Software Licensing

A software license is a legally binding agreement that specifies the terms of use for an application and defines the rights of the software producer and of the end-user. All software must be legally licensed before it may be installed.

Software licensing can be a confusing subject. There are different types of licenses and licensing contracts, and different vendors may use different terms to describe their licenses. Here are some key terms to help you navigate through these murky waters.

Types of licenses

Proprietary license

Most software licenses are "proprietary" licenses, meaning the software publisher grants a license to use one or more copies of software, but that ownership of those copies remains with the software publisher. The user must accept the license before they are permitted to use the software.

GNU General Public License

These are agreements under which much "open source" software is licensed. End users may do things like change the source code, but any refinements of the software must also be made available under a GNU GPL license. Often referred to as "free, copyleft" licenses, the software may or may not be distributed for a fee - "free" refers to the ability of users to change and distribute modifications of the software, not to cost. .

The GNU General Public License (GNU GPL or GPL) is a widely used free software license, which guarantees end users the freedom to run, study, share and modify the software. The license was originally written by Richard Stallman of the Free Software Foundation (FSF) for the GNU Project, and grants the recipients of a computer program the rights of the Free Software Definition. The GPL is a copyleft license, which means that derivative work can only be distributed under the same license terms. This is in distinction to permissive free software licenses, of which the BSD licenses and the MIT License are widely used examples. GPL was the first copyleft license for general use.

End User License Agreement (EULA)

Also called "clickwraps" or "shrinkwraps," EULAS indicate the terms under which the end-user may use the software. Agreements with organizations or companies often take the form of contracts between the organization and the software publisher or vendor, and specify the terms of use for all users from the organization, superseding any EULAs which may come with the software.

Workstation licenses

These are licenses that permit the installation of an application on a single computer. You may not install the software on more than one machine unless you purchase a license for each additional machine. Most workstation license agreements allow you to make a single backup copy of the software as long as that backup copy is used only to restore the software onto the same machine, or a separate machine if the software is removed from the original computer.

Concurrent use license

These are licenses that permit you to install the software onto multiple machines as long as the number of computers using the software at the same time does not exceed the number of licenses which you have purchased. Concurrent use licenses are usually used in conjunction with "license manager" software that prevents the number of licenses from being exceeded. At UNCG, ITS uses KeyServer software that monitors and controls the use of concurrent use licensed software.

Site licenses

A site license permits the use of software on any computer at a specified site. Unlimited site licenses allow the installation of software on any number of computers as long as those computers are located at the specified site. Some site licenses permit the installation on computers owned by a particular entity (such as a university) regardless of the physical location. Some vendors refer to their licenses as site licenses but restrict the number of computers on which the software may be installed. The only way to know for sure is to read the license specifics.

Perpetual licenses

These are licenses without expiration dates, which permit use of the software indefinitely, without requiring a recurring fee for continued use. Most software that individuals buy for use on their home computers are perpetual licenses.

Non-perpetual licenses

These are licenses that "lease" the software for use for a specified period of time, usually annually or sometimes bi-annually. Users are required to remove the software from their computer if they cease paying the license fee.

License with Maintenance

Some license agreements allow the user to purchase "maintenance" or "software assurance" along with the original license fee, which entitles the user to receive new versions of the software for one to two years until the maintenance agreement expires.

**Why “Free Software” is better than “Open Source”**

While free software by any other name would give you the same freedom, it makes a big difference which name we use: different words convey different ideas.

In 1998, some of the people in the free software community began using the term “open source software” instead of “free software” to describe what they do. The term “open source” quickly became associated with a different approach, a different philosophy, different values, and even a different criterion for which licenses are acceptable. The Free Software movement and the Open Source movement are today separate movements with different views and goals, although we can and do work together on some practical projects.

The fundamental difference between the two movements is in their values, their ways of looking at the world. For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one. As one person put it, “Open source is a development methodology; free software is a social movement.” For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution. (The two terms describe almost the same category of software, but they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement. For the free software movement, free software is an ethical imperative, essential respect for the users' freedom. By contrast, the philosophy of open source considers issues in terms of how to make software “better”—in a practical sense only. It says that nonfree software is an inferior solution to the practical problem at hand. Most discussion of “open source” pays no attention to right and wrong, only to popularity and success; here's a typical example.)

When we call software “free,” we mean that it respects the users' essential freedoms: the freedom to run it, to study and change it, and to redistribute copies with or without changes. This is a matter of freedom, not price, so think of “free speech,” not “free beer.”

Practical Differences between Free Software and Open Source

In practice, open source stands for criteria a little looser than those of free software. As far as we know, all existing released free software source code would qualify as open source. Nearly all open source software is free software, but there are exceptions. First, some open source licenses are too restrictive, so they do not qualify as free licenses. For example, “Open Watcom” is nonfree because its license does not allow making a modified version and using it privately. Fortunately, few programs use such licenses.

