Towards open source software licenses compatibility check

Georgia M. Kapitsaki
University of Cyprus
Aglantzia, Cyprus
gkapi@ucy.ac.cy

Athina Paphitou
University of Cyprus
Aglantzia, Cyprus
athensp22@gmail.com

Achilleas P. Achilleos
Frederick University Cyprus
Limassol, Cyprus
com.aa@frederick.ac.cy

ABSTRACT

The use of free and open source software is increasing and there is currently a tendency towards more openness in the provision of open source software. However, libraries that are used in conjunction with the software may affect the final license selection of the open source software and special caution is needed by software developers. Existing tools provide the means to extract license information from software projects, but this information has not been utilized towards recommending licenses that do not cause license violations. In this paper, we present our work towards the recommendation of licenses that satisfy the license compatibility requirement taking into consideration the licenses of third party libraries used in the software project. We have employed a dataset of 160 open source software projects to compare license compatibility using license extraction techniques implemented in different tools, i.e. Nomos and Ninka, whereas we have integrated the license extraction process in the *findOSSLicense* open source license recommender system, in order to recommend licenses that do not cause violations. The evaluation results and a small scale user study demonstrate the added value of the approach for the software developers in being better informed about license compatibility.

CCS CONCEPTS

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

KEYWORDS

Free/Libre and Open Source Software, licensing, license compatibility, recommender system

ACM Reference Format:
Georgia M. Kapitsaki, Athina Paphitou, and Achilleas P. Achilleos. 2022. Towards open source software licenses compatibility check. In *26th Pan-Hellenic Conference on Informatics (PCI 2022), November 25–27, 2022, Athens, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3575879.3575973

1 INTRODUCTION

Free/Libre Open Source Software (FLOSS) is accompanied by licenses that define the conditions under which the software can be used, modified and distributed [11]. Since open source software relies often on the use of third party libraries for various functionalities, the license of these libraries is very important for the licensing scheme of the resulting software, as failure to comply with the terms of the licenses of the libraries may lead to license violations and have legal consequences [12]. Licenses are divided into three main categories: permissive licenses, such as MIT, provide more flexibility, weak-copyleft licenses, such as LGPL (GNU Lesser General Public License) licenses, place more restrictions, and strong-copyleft licenses, such as GPL (GNU General Public License) licenses, are even more restrictive requiring that a derivative work that modifies or adds to the original work is also made available as open source under the same or under a compatible license, when distributed.

Websites and tools that assist users in the license selection have emerged, such as choosealicense[1] used by GitHub informing users about some open source software licensing options and the open source license recommender *findOSSLicense*[2], whereas other web locations are also informative about the content of licenses, e.g. TLDRLegal[3] provides information about the content of a large number of open source licenses. However, these information sources do not consider the existing licenses in the source code used within a software project (e.g. from the use of third party libraries in the code). License extraction tools can be a facilitator in this area, as they scan the software source code and inform users on retrieved licenses. *FOSSology* [6, 8], *Ninka* [4] and Automated Software License Analysis (ASLA) [15] are examples of tools that offer this functionality, whereas some provide also limited support for license compatibility. Although the above are useful, they provide scattered solutions - either for informing about licenses and their content, or for extracting licenses from source code - and there is no integrated approach to license compatibility that considers both aspects, recommending to users appropriate licenses that do not cause violations to the existing licenses from the libraries used.

