The FOSSology Project: 10 Years Of License Scanning

Michael C. Jaeger,[a] Oliver Fendt,[a] Robert Gobeille,[b] Maximilian Huber,[c] Johannes Najjar,[c] Kate Stewart,[d] Steffen Weber,[c] and Andreas Würl [c]

(a) Siemens AG, Corporate Technology (b) Freelance Consultant
(c) TNG technology Consulting GmbH (d) The Linux Foundation

DOI: 10.5033/ifosslr.v9i1.123

Abstract
FOSSology is an open source project developing a Web server application and a toolkit for open source license compliance. As a toolkit it allows performing license copyright and export control scans from the command line. The FOSSology Web application provides a database and Web UI for implementing a compliance workflow.
The FOSSology project published the first version of its software in December 2007. Given this ten year anniversary of license scanning this article presents the motivation for building and using FOSSology its history and its status as of today. Because SPDX represents the *de facto* standard for exchanging license and copyright information about software packages an introduction to FOSSology's support for exporting and importing SPDX documents is also presented.

Keywords
Free and Open Source Software, License Scanning, Compliance Tools, SPDX, OSS Analysis

Introduction

The use of software is granted under a specific license. Open source software, like proprietary software, has conditions that must be complied with. In absence of a license, the software must be treated as all rights reserved, and not distributed further. As a result, understanding the license is key to being able to determine what one is allowed to do with the software.

Open source software is licensed using open source compliant licenses. The Open Source Initiative provides a definition of open source[1] and an open source compliant license has to comply with all parts of this definition. However, the list maintained by OSI is only a small set of the licenses currently in use of open source projects.

Over the past few decades a lot of different Open Source compliant licenses were drafted by different authors. Today we are confronted with the so called "OSS license proliferation" problem. More than 1000 Open Source compliant licenses are "in use", some of them differ only in a few words, while

1 Open Source Initiative: The Open Source Definition https://opensource.org/osd - 2017

others pursue a totally diverse interests. The SPDX working group collects main open source licenses and as of today, this effort has identified more than 340.[2] While a few licenses are very popular, meaning that many open source projects apply them to their work, as can be seen by usage statistics,[3] some other licenses are just published and applied by individual organizations to their software. Examples of popular licenses are the GNU General Public License 2.0 or the MIT License; examples of organization specific licenses are the Apple Public Source License 2.0 or the Microsoft Public License. Very briefly, authors and organizations have created all these different licenses for multiple reasons, among which:

- Authors of open source software have particular intentions for the use of their open source software;
- Commercial organizations strive to protect their commercial interests; or
- Non-profit organizations strive to protect or promote the use and adoption of open source software.
- Given the three points above, individuals or organizations have authored updated versions of their licenses, adding to the number of existing texts with even more new texts.

This article does not intend to compare or discuss all the different licenses. Rather it points to another challenge that results from the high number of existing license texts: Assuming the redistribution of an open source software component, regardless if it is as part of a commercial product or as part of a new open source project, this step requires the determination of the exact text of the applicable license for multiple reasons:

- Some licenses request providing the license text along with the redistribution of the software component.
- Some licenses express particular conditions when exercising the granted right of redistribution.
- Some conditions of some licenses are not compatible with conditions of other licenses. In this case combining two components licensed with incompatible licensing conditions between them is not possible.

As a result of the explanations given above, the first step of redistributing open source software is to determine the exact license text. However, realistically though, because each open source project tends to borrow from others, a mix of licenses tends to be present in most open source software components. When there are tens of thousands of files that make up a modern software package, it becomes a significant amount of work to properly respect the licenses. Therefore, the challenge is not only the great number of existing license texts, but also to cover the fact that many open source components show multiple open source licenses applying for some parts of the component.

In addition, a third challenge arises: authors of open source software do not use, in many cases at least, a standardized form of licensing. While licenses, such as the GPL versions, have standardized headers for source files to express a common way of licensing, many authors have found individual ways of referring to a license, sometimes using prosaic language. Thus, license statements which refer to a common license text can be either not unambiguously pointing to a particular license or are just hard to identify as a licensing statement.

