Package Review as a Part of Free and Open Source Software Compliance

Martin von Willebrand[a], *Mikko-Pekka Partanen*[b]

(a) Attorney, Partner, HH Partners, Attorneys-at-law, Ltd; Chairman, Validos ry; (b) Attorney, HH Partners, Attorneys-at-law, Ltd; Open Source Specialist, Validos ry.

DOI: 10.5033/ifosslr.v2i1.37

Abstract
Free and open source software ("FOSS") package review is an essential part of license compliance when businesses take into use FOSS. This article discusses the practical process of package review and the legal questions that arise and conclusions that can be made. Furthermore this article presents the process and a number of legal conclusions applied by Validos ry, an association for performing package review and sharing its results. The purpose of presenting a particular process is to share and improve the applied methodology with a long-term vision of unifying the expectations for package review and license appraisal, thus contributing to the ease of taking into use of FOSS by businesses.

Keywords
Law; Information Technology; Copyright; Licensing; Free and Open Source Software; Compliance; Sharing

Info
This item is part of the **Articles** section of IFOSS L. Rev. For more information, please consult the relevant section policies statement. This article has been independently peer-reviewed.

1. Introduction

Free and open source software compliance processes aim to enable compliant use of Free and Open Source Software ("FOSS") packages. This article addresses a part of the FOSS compliance process from a perspective of a company and also, to some extent, by a group of companies. The part addressed is package review. In order to achieve compliant use of FOSS, the package review process must identify and record the correct package, identify and record all applicable licenses and their obligations and to some extent copyright holders, identify eventual license

incompatibilities and report all of this information in a manner that allows compliant use – even correction of incompliances – in the needed use scenarios. This article's viewpoint to these objectives, is the process through which the objectives can be achieved: the process to identify a package (section 2), the process to inspect such package and its licensing (section 3), the legal conclusions used in appraisal of licensing (section 4), reporting and storing or even sharing the results of the review (section 5) and suggesting corrective measures for non-compliant packages (section 6).

Open source compliance is a wider question to which the package review process belongs. For both a strategical and organizational view on open source compliance, an overview with practical examples is presented by Richard Kemp in his article Towards Free/Libre Open Source Software ("FLOSS") Governance in the Organisation.[1]

1.1. Purpose of this Article

The writers of this article perform compliance work for Validos,[2] an association established for performing compliance review work and sharing the results between all participating companies. The compliance review reports created by Validos are stored in a joint and growing database in a manner that enables reuse by all member organisations,[3]

The purpose of this article is not only scholar, but also practical. This article presents package compliance review processes used by Validos[4] for the purpose of sharing information and also opening up a documented compliance process for criticism and therefore improvement. With criticism and improvement suggestions, this article can be developed into a robust and practical guide on legal package review in open source compliance. We invite all readers to participate into such development.

1.2. Scope and limitations of the Article

A package, as used herein, means typically a single identifiable file that is offered for download by

1 KEMP, R.. Towards Free/Libre Open Source Software ("FLOSS") Governance in the Organisation. International Free and Open Source Software Law Review, North America, 1, dec. 2009. Available at: http://www.ifosslr.org/ifosslr/article/view/19/51. DOI: 10.5033/ifosslr.v1i2.19.

2 Validos ry (http://www.validos.org) is an association based in Finland with 12 member companies representing a turnover of over EUR 1 Billion.

3 Validos shares all reports on open source packages between all members. The basic logic is that each member participates on at least a certain level (depending on the member revenue) and that the results of all of the work is shared via an extranet. An important element in Validos is that the review process and the reporting has been geared to support reuse in different use scenarios – at the same time this means that members still have need for member-specific decisions on, *e.g.*, linking questions in relation to member's proprietary software. The Validos database grows via the requests for review by members. Currently the database holds reports on more than 200 FOSS packages, and approximately 4 new packages are added per week. The reuse rate of new review requests varies between 0 - 75 %, meaning that at best 75 % of the packages in a review requests can be obtained from the database with no new work required.

4 However, even if most of the process description reflect Validos processes, this is not an exact description as the process is continuously developed and also member preferences affect the compliance work of individual packages.

an open source project. A package may come in tar.gz, zip or other compressed format.[5] As a compressed format, a package may include any number of files and subdirectories, as determined by the project offering such download.

Package review is not limited to pure FOSS packages. This is due to practical reasons: packages tend to contain files or subdirectories or other elements that are not open source, according to definitions by the The Open Source Initiative (OSI) or the Free Software Foundation.[6] Although some files may not be pure FOSS, their use might anyhow be relatively unrestricted from a corporate perspective, and therefore, their use is often controlled by the same process as the use of FOSS. From a company's perspective the compliance process can be similar or same regarding all software packages that can be obtained without charge from internet sources, such as FOSS, public domain software or freeware.

An open source project can be either a group of individuals, a separate legal entity or a part of the activity of an existing company, or even a mixture of these. In this article, we use the term "open source project" to describe these. We refrain from analysing eventual differences resulting from the type organization of the project. However, compliance review is done in relation to the copyright holder and eventual license grant by the copyright holder.

Package compliance review results in information that is generic and may be used by many companies and may also result in information that is specific to a use case, and as such may not be used as well by others. Since the generic review results are useful to many, its creation can be done in a collaborative fashion. However, reuse by many poses requirements on the review process. One perspective in this article is satisfying such requirements and enabling sharing of the generic elements of review results. Also, this article concentrates on the generic elements of the review process and not on the specific questions, such as linking with member-specific software.

This article aims not to discuss eventual risks in non-compliant use of FOSS packages: it aims to discuss a part of the process for ensuring compliant use of FOSS packages. Further, we do not intend to thoroughly or orderly discuss methods of analysing and assessing risks in relation to use of FOSS packages or licensing uncertainties, although some parts of the article touch risk appraising questions.

