.0
8	HandBrake	0.10.1	9.96 MB	1254	C#, C, Objective C	GPL-2.0
9	Hunspell	1.3.3	962.97 KB	813	C++	GPL-2.0+, LGPL-2.1+, MPL-1.1+
10	Jexcelapi	2.6.12	1.82 MB	1268	Java	LGPL-3.0
11	Joda–Time	2.8	940.72 KB	455	Java	Apache-2.0
12	Mlpy	3.5.0	2.45 MB	181	Python	GPL-3.0
13	MPC-HC: Media player classic home cinema	1.7.9	13.35 MB	2874	C++	GPL-3.0
14	MrBayes	3.2.5	5.94 MB	37	C	GPL-3.0
15	Odoo	8.0	66.39 MB	13,040	Python, JavaScript, XML	AGPL-3.0
16	opencsv	3.4	303.41 KB	298	Java	Apache-2.0
17	Previsat	4.4.5	1.24 MB	155	C++	GPL-3.0
18	py2exe	0.6.9	146.18 KB	87	Python	MIT/X11, MPL-1.1
19	Shopware (Community edition)	5.0.1	34.31 MB	9241	PHP	AGPL-3.0
20	VirtualDub	1.10.4	1.82 MB	107	Assembly language, C++	GPL-2.0

- Verifies whether the correct declared license(s) is/are indicated for the software package in the SPDX file as indicated in the *licenseDeclared* field.
- If violations are found, it gives information on problematic license combinations that have caused this result.
- If no violations are found it proposes a number of alternate licenses that can be used for the package using the license information encountered in the different files of the package captured in the *licenseInfoFromFiles* fields of the SPDX file.

In the current version this does not include the proposal of multiple licenses (i.e., dual licensing or tri-licensing). Regarding the tool implementation the violation checker has been implemented in Java as a new tool to complement the available list of SPDX tools listed on the website of the specification. The violation checker is using the RDF versions of SPDX as input in order to guarantee the compliance with ontology standards.

This violation check can be used along with a license extraction tool that identifies licenses from software packages in source or binary format, such as the aforementioned FOSSology (Gobeille, 2008) or ASLA (Tuunamen, 2009). Although not described in the current paper, the above process of SPDX-VT makes it also feasible to examine whether two or more SPDX files can be combined in a new package, assuming that all separate files have been scanned initially for potential violations. Note that, although the latest version of SPDX is 2.0, in the tool version 1.2 is considered due to tool availability corresponding to SPDX and license identification.

5. Evaluation and discussion

For evaluation purposes we have selected a set of projects based on their popularity and the programming languages used. For the step of license identification of SPDX-VT we have used the Nomos and Monk license agents of FOSSology, whereas the FOSSology+SPDX tool was used for the SPDX file creation. The results are discussed viewed both from the presence of violations in software projects and from the accuracy of the adopted tools.

5.1. Projects dataset

For the testing set selection we have aimed at a fair representation of popular programming languages with the restriction that the applied licenses need to be covered by the license graph. The testing set presented here is larger than the initial set used in the previous work, where the initial version of the SPDX checker was presented (Kapitsaki and Kramer, 2015). The project characteristics are presented in Table 2. Many projects carry a GPL license, whereas some multi-licensing schemes also appear. Although SPDX has been adopted by some companies internally (e.g., Intel, Alcatel–Lucent) real SPDX files are not publicly available. For this reason the generation of these SPDX files from existing open source projects had to be performed.

Table 3
License mappings for FOSSology identifications with ambiguities.

License in FOSSology	Mapped license
MIT-style, X11-style	MIT/X11
BSD-style, BSD, ClearSilver	3-clause BSD
MPL	MPL-2.0
LGPL, GPL-2.0-with-classpath-exception	LGPL-3.0
GPL-2.0(+)-with-bison-exception	GPL-2.0(+)
GPL-style, GPL, GPL-exception, GPL-2.0(+)-with-autoconf-exception, GPL-3.0(+)-with-autoconf-exception[a]	GPL-3.0(+)
AGPL	AGPL-3.0

[a] http://www.gnu.org/licenses/exceptions.html

5.2. Experiments

We used the newest release of FOSSology (version 2.6.2, code revision 0a62dd) that includes 371 license artifacts in total. The FOSSology + SPDX tool would use the licenses found in the various files of the software package to set the declared package license (as a conjunctive license set) instead of using the license under which the software package was published by its creator. For this reason we have added the declared license information manually in the SPDX files as declared inside the package itself (e.g., in a LICENSE, COPYING or README file). Also the license list modelled in the graph was augmented by LGPL-2.0 and LGPL-2.0+, since the former license was encountered in three of the projects in the testing set. In essence LGPL-2.1 derived from LGPL-2.0 with minor changes.

In some cases licenses encountered by the license scanners of FOSSology state the license name without the license version or identify an indicative license that is similar to an existing license without finding the exact license text. GPL and MIT-style are examples of these cases. We handled these cases in the following way:

- When a specific license indication of the same type existed in the software package these abstract cases were neglected; for instance, for the GPL case, if GPL-2.0 was found in another file of the same software package, then the extraction of GPL by FOSSology was neglected and GPL-2.0 was used.
- When no similar license was found (with no version or with *-style* indication after the license name) the license was replaced by the "closest" license, i.e., the license of the newest version. For instance, LGPL would be substituted with version 3.0 of the LGPL.

