Validate Your SPDX Files for Open Source License Violations

Demetris Paschalides
University of Cyprus
1, University Avenue
2109, Aglantzia, Cyprus
dpasch01@cs.ucy.ac.cy

Georgia M. Kapitsaki
University of Cyprus
1, University Avenue
2109, Aglantzia, Cyprus
gkapi@cs.ucy.ac.cy

ABSTRACT

Licensing decisions for new Open Source Software are not always straightforward. However, the license that accompanies the software is important as it largely affects its subsequent distribution and reuse. License information for software products is captured - among other data - in the Software Package Data Exchange (SPDX) files. The SPDX specification is gaining popularity in the software industry and has been adopted by many organizations internally. In this demonstration paper, we present our tool for the validation of SPDX files regarding proper license use. Software packages described in SPDX format are examined in order to detect license violations that may occur when a product combines different software sources that carry different and potentially contradicting licenses. The SPDX License Validation Tool (SLVT) gives the opportunity to check the compatibility of one or more SPDX files. The evaluation performed on a number of software packages demonstrates its usefulness for drawing conclusions on license use, revealing violations in some of the test projects.

CCS Concepts

•**Software and its engineering** → **Reusability; Open source model;** Software libraries and repositories;

Keywords

open source software; licensing; Software Package Data Exchange

1. INTRODUCTION

Free Open Source Software (FOSS) is nowadays widely used for the creation of large software systems. Existing software components are integrated in new implementations leading to derivative works of existing products, i.e., works adapted from the originally copyrighted item [11]. These reusable software components often carry different licenses with different terms in respect to the software use and distribution [14]. Licenses provide the terms under which the use of the intellectual property of the licensor, i.e., the software creator, is feasible. Choosing the correct license to apply on a new software system is not a trivial task due to the differences in license terms and it is affected both by the intentions of use for the software under construction and the integration of third party software that may carry specific - and sometimes conflicting - licenses [2].

Unfortunately cases of incorect license use can be encountered and these may lead to legal conflicts [1]. Users may opt for using popular licenses (e.g., Apache license, MIT license) instead of the correct license that does not violate the terms of the license of adopted software. They may also not consider properly the three main license categories of permissive, weak- and strong-copyleft [6]. Therefore, the final license(s) used on a product is(are) not allowed to conflict with the licenses of existing third party software that forms part of that product, e.g., code from third party software may have been included statically or be linked dynamically to the product. If we consider, for instance, a software product that carries an Apache version 2.0 license (Apache-2.0), but integrates software that uses the GPL version 2.0 license (GPL-2.0), a violation exists, since these licenses are incompatible: software that carries the GPL-2.0 license cannot be combined with software that carries the Apache-2.0 license and be released under that license, and vice versa.

At the same time there is an emerging trend by many organizations to adopt the *Software Package Data Exchange (SPDX)* format for describing their software resources [12]. SPDX is *a standard format for communicating the components, licenses, and copyrights associated with a software package* [3]. License information is an important aspect of SPDX. SPDX includes information for all software licenses used in the software product covering multi-licensing schemes and licenses from third party software. Although SPDX is currently gaining importance, its adoption is performed mainly on an internal level. Wind River is one of the early adopters and provides also a service that creates SPDX files from uploaded packages[1]. Other organizations that use SPDX internally can be found in Alcatel-Lucent, Texas Instruments, Samsung and the Yocto project.

On the one hand, the creation of valid SPDX files in terms of licensing is important since they will accompany each software component forming part of its metadata. On the other hand, the SPDX files received through such communications need to be used correctly to avoid violations in cases of

FSE'16, November 13–18, 2016, Seattle, WA, USA
© 2016 ACM. 978-1-4503-4218-6/16/11...$15.00
http://dx.doi.org/10.1145/2950290.2983939

[1]http://spdx.windriver.com/pkg_upload.aspx

software combinations. Based on the above, the availability of appropriate mechanisms and tools that assist the use of SPDX in the framework of license compliance is vital. In this demonstration paper we present the *SPDX License Validation Tool* (SLVT[2]) that examines license violations that may exist in a single or multiple SPDX files. The *SPDX License Validation Tool* bases the license violation process on the directed acyclic license graph presented in previous works of the authors [7, 8]. The graph has been expanded from its original version and may be further expanded by SLVT users. The contributions of the *SPDX License Validation* online *Tool* are the following: 1) we provide an online tool for the analysis of a single SPDX file for violation purposes and the examination of the combination of SPDX files into a single software product, 2) we provide a visualization of the license compatibility graph [7, 8] that can be expanded by SLVT users, and 3) we expose a RESTful API for the license violation analysis and the graph expansion services that can be used for integrating license violation examination into new systems. A video presenting the functionalities of SLVT is available online[3].