This article only discusses use of FOSS packages by companies in relation to redistribution of the FOSS package by the company, and not other use scenarios (such as use for a commercial service or internal use). The boundaries of redistribution are not discussed. This article does not discuss possible liability questions in collaborative production of compliance information.

We have refrained from analysing compliance review questions from a perspective of any single jurisdiction. Traditional legal sources do not address the practical questions of package compliance review: at least we have not found such information from any jurisdiction. At the same time, the companies need to apply the review results in multiple jurisdictions in a unified manner or at least with only small variations. In defining the legal conclusions we present here, we have assumed

5 For example Unix source code of Apache http server version 2.2.15 is distributed in package httpd-2.2.15.tar.gz.

6 The definition of open source by the Open Source Initiative: http://www.opensource.org/docs/osd (retrieved on 4 May 2010) and the definition of free software by the Free Software Foundation: http://www.gnu.org/philosophy/free-sw.html (retrieved on 4 May 2010).

that the general principles of copyright and norms in interpretation of licensing or similar texts, are similar in most jurisdictions. To the extent this is not so in relation to the legal conclusions we present, we would be delighted to be corrected. The writers of this article are Finnish lawyers and thus come from a European continental, and more specifically Nordic, background.

1.3. Methodology

From a perspective of legal methodology, what is the value of publicising a package review custom, even a developing one? We have noted that current legal literature does not much address very practical questions that need to be addressed in package review. Also, we see it highly improbable that such legal conclusions that we have come into would be determined by an authoritative source, such as a court, any time soon. Taken into account the amount of jurisdictions and the amount of different conclusions, this is evident. Thus, in lack of a way to establish a correct package review methodology by investigating traditional legal sources, we have decided to strive for the development of a consensus by the legal community interested in FOSS and thereby also the wider legal community engaged with open source projects. If such a consensus would be established, the practical risk of non-compliance would be lower, since if a company adhered to the consensus, the probability that a right holder in an open source project would require a company to interpret licensing differently, would be lower.

1.4. Introduction Validos FOSS Review Process and the Legal Conclusions Used Therein

The Validos process returns a compliance value for a package which is quite simple: a package is found to be (i) compliant or valid, (ii) possibly incompliant or (iii) incompliant /containing clear risks.[7] Compliant means that the licensing of a package was found clear and no incompatibilities within the package were found. Evaluation against the redistribution license of the member is out of scope (a member-specific question that is not part of a generic compliance process: that information has little value for reuse in other use situations). In addition, one outcome of the process is the use instructions for redistributing the package and other reports (section 5) and possibilities to correct found possible or clear non-compliances (section 6).

FOSS packages are often licensed in ways that are not clear or unambiguous. The process described in this article has been used within Validos on more than 200 packages containing thousands of sub-packages. Of those packages 65% have been found fully compliant in accordance with the process.[8]

This percentage includes a set of legal conclusions applied: we have deemed that certain typical situations are considered compliant, as long as defined criteria are fulfilled and contrary

7 In Validos, each package receives a value “Compliant”, “Possible Incompliance”, “Clear Incompliance” for every standard use case (redistribution, commercial service, development tool and internal use). This is viewable as one line information. We also use a marker to signify that decisions are required (*e.g.*, a GPL-licensed package can be fully compliant but will anyhow require a decision regarding linking by the user of the package). The first level report will then offer a risk pointer with short explanation of the reason for the given value, *i.e.,* “Why was this package tagged with Possible Incompliance?” and will also detail the files and folders affected. The second level report expands the first level report with all the details of the review process.

8 The percentage does not reflect the gravity of the non-compliance: a single file in a 25,000 file package may create a (possible) non-compliant tag: it is the purpose of the use instructions to reflect the gravity and point to the risk in question. It can also be very easy to correct a non-compliance.

indications are not found. The legal conclusions applied are presented and discussed in section 4.

Legal conclusions could be also made on an *in-casu* basis. However, due to the frequent occurrence of most of the situations meant in the conclusions, we assert that most companies would do well to form a policy on these questions. Leaving every question to be individually appraised by an open source review board will not result in better compliance decisions, but rather ineffective working methods and unnecessary variation.

It is to be noted that the approach described aims to enable companies to utilize free and open source software. Instead of an approach to enable companies to use open source software, a company could adopt a very risk averse approach, and not, *e.g.*, use any files which do not contain a clear license header due to perceived licensing ambiguities. This would lead into non-use or a very limited use of open source packages by such company. Thus, we acknowledge that not every company may be willing to accept these conclusions as a basis for their compliance work.

A number of questions will still need to be answered separately from the conclusions described in this article. Typically these questions are submitted to an open source review board, or similar ad-hoc formation including an engineer, a process controller and an open source knowledgeable lawyer. These situations include:

1. Situations in which the legal conclusions do not apply, *e.g.*, when the process finds a or possibly incompliant package; and
2. use-case specific decisions, such as questions on interaction with proprietary software (*e.g.*, linking).

2. Identifying a Package

Objective: When a compliance review request is received the first compliance task is to identify the package described in the request. Correct identification of the package ensures that the compliance work is done for the correct package and that the information can be later reused by others: others need to be able to match the reviewed package with the package they are planning to use.

