

OPEN SOURCE SOFTWARE LICENSING

Markus Weissmann <mail@mweissmann.de>

Georg Simon Ohm University of Applied Sciences,
11/09/2005

<i>Introduction</i>	3
<i>Software Licensing</i>	3
<i>Copyright and Ownership</i>	3
<i>Classic Licensing</i>	3
<i>Open Source Licensing</i>	4
<i>BSD License</i>	5
<i>GNU Public License</i>	6
<i>Summary</i>	7

Introduction

Not later than Linux hit the enterprise market with support from big companies like IBM, HP, Intel and Novel, everybody has heard about its development model: Open Source. This approach to license software seems to develop great momentum for quite some projects. Many of these projects already have superseded or are in the process of replacing established proprietary software. It has become very important for decision makers, being it programmers or managers, to know about the advantages and disadvantages of making your own software open source. But you also have to know about the risks and benefits of using software that already is licensed in this way.

Software Licensing

COPYRIGHT AND OWNERSHIP

Most of the world's nations today feature some kind of copyright, that grants the author of e. g. a poem certain rights on his act. Programs also do enjoy this protection, as basically every written text does. This makes the author of a program - or the person employing the author - the copyright holder, basically giving her the exclusive rights to decide what should happen with her creation. As this kind of ownership is accepted in all modern developed nations - especially the economically important G7 - the now following evidences are valid for world wide software development.

CLASSIC LICENSING

The classic software licensing - or more precisely the closed source software licensing - was made popular by Microsoft licensing MS-DOS to IBM in the 1980s. Originally no one thought about software licenses during the mainframe time: The computer hardware was so expensive and proprietary that something like software piracy practically was unheard of. Most software was written directly in machine language or at least a programming language that was only useable on very specific set of computers. Also only very few people had access to computers at all, mostly scientists working for the military and research laboratories.

In the advent of the post-mainframe workstation market and the upcoming personal computer revolution, small companies like (then) Microsoft began selling software. Buying software now meant licensing software, just getting the right to use a certain version of a software. Meanwhile this approach to making money with software has become the established one for quite some share of the software market.

This market strategy gives a lot of power to the software companies and often leaves users at the mercy of the software supplier, especially if the company acquired a monopoly-like position in their market segment. The so called vendor lock-in is another problem that might strike users: The more of your data exists in a proprietary, closed format - e. g.

Microsoft Excel file format - the higher the costs of migrating the data to a new format of another product - be it closed source or not - will be. These costs may well inhibit a migration at all due to economic reasons. Most often this situation gets worse over time, getting you more and more dependent on one particular supplier. So what can you do to escape this cycle? What can serve as security for your customers not to get in?

OPEN SOURCE LICENSING

In contrast to classic software licensing stands the open source software licensing, or abbreviated as OSS licensing. The idea behind OSS is to share the source code of a program, to share the ideas and their implementations and make it accessible to everyone.

The economics behind this software development model seem strange at first sight: How can you make money if you give away your product for free? Won't it get copied and sold by a supplier of proprietary software? Who is paying the bill of the servers, etc. if you do not make money?

There are several ways to make money from open source software:

- Hold training how to use your software. (e. g. Red-Hat makes certification tests and training for their Linux-based operating system)
- Develop cooperatively for a common goal instead of buying software. (e. g. Fujitsu-Siemens and Sun are both involved in the development of the free PostgreSQL database, instead of paying - or have their customers pay - for Oracle licenses)
- IT-Specialists who wrote OSS may well be recruited by companies who want to deploy the software he is the author of.
- Get people on board who enjoy working for you for no money. (e. g. Sun just started this approach with their OpenSolaris project)

So what exactly is open source software? The definition from the Open Source Initiative¹ - a non-profit organization - is the common sense here. It's Open Source Definition² has ten requirements for a license to be an open source license:

1. Free redistribution
2. Source code
3. Derived works
4. Integrity of the author's source code
5. No discrimination against persons or groups

¹ Open Source Initiative - OSI, <http://www.opensource.org/>

² Open Source Definition - OSD http://www.opensource.org/docs/definition_plain.php

6. No discrimination against fields of endeavor
7. Distribution of license
8. License must not be specific to a product
9. License must not restrict other software
10. License must be technology neutral

The OSI also approves open source licenses and has a list of accepted licenses.

Such a license shifts a lot of power originally owned by the author to the user of the software. As the user now owns the source code, she may well have a look at it to create a converter to bring her data to a different file format. This way, the customer does not need to fear a vendor lock-in.

Also if the vendor gets out of business by going bankrupt, abandoning the software, changing his priorities etc. the user is not lost. If he finds a bug, needs a new feature or other extension, he may well pay another software company to modify the program, as he owns the source code.

There are quite some licenses that fulfill the OSI's requirements (it is even considered that there are too many OSS licenses to keep an overview over all of them). We will have a look at the two most known ones and see what advantages and disadvantages they bring along for software products.