2 SPDX Workgroup - a Linux Foundation Collaborative Project: SPDX License List https://spdx.org/licenses/ - 2016
3 Github.com: Open source license usage on GitHub.com https://blog.github.com/2015-03-09-open-source-license-usage-on-github-com/ - 2015

In summary, we have three different challenges for finding exactly the applicable license texts for use when redistributing open source software:

- A high number of licenses exist (OSS license proliferation);
- Multiple licenses can be found in a single open source component;
- Authors sometimes do not unambiguously refer to a particular license including its text.

Software tools exist to cover these challenges: license scanning software searches in open source software code for known license texts, licensing expressions and license relevant statements. One of these tools is provided by the FOSSology project. The FOSSology software is designed to determine the licensing condition of open source components. FOSSology was first published in 2007 by a group of Hewlett-Packard (HP) engineers, which is about 10 years ago. Therefore, it is now a good time for updating what has happened with FOSSology and its status as of today.

This article is organized as follows: the next section introduces the FOSSology project and gives a brief overview of its history. A subsequent section explains some of the technology used in FOSSology. Another section provides an overview of FOSSology and SPDX and the last section concludes this article.

Project History

FOSSology was first published by engineers from HP. An early first version of the software existed inside the company. Before FOSSology came in to being, an HP software engineer, Glen Foster, wrote some tools to perform license scanning. The focus was on scanning Linux distributions released with HP products. At that time Linux distributions were already large portions of open source software. Thus, a scan tool with the capability to scan large archives was the focus from the beginning. A first version of the software consisted of individual shell scripts. Subsequently, those scripts evolved into C language and compiled executables for speed. Then, the C code was enhanced to make it more capable for extension with future license texts and more licensing statements. This resulted in the original Nomos license scanner. FOSSology combined Nomos with a license categorization concept named buckets: users could define buckets based on detected licenses. With this approach, software was scanned with individual file focus. At the same time the software provided a large number of static HTML files for reporting.

In a subsequent effort, Robert Gobeille became involved by leading a project to speed this process up. The basic approach was the reuse of scans: Files that had been scanned already would not show a different licensing when scanned again. By including a database, the software avoided rescanning a large percentage of the files in a distribution. Another point was the reporting, which was at first, the standard output of the executables. By creating a plugin Web interface served by a Web server, dynamic and configurable reporting could be easily added.

After this setup had been established, the project came to realize that the value of the FOSSology project was in free use for all organizations, while plans to productize it seemed unlikely. Rather, the idea emerged that its development could be leveraged by making it an open source project. Thus FOSSology was born. One of the visions for FOSSology was to make it an open source data mining system, not being limited to scanning for license texts and statements, hence the name FOSS + ology - the study of Free and Open Source Software. Therefore FOSSology implemented an architecture of pluggable agents that can be composed into a pipeline of tasks applied to an uploaded open source component. After FOSSology was made open source, it was not only limited to providing the community with a license scanner (at that time called “license detector”). In 2008, the FOSSology

project identified the potential in providing the analysis data for a public repository of software and license metadata at FOSSology.org.[4]

FOSSology 2

In 2012, the project released version 2 of FOSSology. A new installation package structure reorganised the software project. Furthermore, an architectural change was made with the implementation of a new scheduler which orchestrates the different scanning agents. This change also helped to design and run the agents more independently. While such changes did not bring new features to the users, the new architecture provided a more extensible structure for the FOSSology project. This followed the overall vision that FOSSology represents an analysis and reporting framework where agents as modules can be combined into a workflow running on OSS components.

Another improvement introduced in later versions of the FOSSology 2 era was the introduction of data access objects (DAO). On the first hand, the DAOs helped making database access more systematic. But with the different report formats, the DAOs also ensured consistency between the different outputs: rather than each reporting agent implementing its own query logic, all agents could call the same functions to query the database, for example, for found licenses in the uploaded OSS component.

With version 2, FOSSology evolved into a multi user Web application that covered two main trends of licensing open source software: Not only did open source software become more and more popular and awareness about license compliance increased, but also the licensing showed more forms of individual statements. Further additions in the FOSSology 2 era were about organizing uploads with tags and the ability to correct findings brought up by the scanners. FOSSology turned from a server based scan tool to a Web application for users to upload, analyse and organize OSS components for their licensing conditions.