Description: Typically an open source project has different versions of the packages it offers, available perhaps on different hosting sites, project pages or source code management systems. This can lead to uncertainty regarding which is the right software package to review, especially in situations where the review is performed by a specialised unit or it is performed much later than the actual software has been taken into use by a company. To avoid such uncertainty and correctly establish the package to be reviewed, the reviewers should note at least the following different versions:

Different Versions / Uncertainties	Comment
Wrong Project	Sometimes another open source project may have a confusingly similar name, or there might be an old, out-dated and no-longer used web-page of the same project.
Sub-Projects	An open source project may be divided into multiple sub-projects
Larger Packages v. Smaller Packages	Packages may be offered in versions which include different combinations of the software of the project or third party software. A typical example is a version which includes all required dependencies compared to a version with only code created by the project. Also, sometimes projects provide versions which do not include certain code (*e.g.*, a patented cipher).
Platform Variation	Packages may differ for different platforms (such as Debian, Red Hat, Windows etc.)
Binary v. Source	Packages may come in versions including only source code or only binaries or both. Sometimes the source version is more encompassing than the binary version (*e.g.*, source is provided for all platforms) or vica-versa (*e.g.*, the source is only provided for the code created by the project and dependencies are only in binary format).
Development Versions	Finally, most projects have different development versions, *e.g.*, versions running from 0.1 to 2.72. Development versions can also be indicated with letters 1.0a etc.

Table 1: Possible uncertainties

Recommended process: Review requests should contain at least the project name, its web page and the development version of the package. File name and URL of the package to be used might not be sufficient, as the user might refer a binary only package and review should be performed on source version or both the source and binary version. This is why the project name and the web page of the project are always needed.

Review requests sometimes refer only binary packages: if this is the case, one critical element is to find the source version matching the binary version. If matching is not clear and it cannot be cleared with the unit requesting the review, the remaining option is to inspect source files and compile the binary to be used from the reviewed source. In this case, only the source version is reviewed.

In practical terms, the information in the review request will need to be assessed against the information provided by the project, in order to establish the correct package to be inspected. Possible discrepancies need to be solved with the company/unit/project requesting review of the package. If request includes partial information, eventually completed information should be sent back to the requesting unit for verification.

As a part of identifying the package, the inspected file should be stored and a unique signum for

the file should be created (such as an MD5-signum). At best, the unique signum is provided by the project and that signum can be checked against the signum created for the stored file. If the projects use a signum, the same signum type can be used for the stored file. In addition, the preferred signum type should be created in any case so that the stored files have one unified signum for each file.

3. Review of Package

3.1. Collect Information

Objective: After a package has been identified in accordance with the previous section, the actual review of the package begins. The objective of the inspection of a package is to collect all information that is relevant for the compliant use of the package and to analyse arising legal questions.

Recommended process: The inspection starts with manual inspection of the web pages of the project and the downloading of the identified package. The web pages are also inspected for license information and any related material such as statements regarding known patent issues or export restrictions. Occasionally additional copyright, license or author information needs to be searched via search engines from other public sources such as related mailing lists. Found license information is recorded for archival purposes by taking a screenshot or printing an electronic copy.

When the package is downloaded and archives extracted, the package is briefly inspected to form an overview of the included folders, documents and libraries. Validos process also includes uploading the package to FOSSology source code analysis software after the package has been downloaded. FOSSology is an open source licensed tool that can be used to analyse source code. At the time, the most useful feature of the software is its ability to find license text matches from the source code of a package. This is done remarkably well as the software identifies reliably also license fragments and modified license texts.[9] We have found that the amount of licenses not found is very low.[10]

There are also other source code analysis software tools, which can be used in open source compliance processes.[11] Sometimes it is necessary to use text search tools such as grep to find and collect copyright and license notices from large code bases.

The practical review of Fossology results can be performed as follows:

1. .Overview of the Fossology Results. When Fossology has processed the uploaded software package, it displays result of the check as a list of found matches. The results should be briefly examined for the purpose of forming an overview of the included licenses and phrase matches.

9 For more information, see http://fossology.org.

10 The current road-map of FOSSology includes the reporting of copyright notices found in packages, to be released in version 1.2, probably very soon. This feature will add to the review certainty, as unidentified licenses can be picked up by their copyright notices.

11 Most known alternatives are different offerings by Black Duck, Palamida and OpenLogic. There is also another free tool, OSLC – or Open Source License Checker (http://sourceforge.net/projects/oslc/, link retrieved 2 May 2010), which is not much developed currently.

2. Review of Phrases. If Fossology finds suspicious text matches that do not correspond with any known license text, it can point them out as phrases. As these findings can potentially refer to proprietary type licenses or other restrictions, the matches need to be reviewed. This is done by reviewing the preview view for the match "phrase", or if necessary, by reviewing each file that contains the spotted phrases.[12]

3. Review of License Text Matches. Fossology scans the uploaded file against license texts in its knowledge base. However, the listing shows only textual matches and not legal matches: e.g., a file with a dual license will probably be shown as a hit with two different licenses.. Manual review of the results is therefore required with help of the interface provided by Fossology. The findings should also be reviewed to check that the matches corresponds fully with the stated licenses and eventual license modifications are found. Most projects luckily employ similar statements so that each file does not need to be inspected separately.[13]

3.2 Analyse Collected Information

Objective: After collecting data from project web pages, documentation and source code it must be analysed. The objective is to identify the level of clarity of licensing and eventual incompliances and other issues.

Recommend process: At Validos, we review at least:

1) Main License Clarity

How strong and reliable is the information on the license applied by the project (we refer to this license as the main license). According to our experience it is not always clear what the main license is. Typically these situations are related to contradictory or incomplete license statements in project web pages and downloadable packages.[14] Sometimes these can be solved satisfactorily by finding a common nominator in the package: (*e.g.*, unclear references to GPL 2 and GPL 3 licenses on the webpage can be solved, if the package thoroughly refers to "GPL 2 or later"). However, in many cases solving in-clarity regarding a main license requires contacting the open source project or the relevant author. If the reviewer needs to make a judgement on the license, the package receives a "possible incompliance" tag from the review process.

2) Compliance of Existing Sub-Packages or Sub-Components

One review item is to find and list the existing sub-projects or third party projects and the

12 Preview view of Fossology displays a one or two row excerpt of found phrase, in many cases this is enough to determine whether the found match is relevant in a licensing sense, or refers to non-compliant license. If a preview is not enough to resolve whether or not there is really any issues, the files source code can be accessed from the tool reviewed in detail.