The above along with other cases with ambiguities are shown in Table 3. GPL-2.0 with linking exception behaves similarly to LGPL; hence, these cases are mapped to the latest version of LGPL. Note that some other indications of FOSSology are neglected in order to make the analysis feasible: cases where no eligible license is found (i.e., *No_license_found, UnclassifiedLicense*), references to other licenses (i.e., *Same-license-as, See-doc(OTHER), See-file, Trademark-ref*) and references to organizations (i.e., *FSF*). Finally, cases that refer to restrictions but do not point to a respective license are neglected (e.g., *unRAR restriction*). We also neglected license detection possibilities (indicated with *–possibility*).

5.3. Results and discussion

Before providing the software packages of the testing set as input to the SPDX violation checker, a manual analysis was performed for specific licenses that are unclear in the results of FOSSology. This was performed for the following licenses:

- ***ClearSilver*** that is detected by the Monk agent of FOSSology for some packages (e.g., Hadoop, Fraqtive, MPC-HC and CuteFlow) is now licensed under the New BSD License and is hence regarded the same as 3-clause BSD license.
- ***Artistic license version 1.0*** appears in some projects (AresGalaxy and MPC-HC), but there has been a dispute on whether it can be regarded as a FOSS license (according to GNU "*we cannot say that this is a free software license because it is too vague*"[7]). For this reason it has been considered as incompatible to all other licenses. By analyzing manually the projects with Artistic-1.0 license, it was noted that the indication was false for AresGalaxy, but true for MPC-HC, where the exact text of the license appears in 18 files. For this purpose Artistic-1.0 license appears in parenthesis for AresGalaxy in Table 4. This notation with parenthesis is used also in other cases of false identifications by the license agents.
- Regarding CuteFlow, FOSSology indicates a ***W3C-possibility*** for one CSS (Cascading Style Sheets) file. By inspecting the file further, there is a reference to an external license file, where GPL-3.0 is indicated as license for that specific file.
- In CKEditor, there is a ***Microsoft-possibility*** in one JavaScript file, but a manual examination of the file shows that this is not accurate. This is also the case for HandBrake, where ***Sun-possibility*** is detected in one file and Previsat where both of the above license possibilities are indicated.

The analysis results of the SPDX files are depicted in Table 4, whereas Fig. 4(a) shows a summary of the results. Proposed licenses are the alternate licenses that can be applied on the package without causing violations, whereas the main violation reasons indicate why the declared license of the package is wrong (in case of violations).

The results indicate that a significant number of projects contain violations (11 out of 20 packages or 55%). The main violation is encountered in projects that contain the MPL-1.1 license that is incompatible to GPL. No applicable license can be proposed for projects containing both MPL-1.1 and any version of the GPL showing that the licenses from the independent packages used in the application cannot be combined to a new software product. In many cases the latest versions of GPL and AGPL are proposed as applicable licenses, since these are among the more restrictive FOSS licenses. We also observed that in some cases the license of the package was changed to a correct license in comparison to earlier versions. This is for instance the case for MrBayes that correctly carries GPL-3.0 in its latest version and was carrying GL-2.0 in version 3.2.2 that was causing violations.

As a general note the fact that many existing open source projects appear to have violations based on the analysis using the implemented tool is not necessarily generalizable. The SPDX input to the violation checker is based on the license identification by the agents of FOSSology that contain false positive cases. Indeed, this is the case for some projects since by analyzing the files manually we can see that the results are not valid for some cases (last column of Table 4). These are attributed mainly to false positives licenses by the Nomos agent. This is less apparent in the Monk agent, since it requires the exact license text for license matching.

Automated license detection is usually based on pattern matching algorithms. By their nature such heuristics are not capable of deriving ultimate answers to complex questions. They can only provide good guesses. From a legal perspective this is odd, because a single mistake or inconsistency among the files contained in a software package can render the entire package practically unusable. In the case of FOSSology the different scanning strategies of Nomos and Monk occasionally lead to seemingly diverting scan results. This relates to the standard pool of licenses and the difference in the scanning strategies of Nomos and Monk. Some examples of the above inconsistencies are visible in projects of the testing set:

[7] http://www.gnu.org/licenses/license-list.html#ArtisticLicense

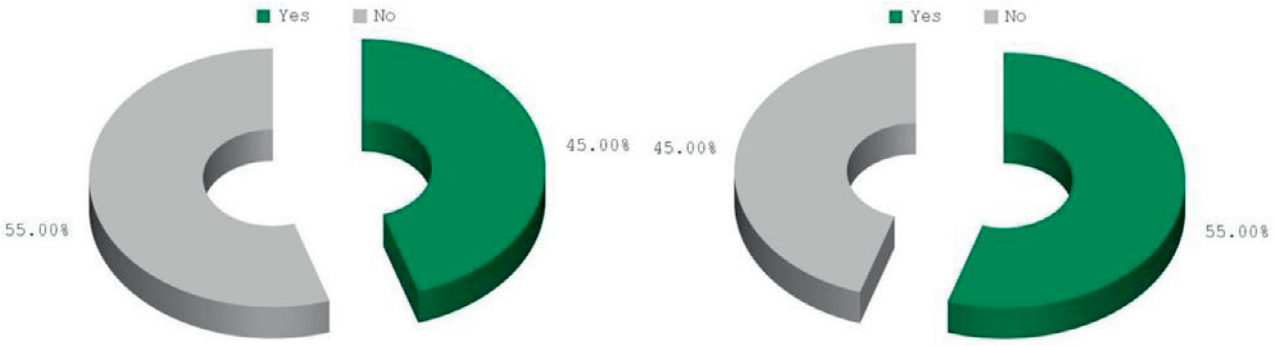
Table 4
Findings of SPDX-VT per software package.

#	Licenses by Nomos and Monk	License declared correct	Main violation reasons	Info licenses combinable	Proposed licenses	SPDX-VT result valid
1	GPL, GPL-3.0, GPL-3.0+, LGPL-3.0+, Public-domain-ref	√		√	GPL-3.0, GPL-3.0+, AGPL-3.0	√
2	(Artistic-1.0), BSD-3-Clause, GPL-2.0+, LGPL-2.1, MPL-1.1	×	License incompat. to GPL-2.0	× (MPL-1.1 incomp. to LGPL-2.1, GPL-2.0+)	*NONE*	√
3	GPL, GPL-2.0+, LGPL, LGPL-2.1+, Microsoft-possibility, MPL-1.1, MPL-1.1+, Trademark-ref	×	Licenses incompat.to official licenses	× (MPL-1.1 incomp. to LGPL, GPL)	*NONE*	√
4	BSD, BSD-3-Clause, ClearSilver, GPL, GPL-2.0, GPL-2.0+, GPL-3.0+, LGPL, LGPL-2.0+, LGPL-2.1+, MIT, MIT-style, W3C-possibility	×	Weak/strong copyleft licenses incompat. to permissive	× (GPL-2.0 incomp. to GPL-3.0+ and vice versa)	*NONE*	√
5	BSD-style, GPL-2.0, GPL-2.0+, GPL-2.0-with-autoconf-exception, GPL-3.0+-with-autoconf-exception, GPL-exception, MIT, MIT-style, Public-domain, X11, Zlib	×	Licenses incompat. to GPL-2.0	× (GPL-3.0+ incomp. to GPL-2.0 and vice versa)	*NONE*	√ (unless GPL-3.0 autoconf compat, to GPL-2.0)
6	BSD-3-Clause, ClearSilver, GPL-3.0, GPL-3.0+	√		√	GPL-3.0, GPL-3.0+, AGPL-3.0	√
7	Apache-2.0, Apache-possibility, BSD, BSD-2-Clause, BSD-3-Clause, BSD-style, ClearSilver, (GPL-2.0), (LGPL), MIT, MIT-possibility, MIT-style	× (√ with manual check)	Licenses incompat. to Apache-2.0	× (Apache-2.0 incompat. to GPL, LGPL)	*NONE*	× (false license detect.)
8	AGPL, BSD, BSD-3-Clause, FSF, GPL, GPL-2.0, GPL-2.0+, GPL-exception, LGPL-2.0+, LGPL-2.1+, MIT, MIT-style, MPL-1.1, (Sun-possibility), Trademark-ref	×	Licenses incompat. to GPL-2.0	× (MPL-1.1 incompat. to GPL-2.0, LGPL-2.1+)	*NONE*	√
9	BSD, BSD-2-Clause, BSD-possibility, BSD-style, FSF, GPL, GPL-2.0, GPL-2.0+, GPL-2.0-with-autoconf-exception, GPL-2.0+-with-bison-exception, GPL-exception, GPL-with-autoconf-exception, LGPL, LGPL-2.0+, LGPL-2.1, LGPL-2.1+, MIT, MPL, MPL-1.1, MPL-1.1+, Public-domain, Public-domain-ref, X11	× (√ with manual check)	Licenses incompat.to official licenses	× (MPL-1.1 incompat. to LGPL, GPL)	*NONE*	× (MPL1.1 part of tri-licens.)
10	BSD-style, LGPL, LGPL-2.1+	√		√	LGPL-3.0, LGPL-3.0+, GPL-3.0, GPL-3.0+, AGPL-3.0	√
11	Apache, Apache-2.0, Apache-possibility, BSD, Public-domain	√		√	Apache-2.0, AFL-3.0, MPL-2.0, LGPL-3.0, LGPL-3.0+, GPL-3.0+, GPL-3.0, AGPL-3.0, OSL-3.0	√
12	BSD-3-Clause, GPL-3.0, GPL-3.0+	√		√	GPL-3.0, GPL-3.0+, AGPL-3.0	√
13	Artistic-1.0, BSD, BSD-2-Clause, BSD-3-Clause, BSD-possibility, BSD-style, ClearSilver, GPL, GPL-2.0, GPL-2.0+, GPL-3.0, GPL-3.0+, LGPL-2.1, LGPL-2.1+, MIT, MIT-style, Public-domain, Public-domain-ref, Zlib, Zlib-possibility	×	Licenses incompat.to GPL-3.0	× (Presence of Artistic-1.0 and GPL-2.0 incompat. to GPL3.0 and vice versa)	*NONE*	√
14	GPL-2.0+, GPL-3.0, GPL-3.0+	√		√	GPL-3.0, GPL-3.0+, AGPL-3.0	√