A major change that users actually noticed was the reworked file contents view for reviewing license findings including the highlighting of text areas of license relevant statements or licensing headers in files. What sounds straight forward turned out to be a complicated programming problem: license headers or license relevant statements are usually put into comment sections of source code. At the same matching license expressions using, for example, regular expressions required cleaning the text from comment sections. Otherwise matching text areas would have been compromised by these. However, for highlighting the matched text area in the Web UI, the file contents are displayed including comment sections. Highlighted text areas would shift because of comment sections previously omitted for the matching. As such, recalculation of the exact text position was necessary. As an additional challenge, source code files can contain multiple licensing statements or headers scattered across multiple locations in a file which require a comprehensive approach to recalculation. In the end this was worth the effort, as it turned out that highlighting license relevant text areas greatly helps to quickly identify and classify license relevant parts of the file on screen.

Also n this period, a first version of SPDX generation was published as an external module by the University of Omaha Nebraska.[5] SPDX is a specification for expressing metadata about software packages including licensing information.[6] Leveraging the modular structure of FOSSology, the SPDX output was implemented as just another reporting agent. A first version of the software generated SPDX 1.1 compliant documents while a subsequent evolution of the generation plugin provided SPDX 2.0 conforming documents as output format.

4 Robert Gobeille: The FOSSology project - MSR '08 Proceedings of the 2008 international working conference on Mining software repositories
5 Matt Germonprez, Gary O'Neall, Sameer Ahmed: Tooling up for SPDX - Open Compliance Summit 2013
6 SPDX Workgroup - a Linux Foundation Collaborative Project: SPDX License List - https://spdx.org/sites/cpstandard/files/pages/files/spdxversion2.1.pdf - 2016

In the version 2 era, the project was also transferred to the Linux Foundation as a collaborative project. A discussion about this step was coincidently taking place at the time of the splitting of the HP company into two organizations.[7] Eventually the Linux Foundation offered to host the FOSSology project where it fits into the Open Compliance Program.[8]

FOSSology 3

FOSSology 3 introduced a new license scanner Monk as a new feature. This scanner finds license texts faster than the Nomos agent. Both the matching and the difference between stored license text and found text is highlighted which helps the user to quickly identify the license. Additionally, the keywords used by Nomos are highlighted in the same text view. These visual hints help in the license decision process where the results can be managed in FOSSology: Differences to reference license texts from the FOSSology database are clearly shown to the user. Another feature introduced with version 3 was the editing capability of copyright phrases found by FOSSology. This is important since there is no rule as to how to indicate copyright ownership and there is a variety of different ways that copyright ownership may be expressed. Although the implemented functionality to extract copyright notices is striving to extract only the relevant information, it is sometimes necessary to postprocess the results, mainly to remove formatting characters.

New JavaScript frameworks like jQuery and jQuery Datatables modernized the client look and feel while refactoring on the server side, such as dependency injection increased testability. FOSSology continued with technical improvements with more use of jQueryUi for a better client experience and the implementation of PHP templating using the Twig library.

Another open source project for license scanning, Ninka[9] has been integrated using a wrapper. The main idea was to have three license scan approaches in FOSSology to allow for more adaptive scanning as well as the ability to conclude licensing based on the results of these. With these license scanners integrated, another agent was added for more automation in the workflow: a decider agent allows for defining rules such as reusing decisions from other packages or automatic decisions if all scanners determine the same license. For decisions based on certain text phrases, another scan agent speeds up the process: users can select distinct licensing statements found in a particular file and apply a rule for the entire upload to alter, confirm or remove a particular license.

Another feature was a refactored SPDX 2.0 RDF file generation. Release 3.1 extended the output formats for the SPDX tag-value notation in addtion to RDF/xml. Release 3.2 added the ability to import SPDX documents from other FOSSology instances or even other software tools. Furthermore, a word processor document report was added in Fossology 3.2, which contains not only licensing information, but also summarises analysis decisions as well as scan findings. And finally, an added JSON output format increases the possibilities to export results for other applications.

Another feature area implemented in version 3.2 of FOSSology is license obligation and risk management. This feature allows for defining obligations and risks and associating them to licenses. When a report is generated, all the obligations and risks of the licenses in effect (the concluded licenses) are generated in the report, given that an administrator of FOSSology has assigned the obligations and risks to the licenses. This especially helps to efficiently deliver a component license analysis without subsequent manual editing steps.