13 For example the tool can show that inspected package contains 10 000 files with match "'GPL v2'-style ", while it is not effective to check each of 10 000 match for unexpected modification, at least some source files should be reviewed. Should any inconsistencies be found, the findings should be reviewed in detail. A feature that separates different types of matches would make this process faster.

14 For example, situation where a project web page contains a statement "Licensed under the GPL." where word GPL contains a link to Free Software Foundation's GPL license page, which nowadays contains the version 3 of the license. At the same time a download package can contain a statement "Licensed under the GPL v.2 only".

applicable licenses (sub-licenses).

Typically FOSS packages include code created by others than the main copyright holder. While code reuse is one of the driving forces of open source development, it is also a common source of legal risks. This is caused by the sheer number of licenses (whether open source or more limited licenses) that are not compatible with other licenses, which combined with the fact that developers tend to be more interested of coding than licensing, causes often situations where some included sub-licensed files or components are not compatible with the main license.[15] Therefore, one of the main tasks of a package review process is to point-out any situations where all license requirements cannot be fulfilled simultaneously when the software is distributed. Equally important is to find files that may not be distributed at all, such as components licensed only for evaluation use. License compatibility checks are done by reviewing stated license information and results of source code analysis. If clearly or possibly incompliant licenses are found, corresponding components are reported with necessary detail, usually at folder or file level and the report summaries receive a corresponding value. Additionally, when discrepancies have been found, corrective measures, which can be used to solve the issue or mitigate risks caused by problem, can be suggested.

3) Other Elements such as Patent and Export Control Related Information

As a note, information that relates to patents or eventual export control related questions, can also be collected.

Occasionally, license problems can be solved by contacting open source projects for clarifications. We have found this approach to be welcomed by projects and in many cases projects correct or clarify issues not only in their replies but also clarified the information provided by the project. We see contacting of projects as a way of contributing back to free and open source projects. The findings of the review process should always be recorded in a format enabling quality control, sharing and reuse.

4. Legal Conclusions in Appraisal of Licensing

As we mentioned above, the Validos process returns a compliance value for a package which is quite simple: a package is found to be (i) compliant or valid, (ii) possibly incompliant or (iii) incompliant /caining clear risks. In addition, one outcome of the process is the use instructions for redistributing the package (section 5 below) and possibilities to correct found possible or clear non-compliances (section 6 below). However, FOSS packages are often licensed in ways that are not clear or unambiguous.

15 In pure package review, as the one described in this article, the incompatibility is assessed within a package or a combination of packages. The question of license compatibility in relation to proprietary or other software of the member organisation is not a part of generic package review. That question becomes member-specific and cannot therefore be shared with other members (the answer to a member-specific question also has little value to others, or little value for reuse in general).

4.1. Files with No License Headers

Issue: Packages contain files with no license headers. Under which license should these files be considered to be licensed?

Conclusion: Files with no license headers are considered to be licensed with the closest main license, as long as there are no other indications. *e.g.,* a library or folder within a package may contain an open source license and 10 files of which the most important one contains a license header and the rest do not have any license header. In this case all the other files are considered to be licensed with the "main license" of that folder, unless there are contrary indications.

Typical Contrary Indications: A copyright notice by a third party that differs from the copyright notice of the rest of the package and there is no indication of that party participating in the same open source project. Statements on proprietary licensing, such as "Copyright ATT 1989. Proprietary and unpublished".

Discussion and arguments: This is the widest question in package review. Most of the packages include files with no license headers. It can be envisaged that bigger projects will embrace detailed policies and licensing practices which solves this question at the source,[16] but the amount of projects will continue to increase and this issue will persist. Companies taking into use FOSS packages will need to resolve this question somehow. Small to medium size projects mostly do not see this as a problem and in lack of a unified approach to offer to projects, companies will mostly need to resort to policy decisions on this.

The conclusion we propose seems to offer a practical solution to this question. The weakness of the argument is that the files do not contain any license headers and the conclusion seems arbitrary. However, it must be noted that there is no widely accepted instruction or practice to include a license notice in each file and one could also ask why not include a notice for each line of code. A notice can equally be placed on the folder level, as it can be placed on a file or package level. Furthermore, we are not aware of any legal obligation in Finland or elsewhere to include license notices on a particular granularity, such as at a file level. We deem this conclusion to reflect most authors' intent taken into account the practice in placing license notices. On the other hand, contrary indications need to be reviewed (see above).

4.2. Modifications to Files

Issue: Files may at times contain notices that they have been modified by another party than the original creator. In many of these cases, there is no license reference in relation to the modification. However, the file may contain the original license notice or references of the original author. Under what license should the modifications be considered to be done?

Legal Conclusion: Modifications to files are considered to be under the same license as the rest of the file, unless otherwise is indicated.

Contrary Indications: In most cases, only a reference to a second license or a statement on other

16 Such as the SPDX initiative hosted by Fossbazaar, a working group of the Linux Foundation (https://fossbazaar.org/content/fossbazaar-face-face-meeting-lf-collaboration-summit, retrieved on 4 May 2010)

type of license (such as a statement referring to proprietary type of licensing) is a contrary indication.

Discussion and Arguments: Since the author of the modifications has not expressly stated a license, it can be asked how is his intent to license the modifications expressed. When reviewing individual files this question may be affected also by how the statement of modifications is formulated and how it is placed in the file, in relation to the existing license reference. Possibilities include placing the modification statement and eventual copyright notice before or after the existing license notice, to include it into the existing license notice or to state it as a comment later on in the file. However, in each of these cases, we conclude that the intent of the author is expressed by the fact that he did not add another notice or reference to another notice. In fact, this same argument applies even if the file does not contain notice of its own, but rather a main license is applied (see legal conclusion on Files with No License Headers).