Having as motivation the above, in this work we propose an approach that utilizes available license extraction mechanisms, while it considers also information from the software project description in order to discover more third party libraries (found in the README.md file in GitHub repositories). As the first step in this approach, we examined the compliance level of 160 open source projects, comparing available license extraction tools, and we have then integrated our license compliance approach in the *findOSSLicense* open source license recommender system [9, 10]. The contribution of our work is two fold: (1) We are offering a comparative view of a number of license extraction approaches, i.e. Nomos agent of *FOSSology*, *Ninka*, and project description analysis, (2) We are

PCI 2022, November 25–27, 2022, Athens, Greece
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9854-1/22/11.
https://doi.org/10.1145/3575879.3575973

[1] https://choosealicense.com/
[2] http://findosslicense.cs.ucy.ac.cy/
[3] https://tldrlegal.com/

proposing a new approach that can be utilized by software developers for choosing the open source license(s) to apply on their software project by recommending licenses that are compatible with used licenses. The approach can be useful for developers that can utilize it in their activities, acting as a guideline for choosing appropriate licenses, whereas the comparative view of license extraction tools can provide insight for better understanding their role and function. We have evaluated our approach using the enhanced version of *findOSSLicense* in a small scale user study with the participation of 16 users.

The rest of the paper is structured as follows. Section 2 provides an overview of related work. Section 3 is dedicated to the presentation of our approach for recommending compatible licenses, whereas its integration in the *findOSSLicense* recommender system is described in section 4. The evaluation of the work and limitations are presented in section 5. Finally, section 6 concludes the paper.

2 BACKGROUND AND RELATED WORK

A variety of tools that assist in identifying project licenses exist. Many of the tools rely on analyzing the source code and accompanying files of a software project in order to detect references to licenses, and on comparing these references against license templates from a data store. The Automated Software License Analysis [15] is a reverse engineering tool that extracts licenses from source code modules. It uses predefined license identification templates, whereas users can add new templates. *FOSSology*[4] is a widely used open source license compliance software system and toolkit [6, 8]. It consists of a number of integrating agents for license identification, including Nomos and Monk agents.

In order to extract information about open source software licenses, Nomos agent uses small phrases (and regular expressions) and heuristics; for instance, a phrase should be near/far from another phrase or phrases. Monk agent performs text-based search and requires good text pieces/patterns against which open source licenses can be searched for. The *Ninka* tool is the third license scanner that has been integrated in *FOSSology* [5]. It relies on scanning the text of the source code and of license files in order to find text that is consistent with that of an open source software license. The first lines from the source code files are extracted for this purpose (approximately 1,000 lines). The Binary Analysis Tool (BAT) covers different functionalities and can point to reused components through string comparisons when operating on unpacked files [7]. Analysis of JARs (Java ARchives) in order to detect licenses is feasible via the Kenen tool that also utilizes Ninka [4]. ScanCode[5] scans the source code files for information about open source licenses, copyright and other information.

Various online resources that provide license information suggesting which licenses are more appropriate for each case or providing more information on specific licenses are available. TLDRLegal is a portal that provides license information indicating for each license what the user can, cannot and must do. Choosealicense shows users how to navigate among some license choices, in order to assist them in selecting an appropriate license for their repository, whereas it also lists properties of popular licenses, divided into permissions (e.g. distribution), limitations (e.g. liability) and conditions (e.g. disclose source). FreeMeLegal introduces the possibility of recommending licenses for open source projects [13]. The user can choose licenses already used in her project, decide about some requirements (e.g. disclose source), permissions (e.g. commercial use) and constraints (e.g. patent use). *findOSSLicense* is a recommender system that assists developers in selecting an appropriate license for their software project based on their needs [9, 10]. *findOSSLicense* combines and models features from widespread open source software licenses, and employs user responses to questions, such as application type the user wants to develop, permissions that the user wants to give to other application users (e.g. permission to use the original software with commercial software), and licenses already in use to suggest to the user appropriate license(s).

Relation with previous works. In relation to the above works, we offer a comparative view of open source software license extraction approaches, enhancing them with a new extraction aspect that concerns the detection of used libraries indicated in the project description (README.md file). We also integrate this compatibility approach to recommend appropriate FLOSS licenses to owners of a project repository in GitHub, assisting software developers in the license selection process with this recommender-based approach that does not exist in current work.

