Understanding Free and Open Source Licenses, Version 2.1

OPTAROS WHITE PAPER

Realize the Benefits of Open Source

Table of Contents

Executive Summary

Open source software is more and more an alternative for large enterprises. Most companies already use open source software, starting with Linux and web servers, but also databases or complete infrastructure stacks to run their applications. Cost benefits, higher quality, and increased independence from vendors are driving factors for this trend. Often, however, CIOs and IT decision makers do not take open source software into consideration due to perceived risks involved. Concerns about legal issues and uneasiness about the involved licenses are mentioned frequently.

Understanding free and open source (FOSS) licenses is not actually difficult. Looking at the history and some of the more important developments allows one to assess the impact of using FOSS in the enterprise. Risks for an enterprise that is only using open source software are minimal or actually comparable to using commercial software.

Introduction

Programmers have been sharing computer programs and source code since we had computers. In the early days, it was often done through professional and user organizations such as DECUS, SHARE, and USENIX. The licenses through which such sharing happened were as varied as the end user license agreements (EULA) of proprietary software vendors today, and all such licenses rely on strong intellectual property laws and copyright laws.

This sharing of software has reached new heights over the past couple of decades, enabled through the ease of sharing across the Internet. The concepts of “free” and “open source” software have become mainstream and licensing is the avenue through which the rules of this particular form of sharing software are established.

Understanding free and open source (FOSS) licenses is not actually that difficult. A little history and a few pointers can clarify some of the confusion to enable organizations to make the best use of FOSS in their own business contexts. With this understanding we will take a look at enterprise considerations around FOSS use, whether as an enterprise that wants to use FOSS in parts of its business infrastructure or a vendor looking for a competitive edge and a new value proposition for its customers. We will close by looking at “dual” licensing, and collected works, and how to think about contributions.

The Licenses

There are primarily three license types or families that have arisen historically:

- Academic licenses (MIT Athena, Berkeley, and Apache)
- Free software licenses (General Public License and the LGPL)
- Mozilla-style licenses (Mozilla, and the IBM licenses)

We will note a few other interesting licenses along the way, but even these derive from the basic models described in these three groups.

The Academic Licenses (Berkeley, MIT, Apache)

During the mid-1980s period, the Computer Science Research Group at the University of California, Berkeley was doing a large amount of research work on early UNIX systems, and acted as a hub for the collaborative research community. The regents of the university developed a simple license for their work to encourage new research and adoption of the software. The Berkeley license essentially:

- Enables the software user to do anything with the software, including extending and selling it.
- Does not require any derived software be licensed under the same license or that the changes be published. This enables “closed” or proprietary products to include such licensed software safely.
- Requires that attribution be given for the work, and copyrights maintained.
- Disclaims any warranties (express or otherwise) just as a proprietary EULA does.

This license style has been referred to as Berkeley-style licensing. This was also the basic model for the MIT Project Athena license (used for the X11 windowing technology, including all the contributions from Hewlett-Packard and Digital Equipment Corporation).

As we will see, Berkeley-style licensing supports a reciprocity belief counter to that espoused by the Free Software Foundation (FSF) – that the software would definitely be freely distributable, but the reciprocity requirement should be encouraged in the community and not commanded in the license.

The Berkeley-style license was also the model used in the early Apache community in 1995. The original Apache web server was created out of work developed at the National Center for Supercomputing Applications, University of Illinois, and the license reflected the research base and collaborative development in that community.

In 2004 the Apache 2.0 license was released. It is a complete rewrite to account for current concerns of software contributions and patents, and is a richer and more complex license in its legal structure, but it remains true to the principles of its history.

Free Software Licenses (GPL, LGPL)

In 1985, Richard Stallman created the Free Software Foundation and his definition of software freedom, where a program's source code was always available and a user could always fix and extend the software without restriction. The General Public License (GPL) defined this particular sharing foundation.

- If the user distributes the changed software, they can only do so by sharing their changes the same way through the same license. This is the reciprocity requirement of the free software definition. This is a primary difference from the academic class of licenses that permit derivatives to be re-licensed under other (possibly closed, proprietary) terms.
- If one uses any of the GPL-licensed source code in his or her program, and distributes the program, the entire newly-derived program including their own source code becomes subject to the GPL.
- The GPL disclaims any warranties (express or otherwise) just as proprietary EULAs do.

A few important points to note are:

- The reciprocity requirements are triggered on distribution of the software, not on using it.
- There is nothing that has forced exposure of the source code to one's application. The license contains its own redress. One can always withdraw the software distribution. (If part of a commercial software organization, this might still prove onerous, and so one does need to pay attention when working with GPL software that will be distributed.)