4.3. Licenses Do Not Automatically Change or Automatically Attach

Issue: Many times an open source package that includes GPL licensed files includes also files with other licenses such as MIT and BSD. The GPL license (both in version 2 and 3) requires that a whole is licensed under the GPL license. *E.g.,* MIT-licensed files are considered GPL compliant since it is possible to fulfil both the requirements of the MIT license and the GPL license, at the same time. However, the practice with open source packages is that licenses are not changed or added onto each other, in the file headers. This means that a MIT licensed file within a GPL package continues to contain only the reference to MIT license, and open source projects and their redistributors, do not add GPL license references to these MIT licensed files. The question is whether the license of the MIT-licensed file has changed into MIT+GPL due to the inclusion of the MIT file to the package containing GPL-files. This question has relevance *e.g.,* in cases where a company wishes to use only the MIT-licensed files and wishes to remove the GPL-files. Are the files still licensed with just MIT or should they be treated to be licensed with both MIT and GPL? Does the license change automatically from MIT to MIT+GPL? Or in case the package contains internal incompatibilities, such as Mozilla Public License files and GPL-files forming a whole in copyleft sense: can such incompatibility be rectified by removing the GPL-files?

Conclusion: Files are considered licensed with the information contained in the file, to the extent there is no information to the contrary. The existence a GPL-file in the same package is not contrary information. Licenses of files are considered not to have changed (or not to change automatically) when the whole package is licensed with another license or contains files licensed with another package, even if the license of the file would allow addition of new conditions or new license.

Contrary Indications: Additions to license headers to support an imposed additional license by the project or the redistributor.

Discussion and arguments: It would require interpretation to deem a file containing one clear license statement to be considered licensed with the stated license and another license, in a cumulative manner. However, in case a MIT-licensed file is contained within a GPL whole, it could be argued that the GPL redistribution requirement (copyleft) implies that the MIT-licensed

file has been redistributed under the GPL and under the original MIT license (both licenses' requirements applying to the same file on subsequent redistribution). This is not reflected in practice in any way: we have not seen any license headers with license additions of this type. However, even if relicensing MIT-files without the GPL addon, could theoretically be considered to be incompliant relicensing of the GPL-parts, it is not very probable that the right holder of the GPL-part would be interested in enforcing the GPL against this type of behaviour since there is hardly any interest in doing so and this type or relicensing is very common. If the GPL-elements are removed from such a package, we would deem it very strange, if the right holder of the GPL-parts could thereafter exercise control over the relicensing terms of the elements that originally were by another party and under another license. This argument applies regardless, if the elements were licensed with a GPL compliant license, such as MIT or a GPL-incompliant license, such as the MPL. Thus the conclusion is that GPL-elements can be removed from a package to allow *e.g.,* linking with GPL-incompatible packages and also, as the conclusion is founded on licenses not attaching automatically, also package incompatibilities can be fixed by removing elements that cause incompatibilities, at least in cases where incompatibilities are caused by GPL licensed elements.

4.4. Software Copyleft in Relation to Firmware

Issue: Firmware files are at times distributed together with non-firmware software with the intention that the firmware files are run on a separate device and the software is intended for running on a computer processor. May firmware files form a whole, in GPL sense, with software intended to be run on a computer processor, outside of the device containing the firmware? Is there a possible incompliance question in cases where GPL-software running on a computer processor interacts with proprietary licensed firmware?

Conclusion: Firmware, which is intended to be placed on hardware, is separate from a software intended to be run on a computer processor. As such it does not form a derivative of software intended for running on a computer processor.

Contrary Indications: A clear statement by the right holder or licensor of the GPL-licensed software. Even this indication does not cause a clear incompliance, but rather a possible incompliance, since it can also be argued that an attempt by a GPL-licensor to control redistribution of firmware elements, is not effective in a copyright sense.

Discussion and arguments: These series of instructions (firmware v. traditional software) are distinctly separate. The question of firmware files containing mixed code (GPL and proprietary) within the firmware is outside the scope of this legal conclusion.

4.5. Autoconf and Other Build Tools

Issue: Build or similar tools that are licensed with a GPL license are widely used and included in open source packages. The question is whether the copyleft obligation contained in the GPL license should be considered to form a whole (as meant in the GPL) with the rest of the files in the same package. In most cases the rest of the files are also output of such tools, *i.e.,* built with such tools. This question applies to GNU libtools and GNU autoconf tools and Bison parser files

Conclusion: GNU libtools and GNU autoconf tools (and Bison parser files), when contained in packages, are assumed to be used as build tools, unless there is indication to the contrary. GPL-licensing of build tools is considered not to pose requirements to the license, as regards distribution of the rest of the software built with those tools, even if the tools are contained in the same package.

When a file contains the autoconf-exception.[17] the exception is applied, if there exists, in the same package, a file that states "generated by autoconf"[18] (it is not necessary to check whether the file actually is generated by autoconf, the statement is enough).

The Bison exception,[19] if it exists, is applied if there are files that state "made by GNU Bison" and the version of Bison 1.24 or higher. While the wording of the exception sometimes refers only to "use", it is concluded that it means to allow all exploitation rights granted by copyright (copying, modification and publication).

Contrary Indications: Typical contrary indications are other GPL-licensed libraries included in the package and the output of the build process. Also, if the software package would be build tool in itself, then this would be a contrary indication.

Discussion and arguments: Build process can be considered legally as copying of the source code and other elements into a slightly different format as object code and other code organised for execution by a computer. Object code could be considered as modified version and as such a derivative, but as it it a mechanical process that does not normally include human creativity, we would consider the object code to be a copy of the source code. Similarly other parts copied in the process are copies. A similar copy would be an analogue piece of music as a digitized copy. Although the build tools may be and probably are works of authorship, the same applies to build tools as any other computer software: output obtained by using them is not subject to the copyright of the computer software (unless elements are directly copied, which is a contrary indication). In some cases, the instructions given to the build tool could be considered creative, but this is similar to other code given for the build tools for processing, such as the source code. Thus instructions, and files containing instructions can be treated similarly as the source code. In the end, the build tool's license does not affect the license of the code processed by the build tool.