3 FINDING COMPATIBLE LICENSES: APPROACH AND COMPARISON

Our approach targets newly created repositories, where the appropriate license to be applied on the repository needs to be determined, focusing on license compliance and taking into consideration the licenses of third party libraries used. The steps of the process are depicted in Figure 1. We are using GitHub to obtain the project source code and its respective description. We are using the source code of the software project and extract information regarding licenses using different tools: 1) Nomos agent, 2) *Ninka*, 3) licenses of the libraries indicated in the project description (README.md file in GitHub). Specifically, the following steps are followed for each software repository, after obtaining the project source code as shown in Figure 1:

- The license scanning is performed on the project source code, using Nomos, *Ninka*, and the README.md analysis.
- The existing project license, as indicated by its creators, is retrieved from the GitHub repository.
- License compatibility analysis is performed for the licenses detected by each extraction tool independently, and for the union of the licenses detected by the tools (i.e. where the licenses detected by each tool are all considered).
- The license recommendation algorithm is executed for the licenses scanned by each tool independently and for the union of the licenses detected by the tools, in order to recommend appropriate compatible licenses that can be applied on the project.
- If no compatible licenses are found for recommendation, the recommendation algorithm is executed again taking into

[4]https://www.fossology.org/
[5]https://github.com/nexB/scancode-toolkit

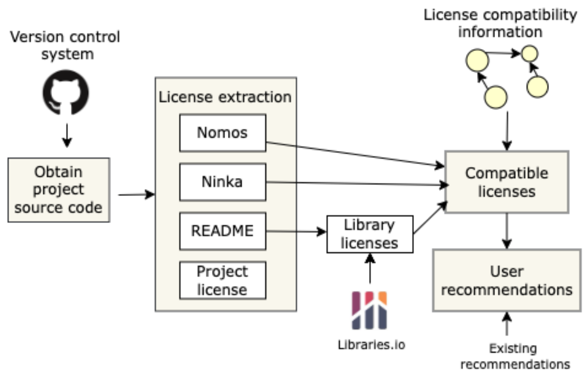


Figure 1: Steps to recommend compatible licenses.

consideration also license pairs marked as "*use with caution*". This indication is provided in some cases by the recommendation process of *findOSSLicense*, when it is not conclusive whether two licenses can be considered compatible or not [9, 10].

The information obtained via the license extraction tools are compared against the project license as indicated by the repository creators (the license is extracted using the GitHub REST API). The main aim of these steps is the comparison of the results of the different extraction tools, whereas we are also considering the case of combining all tool results, by comparing the union of the licences extracted by the tools against the project license, under the assumption that this combination might improve the license compliance process, as it compares the project license against a larger number of licenses detected by all tools.

For the compatibility information between licenses, we are employing the compatibility matrix of *findOSSLicense*, which indicates compatibility or incompatibility between licenses. There are some known cases of license incompatibility. For instance, the Apache-2.0 license is not compatible with the GPL-2.0 license, but it is compatible with GPL-3.0 and later versions of the license. The license compatibility matrix of *findOSSLicense* is based on analysis performed in previous works of the authors [2, 16]. Since *findOSSLicense* covers 32 licenses, when compatibility information is not available for some license pairs, the users are advised to use the license combination "*with caution*". During the compatibility analysis these licenses are initially not considered. However, if no compatible license can be found, these cases are considered as compatible in order to be able to provide a result to the users.

4 INTEGRATION IN FINDOSSLICENSE RECOMMENDER SYSTEM

findOSSLicense already provides some support for license compatibility by informing the users on compatibility between the recommended licenses and any license(s) the user may have manually indicated in the recommender system as already used in her project (in this case the user needs to add the exact license names and version). Via the integration with GitHub, the *findOSSLicense* user can log in with her GitHub account and request an examination of an existing repository the user is working on for compatibility purposes.