2. BACKGROUND AND RELATED WORK

License compatibilities are targeted in previous works, but no existing tool provides an automated approach for validating license combinations. Some previous approaches focus on modeling license compatibilities via a graph that captures main open source licenses [2]. The license slide of Wheeler [15] presents compatibilities among the following licenses: MIT/X11 license, BSD-new, Apache-2.0, LGPL-2.1 and LGPL-2.1+, LGPL-3.0 and LGPL-3.0+, MPL-1.1, GPL-2.0 and GPL-2.0+, GPL-3.0 and GPL-3.0+, and AGPL-3.0. The more recent license graph captures a total of 22 licenses [7]. As aforementioned we have adopted the latter graph in this work enriched with additional licenses. Note that the license name and version notation used throughout this paper follows the SPDX notation.

Other works provide more detail on the license structure referring to rights and obligations present in the license text. In Qualipso [5] an ontology for the modeling of software systems using the Web Ontology Language (OWL) contains 8 licenses [13]. A more recent work uses license rules expressed in a legal rule language in XML (Extensible Markup Language) format applying them to the model of a software system in order to detect possible licenses automatically through the Carneades argumentation system [6]. This latter approach examines license rights in order to suggest an applicable license for the system, although no specific license is recommended for use at the end of the process.

In the framework of the SPDX Workgroup and its community, some tools that integrate SPDX with other systems have been proposed, although none of them addresses license compatibilities. An example can be found in the integration with FOSSology resulting to the *FOSSology+SPDX* tool. FOSSology is a license identification tool that can extract different licenses contained in textual files (e.g., source code, configuration files) [4]. It relies on an heuristic pattern-matching algorithm expressed by regular expressions. For an overview of further related work on license tools available including license identification the reader is referred to [8].

[2]http://spdxcompatibility-licensetools.rhcloud.com/
[3]https://youtu.be/9fgRP24JdKs

In this work we evolve the previous work of license violations in the framework of SPDX [7] by performing validations on the content of SPDX, considering additionally more than one SPDX files providing a relevant tool that can be integrated in new systems. To the best of our knowledge, there is no online system that provides the opportunity to perform examination of software products for license violations.

3. SPDX LICENSE VALIDATION TOOL

The *SPDX License Validation Tool* offers a number of options for examining SPDX files on license use. Although the aim is the handling of license compatibilities, the process is applied on SPDX-structured files, since SPDX is the only option for formalizing the description of licenses used in a software system. The proposed tool is operating in the following way as depicted in the process of Figure 1:

1. asks the user to upload one or more SPDX files,
2. performs validation on the structure of each input,
3. examines the compatibility of the licenses of the software package considering the licenses used in each file of the package independently (i.e., *extracted licenses*),
4. in case of more than one SPDX files as input, SLVT proceeds with analyzing the combination of the files,
5. in case of violations but with the possibility of correction(s) to the SPDX file(s) (in the *declared license*), valid SPDX file(s) created by SLVT can be downloaded by the user.

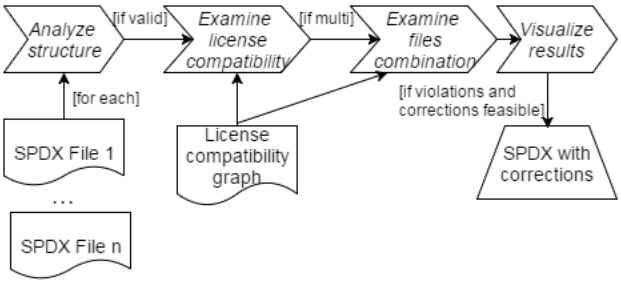

Figure 1: Analysis steps in the SPDX License Validation Tool.

SPDX files follow the RDF (Resource Description Framework) format [9]. The aforementioned *declared license* refers to the product license indicated by the authors of the package and is captured in a respective element, for example:

<licenseDeclared rdf:resource=
"http://spdx.org/licenses/Apache-2.0" />

The file contains also a number of *extracted licenses* that can be found in files and libraries that comprise the main software product. This latter category is indicated in SPDX in *licenseInfoFromFiles* elements.