4.6. Dual License

Issue: Many open source packages refer to "dual licensed" files or packages. Many times the wording "dual licensed" is explained to mean that the licensee may choose either of the stated licenses, but also others expressions exists, *e.g.*, "dual licensed with MIT and GPL" or licensed with "CDDL+GPL" with references to "dual license".[20] These latter could be interpreted to mean

17 For example: "# As a special exception to the GNU General Public License, if you distribute this file as part of a program that contains a configuration script generated by Autoconf, you may include it under the same distribution terms that you use for the rest of that program."

18 For example, a file named "configure", which contains text "# Guess values for system-dependent variables and create Makefiles. Generated by Autoconf 2.52."

19 For example, " /* As a special exception, when this file is copied by Bison into a Bison output file, you may use that output file without restriction. This special exception was added by the Free Software Foundation in version 1.24 of Bison. */"

20 For example, see http://wiki.java.net/bin/view/Projects/GlassFishCodeDependencies (retrieved on 2 May 2010)

that both of the licenses need to be applied. Sometimes several licenses are referred to with a separation using the word “or”. How should not clear references to “dual license” be interpreted?

Conclusion: We conclude that the wording “dual license” or use of “or” means that the licensee may choose between the licenses offered, unless there is contrary indications.

Contrary Indications: A contrary indication is an explanation of another type of licensing scheme than a pure dual license where the licensee may choose the applicable license.

Discussion and arguments: The statement “dual license” is also sometimes used to refer to an offering, where obtaining a second license requires payment of a license fee (*e.g.,* a proprietary like license with no copyleft obligations instead of a GPL license, in exchange for a license fee). Except for this situation, we feel that every project using some kind of statement of “dual license” means that the licensee may choose between the licenses. Sometimes the dual license choices are also incompatible with each other, such as the Mozilla Public License and the GPL: in these cases, the theoretical assumption of licensing with both licenses, would not be possible due to that the requirements of these licenses cannot be simultaneously satisfied, and thus a project hardly would require such a license scheme from its users.

4.7. Short License References

Issue: It is not uncommon that FOSS packages or files just refer a license, without containing the actual license text. In these situations it is not necessarily clear what is the applicable license that must be complied when the software is redistributed. *E.g.*, many files refer to a MIT-license without clear definition of the MIT-license. Which MIT-license should be applied? This issue does not refer to ambiguity in license version numbers in cases where there are clear license texts and license versions, but rather to licenses which are more varying (mostly MIT and BSD).

Conclusion: If the license text is not provided, the applicable version is that which is provided by the project that has introduced the respective license. If there is no such project or organization, or it is likely that such initial publisher is no longer maintaining the license, the source of the license text is Open Source Initiative’s list of approved licenses. *E.g.*, the MIT-license text, if not otherwise indicated, means the MIT-license text approved by the OSI (www.opensource.org).

Contrary Indications: Any reference by the right holder or the project to another type of license.

Discussion and arguments: This is really a practical assumption, not necessarily a legal conclusion. Still, it quite probably results in a license and license content accepted by the right holder. The license contents in different MIT-license versions are, from a risk assessment perspective, quite similar: all allow copying, modification and redistribution, so any risk would relate to lesser obligations. A right holder requiring remedies based on application of a certain MIT-license not specified by him, might also have difficulties in such claim. Of course, one could theoretically argue that there is no license.

4.8. GPL and LGPL Version Incompleteness

Issue: Many times projects refer (at project pages, root of the package or source files), in an incomplete manner to licenses and do not state the version of the license, or the information is contradictory. Typically this occurs between GPL version 2 and 3 and LGPL versions 2/2.1 and 3. Which license version should be applied? What if the project cannot be contacted and it is inactive?

Conclusion: When there is incomplete information regarding a license's version, a single point (*e.g.,* source file) defining the license version completely is enough, provided there is no conflicting information. If the version is totally unspecified in every place, then the rule on all LGPL and GPL license versions applies: user may choose the version of the given license. If there is no single point that defines the license version, and the project web pages refer to http://www.gnu.org/licenses/gpl.html and the date of the package is earlier than 29 June 2007 and the project is inactive (does not reply to queries), then we consider GPL version 2 to be the correct license.

Contrary Indications: In relation to a single point establishing a license version, any contradictory reference to another version will create a possible risk.

Discussion and Arguments: Regarding references to http://www.gnu.org/licenses/gpl.html for packages earlier than 29 June 2007, it could be said that the project might have earlier referred a completely different license, but this is quite improbable. The best way to solve this question, is to ask the project, but sometimes the project is inactive. Inactivity of the project supports that the project has not intentionally changed its license.

4.9. Source Code as Documentation

Issue: Several licenses require provision of copyright, license and similar notices in the documentation to the end-user. How can this be fulfilled in outbound licensing, *i.e.,* what are the exact requirements of the licenses of the packages to the redistributor? Is it enough that the notices are provided in electronic form and can they be provided as a part of the source code? Is it enough that just source code is provided to the licensee / end-user?

Conclusion: Provision of source code to the licensee / end-user fulfils the requirement to provide the copyright, license and similar notices to the licensee / end-user.

Contrary Indications: Clear text to suggest different method of provisioning the notices.

Arguments and Discussion: Notices are contained in the source code. Typically provisioning of the source code is considered as providing the end-user more than just the notices. Thus, if the redistributor provides the end-user the source code containing the notices, the notices are provided to the end-user. It can also be discussed, whether the source code, when delivered like this, is documentation or not. Source code can also be considered as part of documentation, since it provides detailed information on the functioning of the software, its authors and licensing. Also, separate notice documents are not very useful and tend to become lengthy, uninformative documents, and we are not sure whether the right holders wish such practices to be undertaken. As

an additional point, we have not seen any license requirement, which would require non-electronic distribution of documentation.