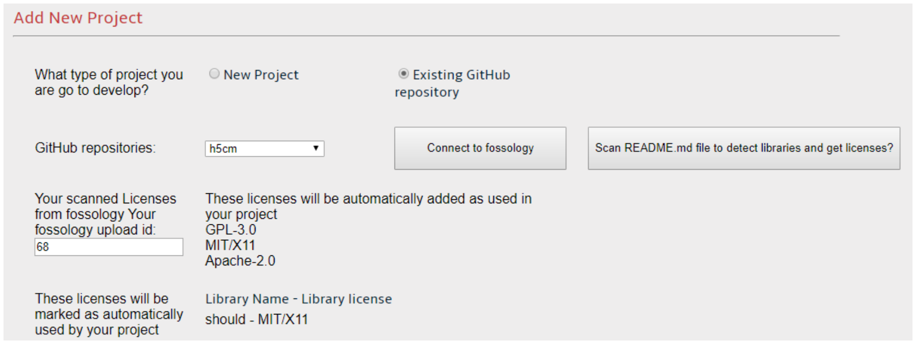
For the license extraction process in *findOSSLicense*, we have decided to use *FOSSology* at the current stage of implementation based on a preliminary comparison we performed between the different tools, and since it offers many capabilities and is widely adopted by various companies and organizations [14]. For instance, the importance of using open source tools for license compliance is presented from the industrial perspective of Siemens AG, including the use of *FOSSology* in [1]. The user can thus, connect to *FOSSology* in order to trigger the scanning of her repository using the Nomos agent. Regarding the extraction agents available in *FOSSology*, we have not used the Monk agent, as it requires closer matches to correctly identify license information and cannot identify unknown licenses. *Ninka* extracts the first lines (approximately 1,000) from source code files in order to use them in the recognition process, and may thus extract a lower number of licenses than Nomos. Using one main scanner has also the advantage of keeping the process simple for the user and is useful especially for users that are not experts on FLOSS or licensing. However, future work will examine providing the user the possibility to choose among different license identification tools in *findOSSLicense*.

In the framework of this work, we have also added the option for the user to receive a recommendation for license compatibility based only on the examined GitHub repository, without taking into consideration the other user requirements provided as input in *findOSSLicense*, e.g. answers to the user on specific questions regarding software usage. The user has also the option to trigger the parsing of the project description in order to identify licenses from used libraries. For this purpose, the knowledge base of *findOSSLicense* was expanded to include information about commonly used libraries and their licenses. Specifically, a set of 100 libraries for each of the eight programming languages supported by *findOSSLicense* based on the popularity of languages in the TIOBE index, i.e. PHP, Java, C, C++, C#, Python, Visual Basic, JavaScript, were included in the data store of *findOSSLicense* along with their respective license. The libraries were selected from *libraries.io*. We used its respective API to collect the first 100 libraries with the most depended count (as depicted in Figure 1).

The part of the screenshot, where the user can select one of her existing GitHub repositories, connect to *FOSSology*, trigger the scanning of the project description and view the respective results with the identified licenses is visible in Figure 2. This information can be subsequently considered in the recommendation process of *findOSSLicense*. *findOSSLicense* recommends licenses based on other user requirements, besides compatibility, but as aforementioned the user has the option of seeing only the licenses that are compatible with the licenses already used in the project, whereas she can receive recommendations based only on compatibility (neglecting thus the other user requirements).

5 EVALUATION

We have evaluated our approach by comparing the license extraction tools using a dataset of open source projects, which was the first step in order to better understand how they behave before their use, whereas we have also performed a small scale user study


Add New Project

What type of project you are go to develop? ○ New Project [✓ Existing GitHub repository]

GitHub repositories: h5cm

Connect to fossology

Scan README.md file to detect libraries and get licenses?

Your scanned Licenses from fossology Your fossology upload id: 68

These licenses will be automatically added as used in your project
GPL-3.0
MIT/X11
Apache-2.0

These licenses will be marked as automatically used by your project

Library Name - Library license
should - MIT/X11

Figure 2: License extraction process triggering in findOSSLicense.

Table 1: Dataset summary for license compatibility comparison.