(*continued on next page*)

Table 4 (*continued*)

#	Licenses by Nomos and Monk	License declared correct	Main violation reasons	Info licenses combinable	Proposed licenses	SPDX-VT result valid
15	Affero-possibility, AGPL, AGPL-3.0, AGPL-3.0+, Apache-2.0, BSD, BSD-2-Clause, BSD-3-Clause, BSD-possibility, CC-LGPL-2.1, ClearSilver, GPL, GPL-2.0, GPL-2.0+, GPL-3.0, GPL-3.0+, GPL-possibility, LGPL, LGPL-2.1, LGPL-2.1+, LGPL-3.0, MIT, MIT-possibility, MIT-style, MPL-1.1, MPL-1.1+, OFL-1.1, Public-domain, Public-domain-ref, Python, Trademark-ref	×	Licenses incompat.toAGPL-3.0	× (MPL-1.1 incompat. to LGPL/GPL/AGPL, and GPL-2.0 incompat. to GPL-3.0 and vice versa)	*NONE*	√
16	Apache-2.0, BSD-3-Clause	√		√	Apache-2.0, AFL-3.0, MPL-2.0, LGPL-3.0, LGPL-3.0+, GPL-3.0, GPL-3.0+, AGPL-3.0, OSL-3.0.	√
17	GPL-3.0+, Microsoft-possibility, Sun-possibility	√		√	GPL-3.0, GPL-3.0+, AGPL-3.0	√
18	LGPL, MIT-style, MPL, MPL-1.1	×	Licenses incompat.to official licenses	× (MPL-1.1 incompat. to LGPL and vice versa)	*NONE*	√
19	AGPL-3.0, MIT, MIT-style, LGPL, PHP-possibility, Trademark-ref, Apache-2.0, MIT-possibility, BSD-possibility, AGPL, BSD, GPL, LGPL-2.1, GPL-2.0, See-URL, LGPL-2.1+,GPL-3.0, LGPL-3.0+, LGPL-3.0	×	Licenses incompat.to AGPL-3.0	× (Apache-2.0 incompat. to GPL-2.0, LGPL-2.1, LGPL-2.1+)	*NONE*	√
20	GPL-2.0+, GPL-2.0	√		√	GPL-2.0	√



(a) by SPDX-VT **(b) with manual check**

Fig. 4. Percentages of license violations found in the testing set. (a) by SPDX-VT. (b) with manual check.

- In *CKEditor*, MPL-1.1 license is indicated as license for the software product and this is correctly identified by FOSSology, although MPL-1.1+ is meant. The respective indications need to be corrected by the package owners.
- *CuteFlow* carries a permissive license (BSD 3-Clause), although its files contain licenses from the copyleft groups. For instance, the presence of GPL-3.0+ is indicated in a large number of files as detected by FOSSology (135 files). By manually inspecting some of the files it is observed that indeed this is the license referred in the file.
- In *FileZilla* GPL-3.0 or higher indication is given in some files recognized by FOSSology correctly as with Autoconf exception. If this license can be regarded compatible to GPL-2.0 then the package can be regarded as violation-free.
- In *Hadoop* violations are detected due to the presence of LGPL in one package file and GPL-2.0 in two files. LGPL is in that case incorrectly identified by FOSSology, whereas GPL-2.0 is correctly identified in files that provide dual licensing and the possibility of licensing under a permissive license. Hence, the results are attributed to false positives by FOSSology.
- In *HandBrake* violations appear due to MPL-1.1 and AGPL licenses. Although AGPL is a false positive of FOSSology, MPL-1.1 is referred to in one project file causing the violation.
- *Hunspell* carries a triple license, but the identification of MPL-1.1 by the license scanners causes violations. Since the file of

this detection is where the triple license is indicated and the same seems to be the case for GPL and LGPL detections, this is a case where the lack of multi-licensing scheme identifications by license scanners can cause false results.

