§Neolex

Open source software in your business – controlling the risks and reaping the rewards

Introduction

The purpose of this memorandum is to provide an overview of the key compliance and risk management issues for companies intending to utilize open source solutions in their businesses. It is a primer, in that it provides general background knowledge on open source and open source licensing in general, but it also provides high guidelines and instructions for companies wishing to set up their own compliance functions either internally or via an external party.

The memorandum is structured as follows. This introduction section is followed by section B, a primer on open source software, section C, a primer on open source licensing, and section D, license compliance. Finally, the conclusion offers some key takeaways for companies wishing to take their compliance function to the next level.

Primer on open source software

This primer consists of an overview of what open source software is, and how it differs (if at all) from other third party software.

What is open source

An obvious starting point for any discussion on open source is of course the definition of the concept itself. When we talk about open source software, what is that we really mean?

There are two different definitions that are commonly encountered, Open Source Software and Free Software. Sometimes they are used interchangeably, but in reality there are some differences between the two.

The term Open Source refers to software licensed under a license approved by the Open Source Initiative ("**OSI**"). All licenses approved by the OSI conform to the ten criteria set out in the Open Source Definition[1] and software licensed under them are considered open source in the formal sense. The vast majority of contractual terms that refer to open source, define open source as software licensed under an OSI approved license.

The term Free Software is advocated by the Free Software Foundation. It is in many ways similar to Open Source, but its definition adopts a more user-centric perspective in its description of what qualifies as Free Software.[2] If a license complies with the Four Freedoms set out in the Free Software Definition, the program licensed under it is considered to be Free Software.[3]

[1] http://opensource.org/docs/osd
[2] http://www.gnu.org/philosophy/free-sw.html
[3] The freedom to run the program, for any purpose (**freedom 0**).
The freedom to study how the program works, and change it to make it do what you wish (**freedom 1**).

From a business perspective the fine differences between the two do not always have a great practical significance but the existence of the two (especially the OSI definition) is important, given that software that does not comply with the OSI definition usually falls outside contractual stipulations on open source.

Once you know you are dealing with open source software, you know that, regardless of the license, it has certain key characteristics. These are the following:

- The software can be used for any purpose.
- The software can be copied freely and distributed free of charge.
- Derivative works (Software based on existing open source software) are allowed and can be distributed free of charge.
- You have access to the source code.
- You can freely combine open source software with other software assuming the other software's license does not prohibit such linking.

How and why open source is different

From a business perspective, it is important to understand that as such, open source is not different from any other third party software your company is using. It is still third party code, which can be either better or worse in quality than your in-house code but you need to do your own due diligence to be sure it suits your needs.

There are, however, certain characteristics that add a special flavor to open source, and which need to be considered in your company's processes and strategies concerning open source. Some of these key characteristics include:

- The code for the software is available in source code form in addition to the closed binary form. This enables investigation and further development of the software to suit your own needs.
- The code is often developed by open communities with a working model very different from the typical enterprise model. This means features can be developed quickly, but there are less opportunities to control the direction development is going. The rigour of the development model and governance obviously differs from one open source project to another.
- The licenses are focused on protecting the users' rights instead of limiting their rights to use the software. Thus where a standard proprietary license limits your usage to what you paid for, an open source license limits the ability of the licensor to restrict your rights.
- Open source software is generally available to everyone from the Internet, without the need to engage in a time and resource consuming procurement process (unless you are dealing with commercial open source vendors).
- Open source software is virtually always provided "as is" without warranties or IPR indemnities instead of (often limited) warranty/liability granted by proprietary licenses.

The freedom to redistribute copies so you can help your neighbor (**freedom 2**).
The freedom to distribute copies of your modified versions to others (**freedom 3**).

§Neolex

A primer on open source licensing

This licensing primer consists of a summary of key terms and concepts relevant to open source licensing, an overview of open source licenses and license types, and a refresher on why compliance is important for businesses.

Open Source Licensing terminology

When trying to understand open source licensing, it is important to first have an understanding of some key legal terminology and concepts specific to open source licenses.