Programming language	License	Count	Programming language	License	Count
PHP	MIT	17	JavaScript	MIT	17
	BSD-3-Clause	2		Apache-2.0	1
	LGPL-2.1	1		BSD-3-Clause	1
Java	MIT	4		CC0-1.0	1
	Apache-2.0	15	C++	MIT	4
	CC-BY-SA-4.0	1		Apache-2.0	9
C	MIT	5		BSD-3-Clause	2
	Apache-2.0	4		MPL-2.0	1
	BSD-2-Clause	1		GPL-2.0	1
	BSD-3-Clause	3		GPL-3.0	1
	LGPL-2.1	1		Unlicense	1
	GPL-2.0	2		BSL-1.0	1
	GPL-3.0	4	C#	MIT	12
Python	MIT	9		Apache-2.0	5
	Apache-2.0	5		MS-PL	1
	LGPL-3.0	1		GPL-3.0	2
	GPL-2.0	1	Visual Basic	MIT	13
	GPL-3.0	1		GPL-2.0	1
	AGPL-3.0	1		GPL-3.0	3
	WTFPL	1		CC-BY-4.0	1
	Unlicense	1		CC-BY-SA-4.0	1
				Artistic-2.0	1

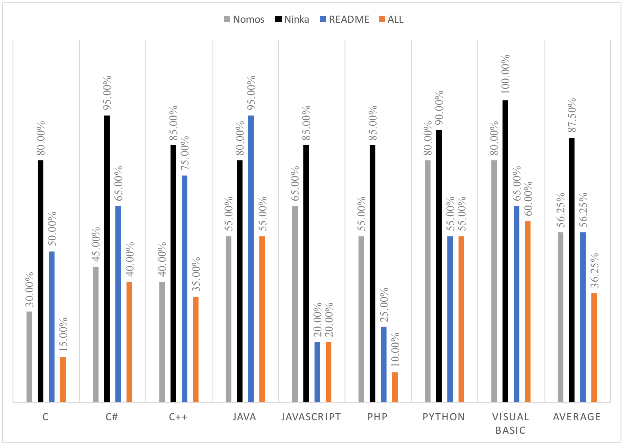


Figure 3: Percentages of projects without compatibility issues per scanning tool and programming language.

in order to receive feedback from users on the use of the license compatibility integration in *findOSSLicense*.

5.1 License extraction tools comparison

In order to discover whether the license added to a project repository in GitHub is respecting license compatibility and suggest to users the appropriate compatible license(s) when it does not, we have collected a dataset of 160 open source projects from GitHub. The dataset, collected via the GitHub REST API, consists of the 20 most starred projects on GitHub for each programming language covered in our work (i.e. PHP, Java, JavaScript, C, C++, C#, Python, Visual Basic). In order to limit the processing time, we limited the size of the selected projects to 100 MB or less. The dataset summary indicating the licenses present in the selected projects for each programming language is available in Table 1. The projects used carry 18 different licenses. The MIT license appears more often, followed by Apache-2.0 and GPL-3.0.

The results that indicate whether the project license on GitHub is compatible with all licenses used as detected by each scanning tool are depicted in Figure 3. For the case of Nomos of *FOSSology* we disregard cases, where no specific license is detected, i.e. when the indication UnclassifiedLicense is provided as result. When the union of all detected licenses (by all tools together) is considered in the compatibility examination, the percentage of projects with an appropriate license is low (36.2%), since in this case a larger number of licenses need to be compatible with each other. When only *Ninka* is considered the number of projects with appropriate licenses is larger, reaching 87.5% on average. Regarding the results based only on the scanning of the project description, i.e. README file, we expect them to be less accurate, as this process does not examine the source code and its content.