7 Hewlett-Packard Co.: HP To Separate Into Two New Industry-Leading Public Companies, Press Release, October 6th 2014

8 The Linux Foundation: Open Complicance Program – A Linux Foundation Initiative https://compliance.linuxfoundation.org

9 German, Daniel M.; Di Penta, Massimiliano and Davies, Julius : Understanding and auditing the licensing of open source software distributions. In Program Comprehension (ICPC), 2010 IEEE 18th International Conference on, pp. 84-93. IEEE, 2010.

Last but not least, FOSSology 3 added features that reduce the time needed for component analysis and scanner corrections by reusing information from previously analysed uploads. For example, when scanning new versions of software, the analysis can be limited to the differences compared to an older version. In fact, this reuse is not only limited to conclusions or corrections of licenses on a file basis: also identified custom text passages in previous uploads can be taken over to new uploads. With this feature the manual correction time of a newer version of a software component is minimized to the actual differences in licensing only.

Technology

FOSSology is a derivative of a LAMP application. LAMP is an acronym that denotes applications that run in Linux, use the Apache Web server, build on MySQL as a database and provide a PHP-based Web UI. In FOSSology, a PostgreSQL database server is used instead of MySQL. Because of its dependencies on the Linux APIs and libraries, FOSSology cannot be easily ported to the Windows or Mac OS X platforms. However, virtual machines or docker-based builds make its use on these platforms possible today.

Database Approach

Since scanning for licenses in open source components yields large amounts of data, the use of a database is a required. PostgreSQL is available on most Linux distributions and represents a mature dependency, while allowing for portability of the FOSSology software.

In the first days of FOSSology, the reference schema was stored in the so called GoldDb. Schema changes were managed via a centralized implementation in `lib/php/libschema.php`. However, some operations cannot be represented as schema updates for an existing database. Therefore, additional steps for migration of data are required during upgrade to a new release. This support is very important as FOSSology users create a growing database of scanned source code files which should be maintained with new versions of the software. The script `install/fossinit.php` executes the correct `install/db/dbmigrate*` files depending on the release that is stored in the database and ends up in a well defined state.

While some queries would work well with other database management systems, some specialized queries rely on PostgreSQL, e.g. recursively computing full path names. The performance gain of executing the logic in the database instead of PHP justifies the dependence upon the database technology. An OR-mapper is not (yet) used, due to the large number of complex, highly optimized queries.

PHP Stack

FOSSology prior to version 2.6 did not use any PHP frameworks. The first use is found in release 2.6 which is, strictly stated, not a minor release, because it changed how PHP dependencies were integrated by using the composer package manager. Composer allows for managing libraries and their (transitive) dependencies. The dependency manager for PHP manages updates from the previous releases of dependencies, and also if the system cannot connect to the Internet. This technology change was required due to the end of life of the formerly used PEAR channel.

The transition to a modern and standardized PHP application is an ongoing process with many different aspects. The first aspect is improving testability of new components. Since PHP is used as Web frontend, the structure is continuously improved to ensure a MVC like paradigm. HTML rendering is migrated from PHP print statements to twig templates. The previously mentioned DAO objects have helped to improve security by using an abstraction for database configuration. Then, a

re-factoring aimed at separating logic from presentation and persistence layer code was started. In this presentation and persistence layer, code was replaced with open source components where possible. Most of the required refactoring has been applied from version 2.6 through version 3.1.

License Scanning

Nomos is the main license scanner in FOSSology and it is based on regular expressions. As indicated above, the text formatting and programming language specific comment characters, such as '//' (or "/*", ";;", "REM", "%" and similar variations) present a challenge for regular expressions. To circumvent this problem, Nomos uses short seed expressions to identify regions of interest. It normalizes a portion of the scanned file in the vicinity and then scans for larger snippets. After the list of matching snippets is established, Nomos determines their positions in the scanned file and the snippets are mapped to license findings.

License findings are either positive matches to known licenses with their version, or unknown licenses in the style of a known license. This design guarantees a low false negative rate, as license relevant portions of a file are identified even if the license text is not yet known in FOSSology. Currently, Nomos holds more than 3000 snippets that map to more than 650 licenses.