Second, and more important in practice, many products containing computers check signatures on their executable programs to block users from installing different executables; only one privileged company can make executables that can run in the device or can access its full capabilities. We call these devices “tyrants”, and the practice is called “tivoization” after the product (Tivo) where we first saw it. Even if the executable is made from free source code, the users cannot run modified versions of it, so the executable is nonfree.

The criteria for open source do not recognize this issue; they are concerned solely with the licensing of the source code. Thus, these unmodifiable executables, when made from source code such as Linux that is open source and free, are open source but not free. Many Android products contain nonfree tivoized executables of Linux.

The fundamental purpose of open source licensing is to deny anybody the right to

exclusively exploit a work.

When a consumer purchases a piece of software, say, Microsoft Excel, she acquires,

along with the physical copy of the software and the manual (if there are such physical copies), the right to use the software for its intended purpose—in this case, as a spreadsheet program. By opening the plastic wrap on the box, the consumer

becomes bound by the so-called “shrinkwrap license” under which she is bound not to copy the work (beyond the single copy made for her own use), not to make derivative works based on the work, and not to authorize anyone else to do either of these two things. The elimination of these three restrictions is the foundation of open source licensing.

The MIT, BSD, Apache, and Academic Free Licenses

The MIT (or X), BSD, and Apache Licenses are classic open source licensing software licenses and are used in many open source projects

The MIT and BSD Licenses were two of the earliest open source licenses.

The MIT License, probably the simplest of those licenses, imposes almost no restrictions

on licensees and no meaningful restriction at all on licensees distributing derivative

works.

By contrast, the BSD License, both pre- and post-1999, imposes explicit limitations

on distribution of both the original and derivative works.\* These limitations include

the inclusion of the enumerated terms of the license so that these limitations will also

govern the use of the derivative work: the non-endorsement provision, the copyright

notice, the acknowledgment of the creator of the original work, and the inclusion of

the disclaimer of warranties

GNU General Public License

The GNU’s General Public License, or GPL, is one of the foundation open source

licenses. Created by the Free Software Foundation (FSF), which has made many contributions

to open source coding, it is the preferred license for projects authorized by

the FSF, including the GNU Emacs Editor and the GNU C Compiler, among literally

scores of others, including the GNU/Linux kernel.

The intentions behind the license and the premise underlying it are explained in the

license’s preamble, which is included here in its entirety. The preamble follows the

copyright notice,† and a notice that prevents modifications, ironically enough, to the

license itself: “Everyone is permitted to copy and distribute verbatim copies of this

license document, but changing it is not allowed.” While the license permits the creation

of derivative works from the licensed code, it does not permit the creation of

derivative licenses from the license itself.

This preamble clearly and concisely sets out the three main purposes of the GPL. The

first, and by far the most important, is to keep software free, in the sense that it can

be distributed and modified without additional permission of the licensor. This

imposes a mirror-image restriction on the licensee: while the licensee has free access

to the licensed work, the licensee must distribute any derivative works subject to the

same limitations and restrictions as the licensed work. The second purpose of the

GPLis to ensure that licensees are aware that software under the license is distributed

“as is” and without warranty. This purpose is not unique to the GPL, as we

have seen. The third purpose (which is really a variant of the first) is that the licensed

software be free of restrictive patents: to the extent that a patent applies to the

licensed software, it must be licensed in parallel with the code. As we discussed in

Chapter 1, a given piece of code may be subject to both a copyright and a patent. In

order for the GPLto function properly, both copyright and patent licenses must be

subject to the terms of the GPL.

These licenses, as applied to the original licensed code, allow that code to be used in

proprietary software and do not require that open source versions of the code be distributed.

Code created under these licenses, or derived from such code, may go

“closed” and developments can be made under that proprietary license, which are

lost to the open source community. For the same reason, however, these licenses are

very flexible and compatible with almost every form of open source license.

GNU Lesser General Public License

The GNU Lesser General Public License (LGPL) is another license created by the FSF

for the purpose of permitting a certain class of programs, generally subroutine libraries,

to be licensed under an FSF license but be permitted to link with non-GPLsoftware

programs. Subroutine libraries provide various functions to other programs,

and because as part of their function they link with such programs, the resulting program

plus library could be considered as a legal matter to be a derivative work.

Accordingly, if the other program were licensed under a proprietary license and the

library under the GPLand the program and library were distributed together under

the proprietary license, the GPLwould be violated, as the program plus library

would be considered a derivative work that would be subject to limitations on copying,

distribution, and modification that are inconsistent with the GPL.\*

The Mozilla Public License 1.1 (MPL 1.1)

In January, 1998, Netscape Communications decided to release the binary code of its

Communicator web-brower for free. Less than 24 hours later, it decided to release

the Communicator source code as well. As a result, at the same time that Netscape

This is the Title of the Book, eMatter Edition

Copyright © 2004 O’Reilly & Associates, Inc. All rights reserved.