- **Copyright** protects your code as its written. Patents protect the inventions underlying the code.
- A **Licensor** (author) gives the **licensee** (user) rights to use the software via a **license** (conditions).
- **Copyleft** is not a legal concept, but a term used in some licenses meaning that redistributions must follow the original license terms.
- **Source code disclosure** is the obligation to make the source code of derivative works available required by some licenses (but not all).
- In some licenses, **linking** is a central term; in simple terms it can be divided into two types, **static linking** (direct incorporation of source code pre-compilation) and **dynamic linking** (run-time use through program calls).
- **Outbound licensing**; the main license covering the whole OSS component/project, e.g. Linux kernel is licensed under GPL V2.
- **Inbound licensing**; the licenses of each part of the OSS component, e.g. sub-component or individual contribution.
- **Dual licensing** means for example that there is one license for non-commercial use, and another for commercial use.
- A **derivative work** is a copyright concept (derived from US copyright law) that is at the core of many copyleft licenses, in particular the GPL and the LGPL In the software world it normally refers to a program developed from or on top of a pre-existing program
    - E.g., an extension, a modification, functionality on top of a library

Open source licenses in general

There is no defined format for open source licenses, and their format and clarity varies substantially. Currently there are 66 OSI-approved licenses.[4] Each of them meets the Open Source Definition, but it would be a mistake to assume they are similar.

The reason for the significant number of licenses is that, historically, communities and companies tended to create their own licenses as they felt that the existing licenses were not suitable for some reason. Due to the concerns about the license proliferation this prompted, many have now scaled back on their license initiatives and seek to use existing licenses instead.

[4] http://www.opensource.org/licenses/alphabetical

According to data gathered by Black Duck Software,[5] the most popular licenses (accounting for close to 80% of all open source license use) at the moment are the following:

1.	GNU General Public License (GPL) 2.0	48.66%
2.	GNU Lesser General Public License (LGPL) 2.1	9.33%
3.	Artistic License (Perl)	9.13%
4.	BSD License 2.0	6.24%
5.	GNU General Public License (GPL) 3.0	5.56%

All of the main open source licenses come with a warranty/IPR liability disclaimer, i.e., the software is delivered "as is".

It is important to note that not all licenses purporting to be open source licences actually are as they do not comply with the Open Source Definition due to one reason or another. Some examples (well-known and not) are the following:

- **The Microsoft Limited Public License.** It restricts the development of derivative works to those that can run on Windows.
- **Sun Community Source License.** It prevents the publication of modified versions of the original code.
- **The Open Public License.** The license requires the user to send all modifications made to the code back to the original developer.

Types of open source licenses

Although there are, as noted, over 60 different open source licenses all with different types of provisions, it is possible to categorise at least the most commonly used licenses to a limited number of categories based on their characteristics, thus making it easier to identify the obligations imposed by these licenses. One usable categorization based on reciprocity is as follows:

Permissive licenses:

Permissive licenses allow redistributions under a different license. Most well known examples are the MIT license, Berkeley Software Distribution (BSD) license, and the Apache 1.1. license.

Weak copyleft licenses:

Weak copyleft licenses allow redistributions under a different license, as long as the original code (and its modifications) is relicensed under the same license as the original. The most well known example of this category is the Mozilla Public License (MPL).

[5] http://www.blackducksoftware.com/oss/licenses#top20

Strong copyleft licenses:

In case of strong copyleft licenses, redistributions of all original code and any additions, modifications or linkages need to be under the same license. The paradigm example of this kind of a license is the General Public License (GPL), Lesser General Public License (LGPL).

Why are open source licenses important?

From a legal perspective, open source licenses do not differ from any other license. Like any license, they define the conditions for using the authors' intellectual property. The scope and nature of these obligations can, however, differ from those in standard proprietary licenses. Typical obligations that open source licenses include are obligations to include the authors' copyright notices in the products, obligation to distribute derivative works under the same license, and sometimes to provide a copy of the source code with your distribution.

Although the license obligations do not seem as strict as in proprietary licenses, compliance with them is nevertheless of vital importance. If you do not comply, you are using the software illegally and your company's right to use it is likely to be revoked. It could also lead to product recalls which can be very costly.

Compliance is very much a hot topic at the moment, with several new litigations launched in the past two years. Examples of these include:

- **December 2009** – The Software Freedom Law Center sues 14 electronics companies for alleged GPL violations. Defendants include Samsung, JVC and Best Buy. (**Source: Software Freedom Law Center**)
- **July 2009** – "Microsoft violated the General Public License v2 (GPLv2) when it distributed its Hyper-V Linux Integration Components (LinuxIC) without providing source code, says the Software Freedom Law Center (SFLC). The violation was rectified when Microsoft ***contributed more than 20,000 lines of source code to the Linux community last week.*** " (**Source: SDTimes**)
- **May 2009** - Cisco settles a GPL violation case with the Free Software Foundation to avoid an injunction prohibiting the sale of its routers. As part of the settlement, Cisco made an undisclosed monetary payment to the FSF, appointed a Free Software Director, and further agreed to take certain steps to notify previous recipients of Linksys products containing FSF programs of their rights under the GPL and other applicable licenses, to publish a licensing notice on the Linksys website, and to provide additional notices in a separate publication. (**Source: Free Software Foundation**)
    - Cisco had paid 500 million USD for Linksys, the company producing the routers, and now was forced to disclose the source code of its thought to be closed source firmware
- **May 2008** – Skype was sued by the copyright owner of GPL-licensed open source code included in a Skype phone. Skype lost, and was forced to disclose the source code of the operating system its phones use (**Source: GPL-violations.org**)

So, it is extremely important to know what the licenses mean to your company and your customers, and follow the license terms!