Since overall the cases with incompatible licenses are relatively high, we performed a manual analysis for some cases, in order to examine if the compatibility checking on scanned licenses was accurate and whether the license scanning found all licenses in use by a project. For this manual analysis, we went through all files where a license was detected, in order to verify the presence of the license in the file. We are providing details for 4 of these cases that were further examined in Table 2. The full dataset used in this part of our work, i.e. GitHub project details, scanning results and manual analysis results for the projects presented in this section and 4 more projects, have been made available for replication purposes in a Zenodo repository [3]. In Table 2, we use parentheses to denote cases where both the generic license family (e.g. LGPL) and the license with version (e.g. GPL-2.0) are detected by the tool. Regarding compatibility, we denote whether our manual analysis detected that the license used is actually not appropriate for the project for incompatibility purposes and explain the reasons that led to a false result in the following cases:

- *composer*[6]: In the case of Nomos, the BSD license is detected because it is used in the automated tests of the repository developed via PHPunit and not in the main source code of the project. AGPL, GPL LGPL licenses are also detected, since they are present in a text file that mentions license examples. All these licenses are not compatible with MIT. For instance, BSD-3-Clause and BSD-2-Clause are not compatible with the MIT license, because MIT has a right that is missing in the BSD licenses. Specifically, the MIT license allows for distribution without contribution credits, whereas BSD licenses do not. However, the location the above licenses were found should not affect the project license. *Ninka* detected the BSD-2-Clause license in a file that contains a method modified and adjusted from the project Sslurp that carries the BSD-2-Clause license. Although Nomos scanning resulted in a false positive for the BSD-2-Clause license, *Ninka* accurately detected the use of a piece of code containing the license that should correspond to a source of license violation.
- *Material-Animations*[7]: incompatibility is indicated due to the presence of the GPL and Apache-2.0 licenses. GPL is falsely detected by Nomos in an animated file in the project repository (*screenshots/scenes_anim.gif*). However, the detection of Apache-2.0 is accurate by both Nomos and *Ninka* tools, as a source code file and an XML file contain on the top of the files a description pointing to the Apache-2.0 license. This project is an incompatibility case according to the scanning that requires further investigation, in order to verify that these files are used in the source code and when the software is running (and are not used, for instance, only for testing purposes).
- *the_silver_searcher*[8]: Incompatibility is indicated due to the presence of the GPL-3.0+ license that is not compatible with the less restrictive Apache-2.0 license. Both Nomos and *Ninka* detected the use of the GPL-3.0+ license in the source code files. The GPL-3.0+ license text indeed appears in the file *ax_pthread.m4*, which contains macros for building C programs. However, the description of the macro states the following: "*You need not follow the terms of the GNU General Public License when using or distributing such scripts, even though portions of the text of the Macro appear in them. The GNU General Public License (GPL) does govern all other use of the material that constitutes the Autoconf Macro*." Both *Ninka* and Nomos fail to detect that this case refers to an exception in the use of the GPL-3.0+ license, providing a false negative result, although *FOSSology* does detect the Autoconf-exception license (i.e. GNU General Public License v2.0 w/Autoconf exception) that is close to the references present in the text. The presence of BSD licenses, identified both by the Nomos and the README scanning, does not affect compatibility, as they are compatible with the Apache-2.0 license.
- *hardseed*[9]: This is an example of a false positive for Nomos that detects the GPL-3.0 license in the *ycm_extra _conf.py* file, although the text of this file reads: "*This file is NOT licensed under the GPLv3, which is the license for the rest of YouCompleteMe*." Although GPL-2.0 is compatible with GPL-3.0, the reverse direction of compatibility is not valid. The Unlicense correctly detected by Nomos does not cause any compatibility issues, as it is compatible with GPL licenses[10].

A key weakness of the README.md file license scanning is that it relies on keyword search without considering the context of the detected keyword. Since a lot of libraries' names consist of common words, this results in some inaccuracies in the license scanning. This problem has appeared in the case of the *flysystem*[11] project with the Apache-2.0 license being falsely detected in the project description file. Similar false positives are the reason for the low percentage of compatibility in the PHP, JavaScript and Python projects, as most of them carry the MIT license, which is one of the most permissive licenses. For the case of Java, four false positives of Nomos were identified by manual analysis, raising the percentage of compatible licenses in Java projects from 55% to 75%. In the cases where incompatibility actually exists based on our analysis further investigation is required in order to verify whether that part of the source code forms part of the main software distribution.