B S D L I C E N S E

The Berkeley Software Distribution license³, or commonly known as BSD license is a very liberal one. It was originally used by the University of Berkeley in California for the BSD Unix operating system. The so called revised BSD license only sports three requirements, which basically claim, that the license has to be retained. The original BSD license also demanded to name the University of Berkeley (as can be seen when booting Microsoft Windows NT), a clause that was removed by the University in 1999⁴.

The license further has a denial of any warranty clause, that shall protect the author from debts from damage caused by the software. This clause can be found in many other OSS licenses: As most OSS is given away for free, you cannot expect warranties from the author, who is not directly receiving money from you.

What does this mean for a provider of BSD-licensed source code? If you give away your source under this license, the recipient may basically do everything with it. He may even use it in his own product without giving credit to you in the public. He is just forbidden

³ BSD license, <http://www.opensource.org/licenses/bsd-license.php>

⁴ Adv. clause rescind, <ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change>

to use your name to promote the product and to hold you responsible for caused damage (though it is uncertain if this non-responsibility clause is effective in Germany).

This effect is neat if you want to spread your ideas, e. g. if you are a scientist or want to create a standard. If you set your reference implementation free under the BSD license, every programmer is free to integrate your code with virtually no restrictions. By this, your code may make it into every product - be it commercial or open source - that can make use of your ideas. For example hardware makers may offer drivers for their hardware this way to get it integrated into as many operating systems as possible, therewith selling more of their devices. The internet protocol also came to it's success this way: A reference implementation was available via the FreeBSD operating system which both Microsoft Windows and Linux copied.

We can conclude that the BSD license is nice to copiers and can spread ideas and reference implementations quickly. But what if you want to hinder the copier to get away with your labour for free without giving something back?

GNU PUBLIC LICENSE

Richard Stallman and his GNU project are the parents of the GPL⁵, the GNU Public License. It was created to ensure that open source software stays open source and also that non-GPLed software - regardless it's license - gets open source. The GPL also is the most popular of the open source licenses. Open source software that is licensed under the GPL is often also referred to as free software.

To summarize the contents of the GPL, it basically has the same requirements as the BSD license - including the no-warranty clause - but also demands that software created from GPL-licensed code always has to be provided by source, too. So if e. g. the IP implementation of FreeBSD had been GPL, Microsoft had to give the source code of Windows NT to every customer who bought a copy of it. This so called viral-effect reaches so far, that even the usage of a library that is under the GPL will make your program GPL itself.

The viral effect can be a thread to software considered to be the crown-jewels of a company. If someone can prove that it uses code from a GPL product (or uses a GPL library), the company can be legally obligated to publish the code (in Germany at least). This can mean the end for a company relying heavily on revenues from sold licenses. On the other hand individuals can be forced to share modifications they did to GPL licensed software, like the Linux kernel - something that already happened to several hardware makers in Germany.

A common business model is to make money from this viral effect by so called dual-licensing of software. This means to release the software under the GPL and simultane-

⁵ GPL License, Version 2, <http://www.gnu.org/licenses/gpl.txt>

ously sell classic-licensed copies. This way you may get the free labour from the open source enthusiasts via the GPLed version, while making money from companies who do not want to use the GPL version. Of course this only works as long as you keep control over the development of the software - other people may well start their own version of the software from your code base.

The most popular software that is GPL licensed is the Linux kernel and the GNU Compiler Collection (gcc). They both use the shared-effort business model, where several companies push forward the development together. Of course no one now can sell the derived product - if it's development happens publicly like for the Linux kernel - but it may well be a way to sell hardware, software that works atop of it or services so customers can use the software with less effort. It also can be used to apply pressure on an established software maker or for customers to avoid paying for classically licensed software from third parties and so indirectly lowering the prices of my own products.

Summary

Open source software opens up a whole new market for software makers. Opening your software in this way may well empower younger companies to compete with bigger companies that do rely on closed source software. There may well be many people and companies out there wanting to help you developing your software, you just have to get away from getting paid for software licenses to compete with software makers that you probably cannot drive out of their market niche with another closed-source application.

For scientists the BSD license offers the opportunity to get your ideas spread out to the world and into real applications. To standardization boards it creates the chance of making your standard clear, to get every product out there to comply to your reference implementation.

Companies can take advantage of reducing costs or strengthen their development forces for free. They probably prefer to use the GPL to either dual-license the product or - if they are only interested in the software as a building stone in their product - to use a GPL-like license to keep the costs for it low by developing cooperatively with other companies or persons.

Open source licensing already works for a lot of software and a lot of companies, be it big ones like IBM or just small software consultant bureaus. We already see giants from the old closed-source world tumble and young open-source companies with constantly rising stock prices. It is time to reconsider old licensing models and react to the soaring open source model and the new opportunities it brings along.