Apart from the regular agent mode, Nomos can be run in the one-shot analysis mode. Here a single file can be uploaded and is scanned on the fly. If FOSSology is installed, Nomos can also be called from the command line and the output can be directed to standard out for plain text processing of scan results.

One structural disadvantage of matching license relevant text findings with regular expressions is the lack of an ability to detect manipulated license text. While this topic is may be interesting from a legal perspective, custom variants of popular license texts are a problem for tool-based license scanning. One example of this problem is the use of the MIT license and the addition of one or more sentences with extra conditions. A regular expression based approach would consequently identify the MIT license, which is a classic example of a permissive license and would possibly not find any "not-so-permissive" custom additions.

For handling this case the agent named Monk was introduced into FOSSology. This agent considers the reference license text collection from the FOSSology database. Originally these texts were added to FOSSology to allow the user to review the original license texts in the UI. The Monk agent uses these texts to compare with the found text in the files of the uploaded software component. Technically, Monk tokenizes the license reference texts and the text found in a file by space or line break characters. Also common comment characters are filtered. Then, Monk computes the Jaccard text similarity index and adds a weighting to the computed index. The weighting assigns longer text matches with less similarity greater weight than shorter matches with 100% similarity. This is necessary because some longer license text includes shorter license text. If the weighting was not added, the shorter 100% match would always be preferred over longer, but not exact matches.

The obvious disadvantage of Monk is that it recognizes only those licenses which are part of the FOSSology license database. In this way, both the Nomos and Monk agents complement each other: Nomos also detects unknown licensing statements or license texts, however, with less precision. At the same time Monk can give very precise detection results for all known licenses.

FOSSology and SPDX

As mentioned earlier, SPDX represents the de-facto standard for expressing metadata about software packages. SPDX stands for "Software Data Package Exchange" and describes an initiative by the

Linux Foundation to set standards for communicating the components and their licensing.[10] As part of the SPDX effort, the specification also defines a comprehensive list of licenses with standard identifiers and specifications for report formats (cf. SPDX License List). Since FOSSology's main functionality is license scanning, supporting SPDX as a report format represents a natural step. Also the license identification uses the SPDX standard identifiers where present.

Since release 3.0, as mentioned above, FOSSology has had the ability to export SPDX 2.0 reports. Since the generated output was already SPDX 2.1 compatible, it is now also labelled to be, the more up to date, version 2.1.

The two output formats from the SPDX definition are supported. The RDF/xml (a Resource Description Format developed by the W3C) and tag-value. The tag-value format is a more human readable output and is similar to the debian-copyright format as used for Debian packages.[11] These reports represent the result of the scanning and analysis in both machine and (in the case of tag-value) human readable format. Scan results are expressed using the "LicenseInfoInFile" tag, while the analysis result is written using the "LicenseConcluded" tag.

Because the implementation of SPDX report generation uses a template library, FOSSology can also generate the well known debian-copyright files. The major difference between the SPDX tag-value format and debian-copyright is that debian-copyright aggregates files by found (or concluded) licenses while SPDX maintains a listing for each individual file. As such, SPDX documents could be converted to debian-copyright files but not vice versa.

Importing SPDX Documents

In 2017, many tools in the area of license compliance were able to write SPDX documents. Since SPDX format is machine readable it is an obvious idea, to implement importing functionality as well. However, to our knowledge, no license scanning tools (The Open Source project Eclipse SW360 can import an SPDX document to generate license documentation for products) were available in 2017 to read or import SPDX formats. This functionality serves two main use cases:

- If a party receives an SPDX document, how would the receiving party review this document? What would be required is a view where the file or directory structure is shown along with the imported SPDX (licensing) metadata similar to reviewing license scan results provided by the agents.

- If a user requires analyses of a software component, maybe the analysis results of an older version of this component would be available for reuse. Existing analysis results could be available to the public to continue working with for future versions of a software component. Importing existing analysis results helps by reducing effort when analysing new versions of a software component.

Since 2017 FOSSology has been able to import SPDX documents notated in RDF/xml to cover these two use cases. In the same manner as with agent scan results, users can use FOSSology as a tool to verify the information present in an SPDX document when applying it to an uploaded software component. After importing, the necessary workflow is simply the verification of a scan result.