License compliance in typical open source use cases

Given the importance of open source licences and compliance (in other words, the **why**), it is only logical to move to the next step, the **how**. In order to avoid explaining compliance in abstract terms, it is best to discuss it in context of various common use cases. The most common ones are

1. internal use
2. distribution in a modified or unmodified form
3. offering as a service

For each scenario, the compliance process consists of the following steps:

i) Determining whether the license obligations are triggered at all in the scenario in question; if yes, then
ii) Identifying the outbound and inbound licenses; and
iii) Determining whether there are any conflicts between the outbound and inbound licences; if no, then
iv) Deciding on what steps need to be taken to comply with the applicable license(s)

We cover compliance in each scenario in turn.

Using open source software internally

A typical scenario is that of a software developer or an IT admin working for the company needing a third party tool to help in his work. Because most open source tools are freely available on the Internet, they are often the tool of choice especially for ad hoc development, debugging and maintenance tasks.

Example: IT System admin X downloads Wireshark to help him solve a network problem plaguing the company's internal network. The software is licensed under the GPL version 2.

How to comply: Assuming the tool is only used for internal use and not distributed further, compliance is rather simple. Because GPL's obligations are only triggered on _distribution_, the company can use the software freely without concerns that any license obligations would be imposed on them. From a compliance perspective it is still important to keep track of internal use as well, as internally used tools in some cases find their way into finished customer products.

Distributing third party open source in a modified or unmodified form

Compliance becomes an actual issue when your company is distributing open source software to its customers or third parties. In the vast majority of cases license obligations binding on the licensee are only triggered at this stage. If the company distributes the open source solution in an unmodified form, compliance is usually relatively simple (with the only issue being the scope of the copyleft obligation), but if the open source software is modified, the licenses' provisions on derivative works and modifications are usually triggered and need to be considered as well.

Example: The company uses a modified version of nHibernate in one of its embedded solutions to allow mapping of .NET classes to database tables. The hardware embedding the solution is sold to customers.

How to comply: In this relatively typical distribution scenario compliance involves several steps.

First, the company needs to identify nHibernate's outbound license. This, according to the project's website, is the Lesser General Public License 2.1.

Second, the company needs to ensure that nHibernate's inbound-licensing does not conflict with the outbound-license. In practice this means that the company needs to ensure that no files or sub-components included in nHibernate are licensed under a license that would either i) conflict with the outbound LGPL license, or ii) in any other way add additional obligations to those imposed by the LGPL.

This is the most difficult, and often the most neglected part, of license compliance. However it is also the most crucial part, especially for hardware companies that need to provide IPR warranties for their product as a whole, because any such conflict, or an additional obligation imposed by an inbound license that is not complied with, results in a copyright infringement and a breach of the warranty. Based on experience, we can say that roughly 25% of open source components have moderate to serious problems with conflicts between outbound and inbound licenses. More often than not this is a case of what one might call *innocent infringement*, in that none of the copyright holders involved would ever consider suing for the infringement in question, but given the increased number of cases brought in the courts nowadays, there is no reason why the company should take such a risk.

Third, once the outbound license is identified and it is clear there are no conflicts or other issues with the project's in-bound licensing, the company needs to determine what steps it needs to take in order to comply with the applicable license(s). In this case the license is the LGPL, meaning that the following issues need to be considered

- The LGPL as a general rule requires the whole software work to be distributed under the LGPL and the source code to be disclosed.
- As an exception, simplifying slightly, the LGPL allows integrating an LGPL component into a larger software solution and licensing the whole under a different license provided that the non-LGPL parts of the solution are not part of the LGPL component but merely use it as a library. Even in this case, the terms of the chosen license need to allow modifications for the recipient's own use and debugging of such modifications.