- In the first experiment by *Joda-Time* it was observed that Apache-1.0 license was added in the SPDX file generated by FOSSology+SPDX, although this license is not identified by FOSSology. For this reason our tool was detecting a false violation. By removing reference to Apache-1.0 license no violation is detected.
- In the case of *MPC-HC* violations are detected due to the following licenses encountered in the package distribution: Artistic-1.0 and GPL-2.0. Although the identification of GPL-2.0 in two files by FOSSology is not completely accurate, since GPL-2.0+ is also detected in the same files, the presence of Artistic-1.0 cannot be doubted.
- In *py2exe* violations are detected due to the presence of LGPL license in one file of the package. The LGPL license is correctly identified by FOSSology.
- In *Shopware* no applicable license can be found to fix existing violations, due to the presence of Apache-2.0 and GPL-2.0 that are incompatible with other licenses of the package. By manually checking the files it can be seen that indeed these licenses are indicated.
- A similar problem appears in *Odoo* due to the presence of the above licenses and also MPL-1.1, although some other licenses not included in the license graph are also detected: e.g., OFL that would not cause problems to the combination with GPL[8]. Regarding the presence of the python license that does not appear in the license graph only some of its versions are compatible to GPL. CC-LGPL that appears on the license list refers to Creative Commons (CC) and is for that purpose neglected by SPDX-VT.

By taking into consideration the above cases of false indications by the SPDX-VT caused by incorrect license detections, the number of packages with violations decreases to 45% (Fig. 4(b)).

An additional disadvantage of using a separate license extraction tool like FOSSology is that it is not possible to know whether a license in the package is mentioned by FOSSology because there is a third party library used in the package that carries this license or because this is the license the creators of the package have decided to apply on the whole package. The above problem appeared for instance in the CKEditor project that carries the tri-license MPL-1.1, LGPL-3.0, GPL-3.0. All these licenses were identified in different files by FOSSology. The above observations are attributed to the fact that an external license identification tool needs to be employed and are also analyzed in the next case studies.

6. Case studies

In the SPDX-VT results some cases of false positives or cases of not so linear license identification by the agents have been detected. In order to better understand the presence of such cases and the role of the quality of the automated scanning results, we conducted an in-depth analysis of three selected FOSS projects from the testing set that are very prominent in their applications fields: 1) the eCommerce solution Shopware, 2) the Enterprise Resource Planning (ERP) system Odoo, and 3) the distributed Big Data analytics tool Apache Hadoop. In case of Apache Hadoop and Shopware the stable versions of the software from the download pages of the projects were used, whereas for Odoo a nightly build of version 8.0 was downloaded.

6.1. Shopware

Shopware is a modular eCommerce solution that is developed based on the Zend and the Symfony PHP frameworks. Shopware is offered in an ecosystem along with various open source and closed source software packages. The open source market leader is Magento (by Ebay Inc.) with a market share of 2.9%. Another prominent FOSS package is PrestaShop with a market share of roughly 1.3%[9]. Shopware that has a market share of 0.1% comes with a dual licensing option (a FOSS and a commercial license). Regarding the FOSS option the community version of the package is licensed under the AGPL-3.0, whereas the templates are licensed under the BSD-3-clause license (both claimed licenses). The reason why we selected this package for an in-depth analysis instead of Magento or PrestaShop is that we wanted to derive meaningful results for the software package via SPDX-VT using the graph with license compatibilities (for instance Magento contains many FOSS licenses that are less popular).

The in-depth analysis of Shopware shows that the license scanning engines of FOSSology (i.e., Nomos, Monk) are not yet capable to derive logically coherent conclusions about the real license. For example, no license has been identified for 4997 files. Some of the files are binary content but the headers point to a central licensing file claiming the MIT license for the respective file. Currently this can only be determined with manual processing. Artificial intelligence and learning algorithms might be useful tools for such cases in future research.

Table 5 shows a subset of the 30 determined classifications. Nomos for example determines 7 files to be GPL-licensed. Reviewing this finding deeper shows that 6 out of those 7 files are dual-licensed including one of the already determined licenses, whereas one file is clearly and solely GPL-licensed. Since that file does not belong to the software package editors' intellectual property (IP), it should hinder the editor to assign a different, i.e., commercial license, to the software package. Shopware contains licenses from the whole range of permissive to strong copyleft licenses. The results of SDPX-VT also demonstrate incompatibilities among the licenses in Odoo. Due to this small coherence asserting a commercial license to the entire package, as the vendor does, seems questionable.

6.2. Odoo

Odoo formerly known as OpenERP is the leading open source Enterprise Resource Planning (ERP) solution. Odoo has over 2 million users and over 500 integration partners worldwide. It is based on a modular framework that consists of around 260 modules covering the basic ERP functions, such as accounting, material resource planning, human resources, marketing, procurement, sales and distribution as well as warehouse management. Since version 8.0 it is the world's first ERP system that natively integrates content management and eCommerce.

Although Odoo is the most prominent FOSS ERP software package, the editor as well as the community do not seem to put too much attention to licensing. Table 6 shows a subset of the 41 identified licenses; nevertheless, a large portion of the files in Odoo (e.g., the translation files) just refer to the license of the parent package, instead of having their own license header attached. The same applies to 4864 files that contain example data, system configurations and similar important content. These files do not even have reference to any license text. In addition, FOSSology tries to bind binary content to a license which seems senseless to a large

[8] http://www.gnu.org/licenses/license-list.html#SILOFL

[9] See market share estimates under: http://w3techs.com/technologies/overview/content_management/all

Table 5
In-depth analysis of Shopware version 5.0.1.