Since the analysis work of a licensing situation can be very time consuming, reuse of existing analyses represents an important capability to reduce effort and avoid duplicate work. FOSSology

10 SPDX Workgroup - a Linux Foundation Collaborative Project: SPDX License List - https://spdx.org/sites/cpstandard/files/pages/files/spdxversion2.1.pdf - 2016

11 Debian Project: Machine-readable debian/copyright file https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/ - 2017

servers can exchange analysis data between each other and FOSSology can exchange analysis information with other license scanning tools, allowing for general reuse between tools.

Conclusion

FOSSology helps to bring clarity to open source licensing, and also supports the adoption of open source software while respecting the intentions of the authors expressed through their licensing. FOSSology itself is licensed under the GPL-2.0 and hosted by the Linux Foundation. Therefore, it matches the slogan "Open Source Compliance with Open Source Tools".

OSS license compliance tooling shall be available to all, including universities, individuals, OSS projects and companies. It should not be the privilege of larger organisations or companies, which can afford to purchase licenses for commercial tools. Since the source code of FOSSology is available, it can be analysed and - if desired - be improved. FOSSology provides full transparency, which improves confidence within the context of license compliance work.

FOSSology has now existed for more a decade. During this time, FOSSology has undergone major renovations in its architecture to keep pace with common technical evolution. It has been improved in the relevant areas of OSS license analysis, such as more precise review functionality, more scanning and detection functionality, automation of conclusions, data exchange using the de-facto standard SPDX and a more modern UI.

FOSSology implements precision, enables workflow and allows its users to review, approve, and correct the results the agents have produced. All these capabilities are required for achieving OSS license compliance.

Although licensing found in OSS components is still heterogeneous and sometimes is expressed in very special ways, its standardization is underway, for example, the Reuse project as proposed by the Free Software Foundation Europe.[12] FOSSology will follow this trend by further automating the license recognition of these standards so that the effort required for manual review is reduced while keeping the high precision and certainty of its license recognition.

Acknowledgement

The authors would like to especially thank Paul Guttmann for his support.

About the authors

Oliver Fendt *has more than 16 years experience in open source software, its license conditions and how to comply to the different licenses. During this time he kicked off several initiatives, among these initiatives are the open source project SW360 and the sponsoring of considerable contributions to the open source project FOSSology. He has developed different trainings about open source software and how to achieve license compliance and has given OSS compliance trainings since 2008.*

Robert Gobeille *is the creator of FOSSology and the original project leader. He works currently in projects with nexB.*

12 Free Software Foundation Europe e.V. (FSFE): REUSE Initiative https://reuse.software - 2017

Maximilian Huber *is a consultant at TNG Technology Consulting Max spends most of the time to develop and support the Linux Foundation project FOSSology and the Eclipse incubator SW360.*

Michael C. Jaeger *is one of the maintainers for the FOSSology project and SW360 (available on Github), both in the area of OSS handling w.r.t. license compliance and component management. At Siemens Corporate Technology in Munich, Germany, Michael works in several roles as project lead, software architect, trainer and consultant for distributed systems, server applications and their development with open source software.*

Johannes Najjar *is a Senior Consultant at TNG Technology Consulting GmbH. He has a background in high energy physics and currently focusses on IOT and Cloud Computing.*

Kate Stewart *is a Senior Director of Strategic Programs at the Linux Foundation responsible for a portfolio of open source projects and standards. With almost 30 years of experience in the software industry, she has held a variety of roles and worked as a developer in Canada, Australia and the US. For the last 20 years she has managed software development teams in the US, Canada, UK, India and China, and focused on delivery of open source based products from Freescale, Canonical & Linaro.*

Steffen Weber *is a software developer with background in algebra and numerics. High ranking in algorithmic competitions favors the focus switch to IT after the PhD in mathematics. Since 2013 he worked as full time developer for projects in certain languages with different frameworks.*

Andreas Würl *has worked for more than seven years as an IT consultant at TNG. His main focus is participating in and improving the agile software development process mainly with sustainable design and architecture. He practises a variety of programming and confguration languages and enjoys contributing to open source software in his free time.*