5.2 Small scale user study

For the small scale user study, senior and master students, as well as software engineers from the local industry were recruited using email and face-to-face communication. Participants were asked to interact with the compatibility process of *findOSSLicense*, using a deployment of the recommender system on a test server and they were then asked to complete an online questionnaire[12] about their experience, the licenses that were recommended to them and the compatibility information. The questionnaire contained also questions about usability aspects, the former experience of the participants with open source software and open source software licenses, and demographic information. In order to participate in the study, the main requirement was to own a repository on GitHub that would be given as input to *findOSSLicense* for the license detection and recommendation process. The participants were free to use any type of repository in the supported programming languages. This way the participants examined our approach, in order to identify an appropriate license for their own software project.

16 users participated in the study. 56.3% are male and the remaining female. 56.3% of the participants are software engineers, 25.1% are research scientists and 18.8% students. Most users use FLOSS very often (56.3%) and 12.5% use it rarely. 50% are familiar or

[6]https://github.com/composer/composer
[7]https://github.com/lgvalle/Material-Animations
[8]https://github.com/ggreer/the_silver_searcher
[9]https://github.com/yangyangwithgnu/hardseed
[10]https://www.gnu.org/licenses/license-list.en.html#Unlicense
[11]https://github.com/thephpleague/flysystem
[12]https://forms.gle/2GaH2DgtHZKK9zZMA

Table 2: Details of analysis for example projects with incompatibility.

Project	Programming language	Official license	Nomos licenses	Ninka licenses	README licenses	False positive	Incompatibility exists
composer	PHP	MIT	MIT, BSD-3-Clause, AGPL, LGPL(-2.1+), BSD-2-Clause, GPL(-3.0+)	MIT, BSD-2-Clause	MIT	Nomos	Yes
Material-Animations	Java	MIT	MIT, GPL, Apache-2.0	Apache-2, MIT	MIT, Unlicense	-	Yes
the_silver_ searcher	C	Apache-2.0	Apache-2.0, BSD-1-Clause, Public-domain, GPL-3.0+, Autoconf-exception	Apache-2.0, GPL-3.0+	BSD-3-Clause	Nomos, Ninka	No
hardseed	C++	GPL-2.0	GPL-2.0, Unlicense, GPL(-3.0), MIT	GPL-2.0+, MIT	MIT, Ruby	Nomos	No

very familiar with FLOSS licenses, whereas 12.5% are only slightly familiar. However, 43.8% are not familiar at all or are only slightly familiar with license compatibility (31.3% of the participants are very familiar) and 37.6% are not familiar at all or are only slightly familiar with license categories (again 31.3% of the participants are very familiar). Regarding the interaction with *findOSSLicense*, most participants (81.3%) indicated that using the connection with *FOSSology* (i.e. Nomos agent) and the README file scanning triggering was clear or very clear. Most users (81.3%) find this information about the licenses used in their project useful or very useful, indicating that this is a useful feature of the recommender system (the remaining 18.8% of the participants gave a score of 3 indicating that this information is slightly useful). Most users (87.6%) were very satisfied or satisfied with the recommendations provided to them based on compatibility, and most (87.5%) indicated that they will adopt one of the recommended licenses for their project. Among the participants that did not intend to use one of the recommended licenses (2 users), one indicated that she is satisfied with the license the project already has, since it is a forked project, and the other mentioned that she did not like licenses, which most probably means that the participant did not prefer to use any license at all (the same participant indicated that she was very satisfied with the recommendation results).