5. Reporting, Storing and Sharing Review Results

Storage and sharing can be considered parts of reporting, since package reuse requires ability to reuse results of earlier compliance reports. Review results should be stored also for quality control purposes.

Thus, reports on review results have a number of requirements and objectives:

- Reports should be easy to use and (thus) enable compliant use of the package, to the extent possible. The language used should be clear and concise, to enable professionals with different educational background to review the reports.;
- Reports should enable risk assessment in cases packages were found possibly compliant / possibly risky; or even risk assessment of packages found compliant, if a certain legal conclusion was used (in case a user wishes not to accept such legal conclusion);
- Reports should enable variations in risk preferences for different use cases; and
- Reports should enable sharing (separation of generic and use-case specific information).

5.1. License Requirement Simplifications

The Validos process uses simplified license summaries to instruct the redistribution of FOSS packages.

The advantage of this method is that the license requirements become easier to understand, more standardised (same requirement in different licenses is expressed in only one way) and faster to apply. The disadvantage is that the requirement might be applied in a wrong way, since the wording has changed from the original license text. The process needs therefore to provide also the information on the licenses applied, so that the user may read the licenses directly.

However, the writers of this article contend that license requirement simplifications result in a better end result for compliance since full license texts can also be misunderstood. Also we further contend that, even if an open source review board is used for each released project or product, not using simplified license information will result in a non-effective and non-standardised working way. In practice, the compliance officers and lawyers will memorise license requirement simplifications or they will reread license documents. It would therefore be more standardised and effective to use license requirement simplifications reduced to writing in an open source review board too.

5.2. General Use Instructions and Package Specific Use Instructions

The Validos-process has introduced a general use instruction,[21] with the objective to help

21 Link to http://www.validos.org/en/about-validos/37-validoksen-toimintatavat/66-general-use-instruction-for-open-

instructing redistribution of individual packages. The general use instruction covers most frequent requirements in FOSS-licenses. The general use instruction can be applied to all packages: it reduces the length of package specific instructions and standardises the redistribution methodology. Package specific instructions complement the general use instructions with requirements that are not covered by the general use instructions.

The general use instruction of Validos, includes the following:

1. Keep all copyright notices, license references, license texts, notice-texts and warranty disclaimers intact and redistribute these together with the package when you redistribute the software package.
2. Do not use the name or any mark of (i) the software, (ii) the project, (iii) any author or (iv) any copyright holder in any marketing, promotional or similar material or for such purposes, nor in the name of your product or in any other such way.
3. When you modify an open source package and redistribute it as modified, you should always mark your own modifications clearly added with the date of your change. This is typically done by markings at the beginning of the relevant file.
4. When you distribute the open source package as binary, you should also preferably always distribute the source code distribution of the original open source package together with the binary and state in the binary that the original source is distributed together with the binary.
5. If item 4 is not possible (*e.g.,* due to space restrictions) verify that all separate text files listed in item 1 are contained in the binary distribution in a corresponding directory.

The general use instruction covers all the requirements in a number of frequent licenses (such as MIT, BSD and Apache 1.1 with legal conclusion (4.9), Apache 2.0 except patents) and many of the requirements of other licenses. The general use instruction standardises the compliance process for all FOSS projects and makes the instructions for additional license requirements simpler. Therefore a package specific report on a purely GPL 2 licensed package needs to cover only the requirements that go beyond the general use instructions (*i.e.*, copyleft requirement).

Even if many licenses do not require source code redistribution, the item 4 in the general use instruction has been found as a useful way to standardise processes and to reduce work in compiling license notices to separate documents. See also legal conclusion 4.9 on using source code as a documentation.

5.3. Risk Preferences and Assessments

Compliance review will mostly find packages as compliant or possibly compliant. When a package is possibly compliant, a risk assessment is required. Typically it is a question on legal analysis: are these licenses compliant, when combined in this way? Or, is this file licensed with license version 2 or version 3, when the license reference is ambiguous and indications to both license versions exist? Do we need a patent license for a certain cipher even if we remove file x?

The above discussed questions can be solved by policy decisions or further review, such as by

source-packages (retrieved on 2010-04-29)

contacting the open source project or research on cipher patents. These actions might still result in not entirely clear answers. This is when risk assessments are required.

Different use cases might have different preferences for risks, costs and time. The preferences may vary depending on the company or may vary depending on the unit within the company or even within different projects within the same unit. The compliance review reports should separate between information and risk assessment so that risk assessments can be done on a use-case specific level. Validos process does this by not doing the risk assessment, just pointing to the risk and explaining it. However, the legal conclusions we have discussed in section 4, can also be seen as risk decisions, although they are very generic. The reports could also include information on which legal conclusions were applied: this would enable policy decisions not to accept certain legal conclusions.

Information that allows risk assessments is not necessarily simple and straightforward. Therefore it might not be suitable for a simple and straightforward reporting of use instructions for FOSS packages. We have addressed this concern by providing only high-level information on a higher level with pointers to more detailed information. The Validos process provides a one-line report on each package using colour coding for different typical use cases, and then, at certain colour codings, a risk pointer in the package specific use instructions. The pointer includes general information on an eventual risk and points to the full report describing the estimated risk in full. The full report not only allows risk assessment, but also quality control.

5.4. Enabling of Sharing

Package compliance review results in information that is generic and may be used by many companies and may also result in information that is specific to a use case, and as such may not be used as well by others. Since the generic review results are useful to many, its creation can be done in a collaborative fashion.