The Lesser GPL (LGPL) was developed later to account for software libraries. Many that would share their software subroutine libraries under the GPL didn't necessarily want to force the recipient to have to share anything other than their changes to the library. The way the GPL was written would unfortunately force the entire software (libraries and the program using the libraries) to come under the GPL. The LGPL enabled a library to be licensed and did not require the entire application to be licensed under the same license (and so enabling it to remain closed), while still requiring changes to the library itself to be published under the LGPL if distributed.

Many of the most important FOSS programs of the past twenty years are licensed under the GPL, including the Linux operating system, the GCC compiler suite, the MySQL database engine, and JBOSS application server. Many vendors (including software vendors) use and develop software licensed under the GPL.

All through much of the rest of the 1980s and 1990s, everyone followed one of these two models with simple variations around such clauses as jurisdiction.

The Mozilla License

Before we cover the Mozilla license, a small detour is in order. When the Perl language hit the scene, Larry Wall created the Artistic License. The Artistic license was intended to maintain the open aspect of the Artistic licensed code, while enabling innovation around the core project to be licensed as appropriate. It tried to find a balance between the hard line sharing required by the GPL and the complete freedom of the academic licenses. It is a popular license, though some consider it legally ambiguous in places.

In the late 1990s, Netscape published the source code to their browser and began to build a community of developers around it. This project was called the Mozilla project, and the license created was the Mozilla Public License (MPL). This is one of the first licenses created by a corporation, and that heritage shows through in its legal structure and depth compared to FOSS licenses prior to that point. It had similar goals to the Artistic License. Essentially, the MPL:

- Requires derivatives of the MPL work that are the original work plus contributions to be licensed under the MPL, thus creating the reciprocity of the GPL for the core project.

- Enables MPL licensed works to be combined with other software and re-licensed into a “Larger Work.” This enables the development of possibly closed proprietary software similar to the academic licenses.
- Discusses patent rights relevant to the licensed work.
- Disclaims any warranties (express or otherwise) just as proprietary a EULA does.

There has been a proliferation of open source software licenses based on the Mozilla license, because other companies wishing to develop collaborative software communities as a business tool invariably want to change the jurisdiction clause and define language around what patent concerns they may or may not have. The language of the Mozilla Public License is very centric to the Mozilla project.

One can see a certain lineage to the Mozilla license in the development of IBM licenses, from the original IBM Public License through the Common Public License to the newest Eclipse Public License that are used around the Eclipse project.

Enterprise Considerations

For the most part, enterprises using free and open source software should have few concerns about licensing for the following key reasons:

- All licenses essentially allow the software to be run (binary form) without restriction.
- Under all licenses, the source code can be modified without restriction if the resulting software is being used internally.
- The GPL and Mozilla families of licenses place requirements for re-licensing and publication on the user only if they distribute the software. This would only have implications on an enterprise if they plan to distribute the software to their customers.

If buying packaged free or open source software or a system that contains such software (e.g. Red Hat Advanced Server, or HP/UX), then the Red Hat or HP EULA is the primary concern, and all other third party license concerns are left to the vendor.

When using open source packages, such as the MySQL database engine, JBOSS application server, or any of the Java frameworks that are FOSS licensed, the license enables free deployment and use and there are no concerns within an enterprise – the enterprise isn't developing software derivatives that they distribute.

Indeed, the ability to copy open source software freely and deploy as much as is needed within an enterprise means the historical (and sometimes litigious) problem of counting users or processors ceases, along with the auditing costs involved. This ability to deploy freely also frees up the architecture of solutions to problems. For example, there was a time when the solution architecture was designed around reducing the number of very expensive licenses one required for application server middleware, and database access. With the ability to deploy as many application servers as is required and distribute the database across systems equally freely because of a lack of per system license fees, the solution can be designed and built to real requirements. The solution can grow more organically to meet the needs of the enterprise at marginal additional costs. The issues then fall back to concerns about support and maintenance.

Dual Licensing

A lot is made about the idea of "dual" licensing and open source business models. This is another area that does not specifically pertain to open source software per se. It is an attribute of intellectual property law that the property owner can license their property to as many people, as many times, and in as many ways as they choose.

MySQL is an example of this. MySQL AB (the company) licenses MySQL (the software) using the GPL. If a company would like to embed the MySQL engine in their own product or appliance without licensing the rest of their software under the GPL, then they can obtain an OEM license from MySQL AB, allowing them to keep their software closed. MySQL AB can do this because they "own" the MySQL software, ensuring they have the rights assigned to them for substantive contributions from the community.

This is not specific to open source software. When purchasing Microsoft Windows XP from the local office supply store, the software is licensed using the End User License Agreement (EULA). When a new PC arrives in a place of business, it may well have been licensed under an enterprise or Select agreement.

Collected Works

Another area of confusion is around the concept of collected works and licensing. Collected works also have copyrights that can be licensed. This idea came from written collections such as periodicals and encyclopedias. Software "collections" also fall under this idea, which makes sense when thinking about Linux distributions.