SLVT uses the license compatibility graph mentioned in the previous section. License compatibility analysis depends on iterating over the license graph to detect licenses that can be combined without causing violations. Specifically, the adopted algorithm applies a modified version of Breadth First Search (BFS) [10]. The structure of the license graph along with the previous version of the approach has been

introduced in a previous publication that adopts a variant of the Floyd-Warshall algorithm [7]. The modified BFS has replaced the variant of the Floyd-Warshall algorithm, as it can be faster in some cases. The traversal algorithm considers two type of edges that can be found in the graph to relate licenses. A *transitive edge* indicates that the adjacent vertices are linked with a transitive relation, i.e., if license A is compatible to license B and license B is compatible to license C, then transitive edges between the vertices indicate also that A is compatible to C. This relation does not apply for *non-transitive edges*.

In short, the modified BFS does not consider non-transitive edges: when performing BFS traversal initiating from a specific license vertex, if a vertex of the graph 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. This modification ensures that incompatible licenses are not reached. If however, that same vertex (reached through the non-transitive edge) is also reachable through another transitive path its children are considered in the set of reachable vertices to be examined:

SLVT applies the modified BFS algorithm on each SPDX file by taking into account each *extracted license* from the different files of the software. This way SLVT determines a set of applicable license(s) among the extracted licenses, i.e., licenses that can be applied on the software package without causing violations. Based on this information SLVT examines whether the declared license(s) of the software package (as indicated by its creators) is(/are) contained in the applicable license(s) set.

Using the above procedure if the result of the compatibility analysis fails (i.e., violations are found), SLVT determines whether new declared licenses that can resolve these violations can be used (i.e., so that all licenses in the software product are compatible). The process performs an iteration over all possible paths having as starting point each extracted license of the SPDX file, and if a license compatible with all other licenses is detected, then this license is considered an appropriate suggestion. The license(s) suggested can be used to replace the declared license of the SPDX file.

Note that SLVT requires the use of SPDX files that can be generated manually or automatically. For the latter case support is provided by the aforementioned FOSSology that offers two agents for license extraction purpose: the Nomos and the newer Monk agent. The *FOSSology+SPDX* tool of the SPDX community offers the possibility to create SPDX files (in tag format) from the licenses extracted from a software product. These tag files can be subsequently converted to RDF format for processing by SLVT. Similar functionality is provided by the tool of Wind River[4] that performs license identification on a software package giving as output an SPDX file (including tag format).

4. IMPLEMENTATION NOTES

The architecture of the SLVT system is shown in Figure 2), whereas its source code along with some examples for use are available on GitHub[5]. SLVT is available under the GPL-2.0 license or any higher version of the license (GPL-2.0+). RESTful services expose the different functionalities providing responses in JSON (JavaScript Object Notation) format:

- *SPDX Validation Analysis Services*: expose the main functionality of SDPX license validation, whereas the possibility of retrieving corrected SPDX files with the suggested license is also provided (for cases, where a correction is feasible).
- *License Compatibility Services*: give the possibility to check compatibilities among licenses relying on the license graph without using SPDX. This service uses as input any number of licenses (with license name and version) and returns whether the licenses are compatible along with information about applicable license(s) in case they are.
- *Graph Expansion Services*: integrate license graph expansion functionalities giving users the possibility to add new vertices and edges to the license graph.

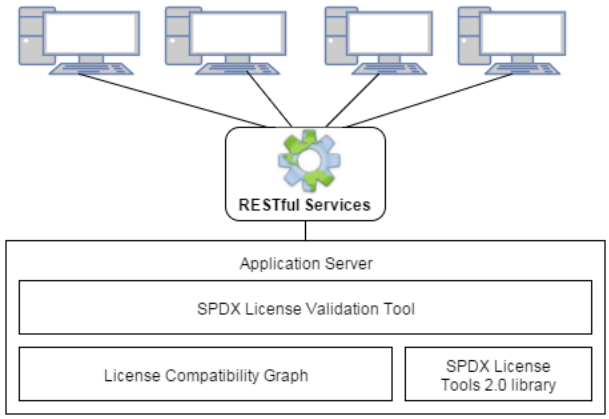

Figure 2: Architectural elements of SLVT.