In order for sharing to become possible, two things must happen: 1) the collaborative production of compliance review information must be more effective than production of the same information by each company separately and 2) the information to be shared must not be confidential. The requirement on effect includes that the information must be readily usable within the processes of the user companies and their supply-chains (upstream and downstream). This in turn means that addition of use-case specific information should be possible without sharing that information to others.

The first requirement is fulfilled by the basic fact that there are many user companies of the same open source packages. (*E.g.,* if the Linux kernel is used as a basis of redistributed products and projects by thousands of companies, then it is not effective for each of the companies to do the compliance review separately, if a working joint way of doing the review exists. The same applies each time a new version of the kernel is issued. Even if a joint compliance effort would need to be much more robust, and therefore perhaps multiple times more costly to produce, still the cost per company would be much lower than individual production of the compliance information by each of the companies).

The key to enabling sharing of FOSS compliance review information is to limit the information to generic information that can be obtained from the open source packages. Another important elements is that the use-case specific information must be easy to add to the generic information.

6. Suggesting Corrective Actions for Found Incompliant Packages

It is not uncommon for FOSS packages to contain code that causes them to pose potential or clear risks when redistributing them. However, the fundamental idea of free and open source software is that code can be modified, and naturally modifications may be used also to fix legal "bugs". In this section we present some options how businesses can deal with packages that are not fully compliant. This is an element that is included in Validos reports, since this is useful information for sharing.

6.1. Removing Problematic Files

Removing problematic files, folders or components from the FOSS package may sometimes be the most efficient method of removing specific legal risks from FOSS packages, caused by, *e.g.*, code which is licensed under incompatible licenses. However, practicality of removing parts needs to be resolved by technical personnel, as incompliant code may be essential to needed functionality or removing code might cause other undesired results such as need for extensive testing. The legal conclusion we have presented above (4.6) discusses legal questions around this.

6.2. Replacing problematic files

Replacing problematic files, folders or components of the FOSS package is closely related to removal of files. Occasionally it can be possible to replace incompliant parts with either compliant versions of needed code or developing such code in-house. Again, practicality of the approach must be evaluated *in casu* since it is dependent on availability of alternative replacements and or amount and costs of developing new code in-house.

6.3. Obtaining Another License

If removal or replacing is not possible for some reason, one alternative which may sometimes resolve incompliance is obtaining an alternative license (FOSS or otherwise) for a code which may not be otherwise redistributable. This option may typically be practical in situations where a FOSS package contains proprietary type software or there is a concern regarding linking copyleft code with other software.

6.4. Accepting related risks

Quite often the legal situation of some FOSS package is subject to true uncertainty caused by ambiguous license terms and lack of relevant case law. These are cases where different but well-founded legal interpretations can be presented but still certainty cannot be reached. Typical issue of this kind is combining and distributing code licensed under the GNU General Public License,

version 2 with software that is licensed under different terms. During the years countless number of bytes has been twisted over the issue, but as to date no definite conclusion has been reached.[22] If the situation is subject to this kind of uncertainty, companies can – and very often will – decide on an internal policy and therefore accept related risks.

6.5. Contacting Open Source Projects

Many times simple contacts to the project can solve risk questions. In working with Validos, we have found most projects responsive and delighted of the contribution regarding licensing questions. Sometimes contacting the authors can be a simple way to solve an uncertainty. In relation to our work at Validos, we have not discussed with projects on their willingness to change clear licensing into another type of licensing either for a single case or more generally, but in some cases that could also be an option to consider.

6.6. Refrain from Redistribution

Sometimes none of the above options, or no other measures, are possible or desired. If non-compliance cannot be solved, then the only available option is to refrain from redistributing certain piece of software.

7. Conclusions

Traditional legal analysis, when applied to copyright law in multiple jurisdictions, FOSS environment and package review would find many uncertainties and arguments pro and contra. In this article we have strived to demonstrate another approach: the approach of creating (legal) community consensus around a given methodology or around a set of legal conclusions and thereby controlling risk and enabling and easing the use of FOSS in a business environment. However, such community consensus is not created by one article, but we hope and envisage that this article could help many in creation of their policies, encourage others to criticize and comment the conclusions presented herein and thereby take a step forward in creation of a consensus by the legal community interested in free and open source software.

About the authors

Martin von Willebrand

22 The GPL linking issue has been analysed in a number of publications, including Determann, Lothar (2006): 'Dangerous Liasons--Software Combinations as Derivative Works? Distribution, Installation, and Execution of Linked Programs Under Copyright Law, Commercial Licenses, and the GPL' Berkeley Technology Law Journal, Volume 21, issue 4. http://www.btlj.org/data/articles/21_04_03.pdf Accessed on 4 May 2010. and Välimäki, Mikko (2005): 'GNU General Public License and the Distribution of Derivative Works'. The Journal of Information, Law and Technology (JILT) 2005(1) http://www2.warwick.ac.uk/fac/soc/law2/elj/jilt/2005_1/välimäki/. Accessed on 4 May 2010. It is to be noted that a working group of the European Legal Network within Free Software Foundation Europe is in process of finishing a detailed document which examines legal and technical situations of combining GPL v.2 software with other code.

Martin von Willebrand is a partner at HH Partners, Attorneys-at-law, Ltd based in Helsinki, Finland. He has introduced the idea of collaborative open source compliance and is currently the chairman of Validos ry, a Finnish association founded for the purpose of easing the use of open source software by businesses and other entities. He is a technology lawyer who has also significant expertise on copyright matters, open source, and related litigation. His technology work is recommended by practically all publications rating Finnish lawyers, such as BestLawyers, Chambers Europe and Legal500.

Mikko-Pekka Partanen

Mikko-Pekka Partanen is an associate lawyer at HH Partners, Attorneys-at-law, Ltd based in Helsinki, Finland. His main specialisation is copyright and open source and he has extensive experience in legal compliance work relating to software development.