Number of classified files	Classification	File types in sample	Qualitative interpretation	Classification after qualitative review
4997	*No license found*	php, png, php, xml, htm, tpl, css, js, ini, gif, rb, less	Belongs to the Symfony PHP framework licensed under MIT.	MIT
2370	*Dual license*	php, js, scss	Contains a special header for dual licensing.	AGPL-3.0
2369	*AGPL*	php, js, scss	Mostly the same contents as above.	AGP-3.0
983	*See Doc*	php	Belongs to the Zend PHP Framework licensed under BSD 3-Clause.	BSD 3-Clause
779	*MIT*	php	Belongs to the Doctrine ORM Mapper.	MIT
604	*MIT-Style*	php	Mostly the same contents as above. Belongs to the Doctrine ORM Mapper.	MIT
85	*LGPL*	js,php	Belongs to TinyMCE WYSIWYG Editor.	LGPL-2.1
7	*GPL*	png, js, css	For example Shopware-5.0.1/engine/ Library/Mpdf/mpdf.php is ONLY GPL licensed.	GPL

Table 6
In-depth analysis of Odoo version 8.0.

Number of classified files	Classification	File types in sample	Qualitative interpretation	Classification after qualitative review
6696	*Same-license-as*	po	Odoo is localized in various languages. These files belong to the translation and are licensed under the same license as the major package.	AGPL-3.0
4864	*No_license_found*	yml, jpg, jpeg, rst, xml, csv	Mostly configuration and example data as well as binary content (jpg, jpeg).	No explicit license given
1141	*AGPL-3.0+*	py	Main code base of the product. Contains application logic.	AGPL-3.0
89	*Python*	po	Wrongly detected as *Python* because of keyword pattern match of Nomos. These files belong to the translation and are licensed under the same license as the major package.	AGPL-3.0
76	*AGPL*	po	Also these files belong to the translation and are licensed under the same license as the major package.	AGPL-3.0
53	*GPLv3*	py, po	The po files belong to the translation. In the source code there are remains from formerly GPL-3.0 licensed code base.	po files AGPL-3.0, py files GPL-3.0
49	*MIT*	js, css	Belong to major JavaScript libraries such as JQuery, Bootstrap and Underscore.	MIT
18	*GPL*	txt, py, po, jpg	po files are AGPL-3.0 licensed, txt and jpg content are not compiled. Files are mostly dual-licensed except two that are solely GPL.	GPL

extent. This should be removed from the scan results before generating the SPDX file of the search results.

Whereas most of the business logic of Odoo that resides in Python files (file type .py) is AGPL-3.0 based and contains an appropriate license text, there are still a few files that contain GPL-based code. This GPL-based code remains from a former license change from GPL-2.0 to AGPL-3.0 that the editor of Odoo, OpenERP S.A., undertook in October 2009. These observations indicate that not sufficient emphasis has been put on license handling and coherence by Odoo. The above are also in alignment with the results of SPDX-VT from the analysis of Odoo (e.g., the presence of MPL-1.0 license in one file).

6.3. Hadoop

Apache Hadoop has been recognized as one of the most vibrant FOSS packages in the field of Big Data. It is a Java-based framework for distributed and scalable software that implements two technology blueprints initially published by Google Inc.: the MapReduce algorithm and an own large scale distributed file system, the Hadoop Distributed Files System (HDFS) (Dean and Ghemawat, 2008; Chang et al., 2006). Apache Hadoop is a top level project in the Apache Foundation and draws on a large community of contributors and users such as LinkedIn, Facebook and Alibaba. Hadoop has turned into a whole ecosystem that inspired a diverse set of add-ons such as Pig, Hive, Hbase, Mahout and Zookeeper that builds the de facto standard for Big Data applications. Large vendors such as IBM and EMC integrated Hadoop into their product portfolio. Hortonworks and Cloudera provide packaged enterprise solutions with a focus on Apache Hadoop.

The results from our in-depth analysis of Hadoop (Table 7) support our initial hypothesis that an FOSS package hosted by Apache would show a higher perceived coherence according to the licenses used. The results of FOSSology show that only 14 different license types are detected, whereas Odoo and Shopware contained 41 and 30 licenses types respectively. Furthermore, the variety of licenses is significantly smaller for Hadoop. Hadoop only uses permissive non-copyleft licenses such as Apache-2.0, BSD-new and MIT but no weak or strong copyleft licenses. Two files with GPL-2.0 appearance belong to JQuery. These files are dual-licensed and provide explicitly the possibility to be licensed under a permissive license such as MIT or BSD-3-clause as indicated also in the SPDX-VT results in Section 5.3.

Table 7
In-depth analysis of Apache Hadoop version 2.7.0.