For implementation purposes Java, PHP and JavaScript libraries have been used. Regarding the handling of SPDX files the *SPDX License Tools 2.0* provided by the SPDX workgroup has been employed. For instance, the *SPDX Viewer* forming part of the tools validates the SPDX document for its format and provides warning messages when parsing issues are encountered. The license graph implementation has been mainly based on the JGraphtl[6] library: a free Java graph library that provides mathematical graph-theory objects and algorithms.

Regarding the results of SLVT the screenshot of Figure 3 shows the analysis of SLVT for the software product Jexcelapi. Users are able to view information on the different licenses used in the software package (i.e., extracted licenses), the declared license and the analysis results indicating a validation success or failure with errors in the file. Warnings are also provided, e.g., if the SPDX files does not contain any *concluded license* that indicates the license that the creator of the SPDX file concluded (according to the SPDX specification).

5. USE DEMONSTRATION

We have evaluated the *SPDX License Validation Tool* using a number of SPDX files generated by Wind River (for

[4]1

[5]https://github.com/dpasch01/spdx-compat-tools

[6]http://jgrapht.org/

Table 1: SPDX file testing set and results of SLVT.

Software package	Package Size	License(s)	Valid	Extracted licenses
Anomos v.0.9.5	824.2 KB	GPL-3.0	✓	GPL-3.0, GPL-3.0+, GPL, LGPL-3.0+
CuteFlow v.2.11.2	3.81 MB	BSD-3-Clause	x	LGPL-2.1+, MIT-style, GPL-3.0+, GPL-2.0, LGPL-2.0+, ClearSilver, BSD-3-Clause, GPL, MIT, BSD, W3C-possibility, LGPL, GPL-2.0+
HandBrake v.0.10.1	9.96 MB	GPL-2.0	x	BSD-3-Clause, GPL, LGPL-2.0+, GPL-exception, BSD, GPL-2.0, MIT, AGPL, Sun-possibility, LGPL-2.1+, MIT-style, GPL-2.0+, MPL-1.1
Jexcelapi v.2.6.12	1.82 MB	LGPL-3.0	✓	LGPL, BSD-style, LGPL-2.1+
Joda-Time v.2.8	940.72 KB	Apache-2.0	✓	Apache-possibility, Apache-2.0, BSD
MrBayes v.3.2.5	5.9 4MB	GPL-3.0	✓	GPL-3.0+, GPL-3.0, GPL-2.0+
opencsv v.3.4	303.41 KB	Apache-2.0	✓	BSD-3-Clause, Apache-2.0
Previsat v.4.4.5	1.24MB	GPL-3.0	✓	Microsoft-possibility, GPL-3.0+, Sun-possibility
Samba v.4.1.0	33.4MB	GPL-3.0	✓	LGPL-3.0+, MIT, PostgreSQL, BSD-2-Clause, GPL-2.0+, BSL-1.0, Zlib GPL-1.0+, Apache-2.0, BSD-3-Clause,LGPL-2.0+, ISC, Python-2.0
VirtualDub v.1.10.4	2.1MB	GPL-2.0	✓	GPL-2.0+, GPL-2.0

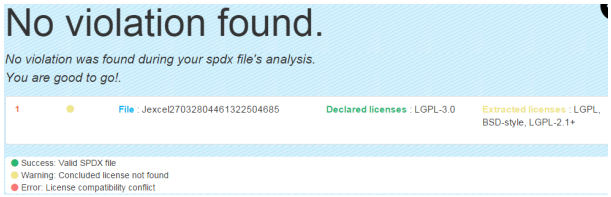

Figure 3: Results of SLVT analysis for Jexcelapi.

Samba) and FOSSology (for the remaining software after undergoing the license identification process offered by the Nomos identification agent of FOSSology). Single SPDX files and some combinations have been used as input. The software projects considered are depicted in Table 1 along with their respective sizes and the official software license(s) as indicated by their creators on the project website. The projects were randomly selected. Permissive (i.e., Apache-2.0, BSD 3-Clause, MIT/X11), weak copyleft (i.e., MPL-1.1, LGPL-3.0) and strong copyleft (i.e., GPL-2.0, GPL-3.0) licenses are considered, whereas the projects differ also in size.