5.3 Limitations

The small scale user study base of 16 participants may have affected the conclusions we reached about the relationships in our data. Future work will replicate the study with a larger number of users. Moreover, some project libraries detected by the extraction tools may be used only for testing purposes without being included in the final software distribution. At the current state, this case is not considered, as it requires further analysis of the source code structure. However, it may have also affected our conclusions, as it may have lead to some false positives in license violation detection in the respective software repositories. We have performed a manual analysis in a small number of projects in order to study the effect of this risk, and we have marked a number of such cases that can also be found in our dataset [3]. Finally, the current approach regarding license extraction from the project description libraries relies only on the information in the respective README file. This process fails to identify cases, where a license is indicated, but is not used in the project source code.

6 CONCLUSIONS

In this paper, we have presented our work towards a compatibility process for FLOSS licenses accompanied by a tool that has been integrated in the *findOSSLicense* open source license recommender system. We are comparing license extraction tools and provide to the users the opportunity to use one of them to guide licenses recommended to them. This process can assist software developers understand license implications better and help them make informed license decisions for their software projects. As future work, we intend to consider more license identification tools for comparison purposes with the aim of improving the accuracy of the compatibility check. We also intend to investigate Natural Language Processing techniques with static code analysis, in order to increase the accuracy of the library usage in the application source code (e.g. neglect cases where a license appears only in test files that are not distributed with the software project).

REFERENCES

[1] Oliver Fendt and Michael C Jaeger. 2019. Open source for open source license compliance. In *IFIP International Conference on Open Source Systems*. Springer, 133–138.

[2] Ioannis E Foukarakis, Georgia M Kapitsaki, and Nikolaos D Tselikas. 2012. Choosing Licenses In Free Open Source Software.. In *SEKE*. 200–204.

[3] Kapitsaki Georgia, Paphitou Athina, and Achilleos Achilleas. 2022. *GitHub project dataset for license analysis*. https://doi.org/10.5281/zenodo.6347648

[4] Daniel German and Massimiliano Di Penta. 2012. A method for open source license compliance of java applications. *IEEE software* 29, 3 (2012), 58–63.

[5] Daniel M German, Yuki Manabe, and Katsuro Inoue. 2010. A sentence-matching method for automatic license identification of source code files. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 437–446.

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

[7] Armijn Hemel, Karl Trygve Kalleberg, Rob Vermaas, and Eelco Dolstra. 2011. Finding software license violations through binary code clone detection. In *Proceedings of the 8th Working Conference on Mining Software Repositories*. ACM, 63–72.

[8] Michael C Jaeger, Oliver Fendt, Robert Gobeille, Maximilian Huber, Johannes Najjar, Kate Stewart, Steffen Weber, and Andreas Wurl. 2017. The FOSSology Project: 10 Years Of License Scanning. *IFOSS L. Rev.* 9 (2017), 9.

[9] Georgia Kapitsaki and Georgia Charalambous. 2019. Modeling and recommending open source licenses with findOSSLicense. *IEEE Transactions on Software Engineering* (2019).

[10] Georgia M Kapitsaki and Georgia Charalambous. 2016. Find your Open Source License Now!. In *Software Engineering Conference (APSEC), 2016 23rd Asia-Pacific*. IEEE, 1–8.

[11] Andrew M St Laurent. 2004. *Understanding open source and free software licensing: guide to navigating licensing issues in existing & new software.* " O'Reilly Media, Inc.".

[12] David McGowan. 2001. Legal implications of open-source software. *U. Ill. L. Rev.* (2001), 241.

[13] Kevin Schmidt. 2016. *License usage analysis and license recommendation in open source software development.* Ph. D. Dissertation. Masterarbeit, Koblenz, Universität Koblenz-Landau, Campus Koblenz, 2015.

[14] Kate Stewart, Phil Odence, and Esteban Rockett. 2010. Software package data exchange (SPDX) specification. *IFOSS L. Rev.* 2 (2010), 191.

[15] Timo Tuunanen, Jussi Koskinen, and Tommi Kärkkäinen. 2009. Automated software license analysis. *Automated Software Engineering* 16, 3-4 (2009), 455–490.

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