Number of classified files	Classification	File types in sample	Qualitative interpretation	Classification after qualitative review
7158	*Apache-2.0*	Java	Contains application logic of the Hadoop platform. Every Java file contains an appropriate license header.	Apache-2.0
3612	*No_license_found*	png, gif, json, meta	Contains binary content and test data.	No license applicable
13	*BSD-3-Clause*	java, js, txt	Are dual-licensed as Apache and BSD licenses. Files are the result of an EU-funded project (ONELAB - Open Numerical Engineering LABoratory).	Apache-2.0
6	*ClearSilver*	C header and class files (c,h)	Belong to LZ4 compression algorithm (licensed under BSD).	BSD-3-Clause
5	*BSD-2-Clause*	Same as above	Same files as above but identified by Nomos.	BSD-3-Clause
3	*MIT*	css, js	Belong to Jquery and Bootstrap JavaScript libraries.	MIT
2	*GPL-2.0*	Js	Belong to JQuery but are dual-licensed also under MIT or BSD.	BSD-3-Clause MIT

7. Conclusions

In this paper we have presented our work on license violation analysis on SPDX files. The proposed license compatibility process is able to identify whether violations exist in the package description and give suggestions for applicable licenses, if the licenses present in the package can be combined in a single distribution. The suggestions are based on the license graph that contains compatibility information for commonly used open source licenses. We have shown that the results are highly dependent on license identification tools that perform the license detection before the main tool is applied on the SPDX files. FOSSology provides a good first insight into the internal licensing status of an open source software package. However, due to some limitations of the license scanners (i.e., Nomos and Monk), it is not yet capable to derive logically coherent, concise scan results. FOSSology does not provide currently a way to improve the coherence of its results. Nevertheless, our contribution constitutes important progress towards the automation on license checks and decisions for software systems in the standardized setting of SPDX. We hope that it will trigger more research on license compatibility and further promote the use of SPDX, as well as contribute to the tools accompanying the specification.

As future work it is also interesting to study license compatibility through a theoretical framework and mathematical logic as performed by Governatori et al. (2014) for licenses associated with datasets. The above will be combined with a more detailed representation of the rights and obligations of the licenses that are now not captured in the license graph. We also envision bringing the tool at a mature state to form part of the SPDX tool set and its latest SPDX 2.0 specification, when the accompanying tools are also made available (e.g., support for SPDX 2.0 by FOSSology+SPDX). A semi-automatic tool-support for deriving a license graph based on specific conditions set for restrictions on FOSS use is a very interesting problem. Although a very useful task, the richness of the legal license text renders it almost unrealistic as discussed in this work.

Appendix: explanations on license graph edges

This Appendix justifies license graph edges for licenses that do not appear in Kapitsaki et al., 2015.

#	From	Connection	To	Reason
a	Apache-2.0	compatible	AFL-3.0	AFL-2.0 is embodying many of the same provisions found in the MIT, BSD and Apache licenses; in addition, it includes certain clauses addressing the application of patent rights to open source software. Since AFL-2.0 is compatible with AFL-2.1 (the latter just rephrases clause 10) and AFL-2.1 is compatible with AFL-3.0 (the most notably differentiation in the latter comes in clause 5, i.e., External Deployment, in which the definition of “distribution" explicitly includes communicating the work as “an application intended for use over a network."), AFL-3.0 is also regarded compatible to Apache-2.0.
	Source: http://opensource.org/proliferation-report and Laurent (2004) Chapter 02			
b	AFL-3.0	compatible	OSL-3.0	OSL-3.0 is a reciprocal license. AFL-3.0 is identical to OSL 3.0 except for the reciprocal source code obligation (clause 1c of AFL-3.0).
	Source: Rosen (2004) Chapter 09 and Alspaugh (2009)			
c	MPL-1.1(+)	compatible	CDDL-1.0	CDDL-1.0 is based on MPL-1.1.
	Source: HYPERLINK " https://docs.oracle.com/cd/E19450-01/820-6173/def-common-development-and-distribution-license.html			

References

Alspaugh, T.A., Scacchi, W., Asuncion, H.U., 2010. Software licenses in context: The challenge of heterogeneously-licensed systems. J. Assoc. Inf. Syst. 1 (11/12), 730–755.

Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M., Gousios, G., 2011. Open source software: a survey from 10,000 feet. Found. Trends Technol. Inf. OM. 4 (3-4), 187–347.

Ausiello, G., Franciosa, P.G., Italiano, G.F., & Ribichini, A. 2015. Incremental DFS trees on arbitrary directed graphs. arXiv preprint arXiv: 1502.07206.

Boyle, J., 2009. The Public Domain: Enclosing the Commons of the Mind. Yale University Press, pp. 255–257.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W.C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R.E., 2006. Bigtable: A distributed storage system for structured data. ACM Trans. Comput. Syst. (TOCS) 26 (2), 4.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51 (1), 107–113.

Di Penta, M., German, D.M., Guéhéneuc, Y.-G., Antoniol, G., 2010. An exploratory study of the evolution of software licensing. In: Proceedings of 32nd ACM/IEEE International Conference on Software Engineering. ACM, pp. 145–154.

Floyd, R.W., 1962. Algorithm 97: shortest path. Commun. ACM 5 (6), 345.

Foukarakis, I.E., Kapitsaki, G.M., Tselikas, N.D., 2012. Choosing licenses in free open source software. In: Proceedings of 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2012), pp. 200–204.

Gangadharan, G.R., D'andrea, V., De Paoli, S., Weiss, M., 2012. Managing license compliance in free and open source software development. Inf. Syst. Frontiers 14 (2), 143–154.