In our example, the company thus needs to determine how the LGPL license affects the licensing of its own solution. The ideal outcome for the company in most cases would be that it could distribute the whole software solution under its own proprietary license and that the LGPL would be limited to the nHibernate component. This can be accomplished if the solution is developed for example in the following manner:

- The solution uses the nHibernate library via system calls made at run-time. In other words the two are compiled separately but can interoperate with each other.
- The modifications made to the nHibernate component, if of commercial value in themselves, are not made directly to the library but via either an intermediary layer added to the non-LGPL part of the solution or by engineering the solution in a way that it for example interprets the mapping values from nHibernate differently without actually modifying nHibernate itself.

Assuming the first point is complied with, the company can be virtually certain that it can license the solution under any license its wishes (subject to the LGPL's provisions on e.g., reverse-engineering and debugging). The original LGPL component still needs to be licensed under the LGPL even in this case. If the second point can be complied with, the company's own modifications to the LGPL component can also be licensed under any license, and not necessarily under the LGPL.

If for technical or resource reasons the second point cannot be complied with, the modifications made by the company needs to be licensed under the LGPL as part of the LGPL library. This means not only compliance with the LGPL's provisions regarding the scope of the grant of rights, but also disclosure of the source code of the modifications as well. An important point to note is that in case of the LGPL (and virtually all other common open source licenses requiring source code disclosure), the source code needs to be disclosed only to the recipients of the solution containing the open source licensed code in question. Thus if a company sells solutions directly to customers without offering them publicly, it only needs to disclose the source code to those customers it sells the products to. If on the other hand it makes the solution available as a download on the internet for the public at large, the code needs to be made available to everyone as well.

Offering open source software as a service

In today's world, service-based offerings are becoming more common every day both for pure software companies (e.g., Google) and those offering complete hardware and software solutions (e.g. EMC being a prime example). Typical to these services is that the customer accesses the service via a client application (e.g., a browser or a mobile application) but never receives a copy of the actual software that provides the service.

Example: Company X develops a cloud storage solution consisting of i) hardware for storage, i) software in the cloud that offers a service for uploading and download of files, and iii) a client application that access the service via the Internet. The software residing in the cloud contains various GPL licensed open source components as well as significant portions of proprietary code developed in-house.

How to comply: The key point to consider in this case is how the service offering is designed. Assuming that it is a true cloud-based offering, the only piece of software distributed to the customer is a client application for accessing the cloud services (and if the client is the browser, not even it is distributed), and thus the customer does not receive any open source licensed code. There is thus no distribution of open source code, and the GPL's obligations are not triggered. The company is free to offer the service without having to worry about the GPL's copyleft effect or its source code disclosure obligations.

It is worthy of note that there are open source licenses that apply to cloud based services as well. The Affero GPL is a prime example of this, and although its usage has generally been negligible it has increased in the past two years. It has become popular with companies developing cloud-based open source solutions for others. For example the Funambol mobile sync cloud service offered to telecom operates is available for free under the Affero GPL which enables prospective customers to try it. However, unless they want to disclose the source code of their own offerings tied to Funambol they will purchase a commercial license from the company before they offer the service to their customers.

Setting up your compliance process

The previous examples demonstrated that compliance with open source licenses is in some cases relatively simple, but in some cases it can require a great deal of work. To ensure that the company is well equipped to handle both the simple and the complex cases, it is recommended that the company has in place the necessary support functions and expertise to enable compliance issues to be handled quickly and efficiently.

Based on our experience of the typical compliance scenarios, a company's open source compliance function would need to include at least the following resources:

- A person with a good understanding of open source development models and communities and a basic understanding of open source licensing.
- A lawyer with a good understanding of open source licensing
- A technical person that can advise on the technical issues related to linking that are important in assessing how certain open source licenses affect the software's own license

Ideally the compliance function would also need a tool for easily identifying open source project's inbound licenses. Currently there are both commercial and free tools available for such a purpose, and some larger players in the sector have developed their own internal tools.

Conclusion

Open source software offers companies unique opportunities for new business models, cost savings, and distributed development that helps both reduce the time to market and to improve the quality of the work produced. At the same time, it poses certain unique challenges as regards compliance because technical considerations play a much larger role than they would in a normal case. To truly understand the opportunities and limits set by open source licenses, the company's compliance team needs to understand both the technical and legal aspects that relate to the open source elements of the developed solution, as well as how the open source communities maintaining the chosen open source components operate. This is a daunting task at first, as it requires different kind of staffing and knowledge than normal third party software evaluations and procurement, but once in place, the team can add great value to the company's business enabling it to reap the rewards lying at the end of the open road.

***

For more information, contact the author at oniiranen@neolex.fi or visit http://www.neolex.fi.