Table 1 indicates also the results of the SLVT analysis on whether the SPDX file is valid and shows the extracted licenses from the different files as appearing in the SPDX file. Note that cases with possible license appearance (e.g., Sun-possibility) are not taken into account for licenses not covered in the license graph. For the licenses in the graph, the latest version of that license category is used when no license version is indicated (e.g., GPL-3.0 would replace GPL). The majority of software projects considered do not have any license violations (80%). In one of the incompatibility cases, the presence of MPL-1.1 causes violations, since this license is considered incompatible with many other licenses (e.g., LGPL and GPL families). For all three projects that contain violations no corrective solution exists.

Regarding combinations projects that were known to be compatible or incompatible were selected in order to verify whether SLVT would provide the correct result:

- *compatibility cases*: 1) Jexcelapi combined with Joda-Time, 2) Previsat combined with VirtualDub.
- *incompatibility cases*: opencsv, Previsat and VirtualDub are incompatible because Apache-2.0 from opencsv is incompatible with GPL-2.0 from VirtualDub, even though each independent file is valid and opencsv can be combined with Previsat without violations. The result of this case in SLVT is depicted in Figure 4.

Note that the results of all experiments have been manually verified in order to assess the accuracy of SLVT. This was performed by using the license graph in order to investigate whether the licenses of the SPDX file are compatible. However, errors resulting from false positives of the license identification tools used cannot be detected.

Figure 4: Results of SLVT analysis among opencsv, Previsat and VirtualDub.

6. CONCLUSIONS

In this demonstration paper we have presented the online *SPDX License Validation Tool* that performs license compatibility examination on SPDX files. The evaluation performed demonstrates that SLVT provides accurate results identifying violations caused by incompatibilities in licenses used in different files of a software package. Regarding the license compatibility graph expansion we rely on users, but do not provide currently any mechanism for validating user input. Our current aim is to evolve the process allowing input from multiple users and their votes for or against graph updates. Further improvements will focus on the decrease of the execution time that is currently high when large files are considered (>10 MB). We are also working on the extension of SLVT by allowing a more detailed modeling and compatibility check based on license content - divided into rights, obligations and additional conditions - instead of relying only on the license name and version captured in the license graph. In that respect modeling approaches of license text captured in ontologies or UML diagrams can be utilized [5].

7. REFERENCES

[1] R. M. Azzi. Cpr: how jacobsen v. katzer resuscitated the open source movement. *U. Ill. L. Rev.*, page 1271, 2010.

[2] I. E. Foukarakis, G. M. Kapitsaki, and N. D. Tselikas. Choosing licenses in free open source software. In *SEKE*, pages 200–204, 2012.

[3] L. Foundation and its Contributors. A common software package data exchange format, version 2.0. 2015.

[4] R. Gobeille. The fossology project. In *Proceedings of the 2008 international working conference on Mining software repositories*, pages 47–50. ACM, 2008.

[5] T. Gordon. Report on prototype decision support system for oss license compatibility issues. *Qualipso*, 79, 2010.

[6] T. F. Gordon. Analyzing open source license compatibility issues with carneades. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law*, pages 51–55. ACM, 2011.

[7] G. M. Kapitsaki and F. Kramer. Open source license violation check for spdx files. In *Software Reuse for Dynamic Systems in the Cloud and Beyond*, pages 90–105. Springer, 2014.

[8] G. M. Kapitsaki, N. D. Tselikas, and I. E. Foukarakis. An insight into license tools for open source software systems. *Journal of Systems and Software*, 102:72–87, 2015.

[9] G. Klyne and J. J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. 2006.

[10] C. Y. Lee. An algorithm for path connections and its applications. *Electronic Computers, IRE Transactions on*, (3):346–365, 1961.

[11] V. Lindberg. *Intellectual property and open source: a practical guide to protecting code.* " O'Reilly Media, Inc.", 2008.

[12] F. Mancinelli, J. Boender, R. Di Cosmo, J. Vouillon, B. Durak, X. Leroy, and R. Treinen. Managing the complexity of large free and open source package-based software distributions. In *Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on*, pages 199–208. IEEE, 2006.

[13] D. L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[14] A. Morin, J. Urban, and P. Sliz. A quick guide to software licensing for the scientist-programmer. *PLoS Comput Biol*, 8(7):e1002598, 2012.

[15] D. A. Wheeler. The free-libre/open source software (floss) license slide. *Online http://www. dwheeler. com/essays/floss-license-slide. pdf*, 2007.