German, D.M., Manabe, Y., Inoue, K., 2010. A sentence-matching method for automatic license identification of source code files. In: Proceedings of IEEE/ACM International Conference on Automated Software Engineering. ACM Press, pp. 437–446.
German, D.M., Di Penta, M., 2012. A method for open source license compliance of java applications. IEEE Soft. 29 (3), 58–63.
Gobeille, R., 2008. The FOSSology project. In: Proceedings of 2008 International Working Conference On Mining Software Repositories. ACM Press, pp. 47–50.
Gordon, T.F., 2010. Report on a prototype decision support system for OSS license compatibility issues. Qualipso (IST- FP6-IP-034763), Deliverable A1.D2. 1, 3.
Gordon, T.F., 2011. Analyzing open source license compatibility issues with Carneades. In: Proceedings of the 13th International Conference on Artificial Intelligence and Law (ICAIL '11). ACM, pp. 51–55.
Governatori, G., Lam, H.P., Rotolo, A., Villata, S., Atemezing, G., Gandon, F., 2014. Checking licenses compatibility between vocabularies and data. In: Proceedings of the 5th International Workshop on Consuming Linked Data (COLD 2014).
Kapitsaki, G.M., Kramer, F., 2015. Open source license violation check for SPDX files. In: Proceedings Software Reuse for Dynamic Systems in the Cloud and Beyond. Springer International Publishing, pp. 90–105.
Kapitsaki, G.M., Tselikas, N.D., Foukarakis, I.E., 2015. An insight into license tools for open source software systems. J. Syst. Softw. 102, 72–87.
Laurent, A.M.S., 2004. Understanding Open Source And Free Software Licensing. O'Reilly Media, Inc.
Lee, C.Y., 1961. An algorithm for path connections and its applications. IRE Trans. Electron. Comput. 3, 346–365.
Lindberg, V., 2008. Intellectual Property and Open Source a Practical Guide to Protecting Code. O'Reilly Media.
Linux Foundation and its Contributors, 2015. A common software package data exchange format, version 2.0. https://spdx.org/sites/spdx/files/SPDX-2.0.pdf [Last accessed: June 13th, 2015]
Lokhman, A., Luoto, A., Abdul-Rahman, S., Hammouda, I., 2012. OSSLI: Architecture level management of open source software legality concerns. Open Source Syst. 356–361.
Mancinelli, F., Boender, J., Di Cosmo, R., Vouillon, J., Durak, B., Leroy, X., Treinen, R., 2006. Managing the complexity of large free and open source package based software distributions. In: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering (ASE 2006). IEEE, pp. 199–208.
Rosen, L., 2004. Open Source Licensing: Software Freedom and Intellectual Property Law. Prentice Hall PTR.
Sojer, M., Henkel, J., 2010. Code reuse in open source software development: Quantitative evidence, drivers, and impediments. J. Assoc. Inf. Syst. 11.12, 868–901.
Tuunamen, T., Koskinen, J., Karkkaiken, T., 2009. Automated software license analysis. Autom. Soft. Eng. 16 (3-4), 455–490.
Wheeler, D.A., 2007. The free-libre / open source software (FLOSS) license slide, http://www.dwheeler.com/essays/floss-license-slide.pdf [Last accessed: June 13th, 2015]
Xu, H., Yang, H., Wan, D., Wan, J., 2010. The design and implement of open source license tracking system. In: Proceedings of the International Conference on Computational Intelligence and Software Engineering, pp. 1–4.

Dr.-Ing. Georgia M. Kapitsaki is an Assistant Professor at the Department of Computer Science in the University of Cyprus (UCY). Her research interests include service-oriented computing, privacy protection, context-aware services and open source software reuse. She received her Dipl.-Ing. degree in 2005, her M.Sc. degree in techno-economics in 2008 and her PhD in 2009, all from the School of Electrical and Computer Engineering of National Technical University of Athens (NTUA). She has published several works in conferences and journals and has participated in the organization of international conferences (e.g., ICWS 2014, WISE 2013).

Mr. Frederik Kramer is a doctoral candidate at the Magdeburg Research and Competence Cluster (MRCC) of Otto von Guericke University Magdeburg and CEO of initOS Gmbh & Co. KG. He studied Business Informatics at the Institute of Technical and Business Information Systems at the Otto von Guericke University Magdeburg. He is a serial entrepreneur who started his first company in 1997 during the early days of the Internet revolution. The topic of his doctorate is strategic engineering of system landscapes for SME. His primary research interest lies in the strategic application of Open Source Software and landscape architecture for SMEs.

Dr.-Ing. Nikolas D. Tselikas is a tenure-track Assistant Professor at the Department of Informatics and Telecommunications in the University of Peloponnese (UoP). He received both his Dipl.-Ing. and PhD from the School of Electrical and Computer Engineering of National Technical University of Athens in 1999 and 2004, respectively. His research interests include services and networking, middleware and software engineering, location-based services, mobile applications and open source software. He has been involved in several research projects on the above areas and has published more than 50 papers in international journals, conferences and book chapters and two books on C programming language.