Red Hat's Fedora distribution has its own license (that is very different from the Red Hat Advanced Server license). This license protects the Fedora trademark by discussing what can and can't be done with the "collection". This doesn't replace the other licenses (such as the GPL on the Linux kernel itself). It covers the collection. Likewise, the OpenSuSE distribution also has a license for the collected work, also protecting the trademark, and different from the Fedora license.

Again, this is not unique to open source software. Microsoft Windows XP contains software from third parties, licensed to Microsoft. As the user of the software, one cares about the collection, and the Microsoft EULA. Likewise, Hewlett-Packard HP/UX ships with a number of tools from the open source world. As a customer, one doesn't care about all the individual licenses per se, but the overarching HP license for the product.

Software Contributions

Software licenses are outbound documents from the project to the user. It helps to understand the inbound agreements at a high level that are sometimes mentioned in the context of contributions.

There are many reasons for contributing work to open source projects, such as bug fixes or enhancements. Many of the larger projects with commercial implications (either directly like MySQL or indirectly such as Apache Software Foundation) have put in place an assignment process whereby one would assign the property rights for a contribution to the project holder, or license software to the project. The assignment process typically involves some form of paper trail that deals with:

- Assignment of (or license to) the contribution itself.

- Warranties that the contributor has the rights to assign or license the contribution. This sometimes requires an individual contributor to obtain signatures from their employer (as many employment agreements claim the ownership of employees work).
- Warranties that to the best of the contributor's knowledge, they have not infringed on anyone's rights (e.g. patent rights) with the contribution.

As a single point of copyright control, the project could license the software in multiple ways such as the MySQL Inc. or Sleepy Cat Software Inc. dual licensing schemes. It can also legally defend the work more easily should the need arise. The Free Software Foundation and the Apache Software Foundation have such assignment processes.

Such assignment processes should also support the ability to maintain rights to contributed work in such a way that one can do with it as they see fit. This is typically by request or directly stated in the assignment documents in some form of a grant back of the rights given to the project copyright holder. One should always maintain the rights to his or her work.

Summary

Open source software licensing is not complex, and relies on strong intellectual property laws. The licensing practices have evolved over time, and since the late 1990s now display more of a corporate tone.

While many licensing subjects are presented as "open source" related, such as dual licensing and collected works, these concepts are actually just attributes of software intellectual property law. There is nothing different for open source software. Developers have shared and licensed software long before the terms "free software" or "open source" were coined.

With some basic understanding, an enterprise can better approach open source software for use within the enterprise to gain the benefits in cost savings and development efficiency.

Getting More Information

The following web sites and books are excellent sources of additional information on free and open source software licensing.

Web sites:

- The Open Source Definition (http://www.opensource.org/docs/definition.php)
- The Free Software Foundation definition of free software (http://www.fsf.org/licensing/essays/free-sw.html)
- Open Source Initiative approved licenses referenced in this document can all be found at the following web site: http://www.opensource.org/licenses/

Books:

- Lawrence Rosen, Open Source Licensing, Prentice Hall PTR, Upper Saddle River, NJ, 2004 (ISBN 0-13-148787-6)
- Andrew M. St. Laurent, Open Source and Free Software Licensing, O'Reilly Media Inc., Sebastopol, CA, 2004, (ISBN 0-596-00581-4)

About the Author Stephen Walli

Stephen Walli is Vice President Open Source Development Strategy at Optaros. Mr. Walli was an advocate for open source at Microsoft, where he was focused on “shared source” business strategies and was responsible for technical implementation of open source-related community projects. Stephen was a long time participant and officer at the IEEE and ISO POSIX standards groups, representing both USENIX and EurOpen (E.U.U.G.) and has been a regular speaker and writer on open systems standards since 1991.

About Optaros

Optaros is an international consulting and systems integration firm that provides enterprises with best-fit solutions to IT business challenges, maximizing the benefits of open source software. Whether your organization has had little exposure to open source software or has open source-based solutions currently in place, Optaros has the business and IT problem solving experience to advise on the areas where open source and open standards can be effective and how to proactively manage the use and benefits of open source software.

Optaros offers a third alternative to the “build versus buy” decision with our proven assembly methodology (OptAM). Within our core practice areas, Optaros ensures successful solution delivery by leveraging our pre-selected open source solution sets and customizing to your specific business requirements.

Contact Optaros

- United States: Brian Otis, Vice President of Sales and Business Development
  email: botis@optaros.com
  phone: (617) 227-1855 x110
- Geneva, Switzerland: Kay Flieger
  email: kflieger@optaros.com
  phone: (617) 227-1855 x225
- Zurich, Switzerland: Heinz Rudin
  email: hrudin@optaros.com
  phone: (617) 227-1855 x224

Disclaimer:

Optaros makes no representations or warranties with respect to the contents or use of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.