The Mozilla Public License 1.1 (MPL 1.1) | 63

was addressing the many technical problems with transitioning Communicator into

open source (including removing substantial amounts of code written by third parties

who were unwilling to have their code “open sourced”), Netscape had to address

the complex licensing issues involved.\*

The Netscape Public License (NPL) and the Mozilla Public License (MPL) were the

result of these efforts.† The NPLwas substantially similar to the MPL, but it reserved

certain rights to Netscape, most importantly, the right on the part of Netscape to

relicense code developed by third parties that is derived from Communicator code

under a proprietary or other license. Third-party modifiers of NPL-licensed code

could thus lose any benefits that might flow from their contributions, without the

guarantee, as for instance under the GPL, that their code will remain available to the

community of programmers. The MPLdoes not contain the particular provisions

embodying this grant of rights to Netscape.

The MPLconstitutes an interesting hybrid of the ideas of the GPLand the BSD

licenses already described. While code that falls within the scope of what the license

describes as “Covered Code” is subject to many of the restrictions present in the

GPL, such as the requirement that it be made available in open source form, the

MPL, through its Section 3.7, also permits the use of such “Covered Code” in

“Larger Works,” meaning that MPL-licensed code can be combined with code

licensed under another license. This latter result is expressly prohibited by the GPL

and permitted by the BSD License. The MPL establishes something of a middle

ground between the two licenses.

The GPL License and the Free Software Philosophy

The impact of the GPL, and its offshoot, the LGPL, on the development of software

cannot be overstated. The GPLproject that grew up with the license, the GNU/

Linux constellation of applications, better known simply as Linux,\* has seen its

acceptance by users grow steadily from the early 1990s to the point where it now

poses the only significant competitor to the Windows operating system.

This success, depending on your point of view, arises either because of, or despite,

the fact that the GPLbars any development of software from GPL-licensed software

that is not itself GPLlicensed. The GPLseems to embody the maxim that “Freedom

in a commons brings ruin to all.”† By requiring that all contributions to GPLprojects

be themselves GPLlicensed, the GPLensures not only that these contributions are

available to other programmers (or at least those programmers willing to work

within the GPLframework) but also encourages contributions from those programmers

to whom it is important that their contributions be made, and remain, “free,”

as that term is used in the GPL.

**Classic Proprietary License**

The classic proprietary license needs relatively little explanation. The license does not

need to distinguish, for example, between source and binary code: the source code is

simply not made available. The license need not distinguish between distribution of

derivative and original works: with one very narrow exception, neither is permitted.

Proprietary licenses, like the one described below, may contain “open source” licensed

software (under the more permissive licenses, like the MITand BSD Licenses), but the

code they license may not be included in any open source project, unless the code is

licensed under a parallel non-proprietary license that permits such use

Sun Community Source License

In addition to the varieties of open source and free software licenses already discussed,

there are licenses that do not fall within the Open Source definition but

incorporate some elements of open source principles. The Sun Community Source

License (SCSL) is one such license, developed by Sun to incorporate some of the benefits

of open source development into two proprietary Sun products, Jini and the

programming language Java. Sun has been very careful not to characterize this

license as an open source license; the license clearly is not such a license. The most

important distinction between this license and open source licenses is the Sunimposed

compatibility requirement. While users are free to modify the licensed

work, they may not deploy modified versions of that work without compatibility

compliance being certified by the licensor, i.e., Sun. This puts substantial limits on

the applicability of the open source model to Sun’s project. Such restrictions may,

This is the Title of the Book, eMatter Edition

Copyright © 2004 O’Reilly & Associates, Inc. All rights reserved.

Sun Community Source License | 121

however, be justified by Sun’s desire to ensure that Java maintains its cross-platform

portability, which incremental tweaks in individual versions could quickly undermine.

In addition, commercial use of the SCSL-licensed code may require the payment

of a royalty, which is, again, inconsistent with the open source model.

Despite the fact that the SCSL is not an open source license, it embodies an innovative

licensing principle—lying somewhere between the classic proprietary and the

open source models already described—and is a natural development of the developer-centric

licenses already described, such as the Mozilla Public License and the

Artistic License.

Microsoft Shared Source Initiative

Microsoft has historically wrung great profits from proprietary software licensing. Its

business model, along with its substantial profit margins, is completely dependent

upon licensing access to the software that it controls.As the

open source movement continues to gain steam, Microsoft, like many other software

companies, has felt pressure to provide public access to its source code.

Microsoft cannot easily turn to existing open source licenses and communities in

order to solve this problem

Within this system, Microsoft has defined five key source code licensing attributes:

1. The ability to view and reference source code without changing it

2. The ability to enhance debugging with source code access

3. The ability to modify source code for local use only

4. The ability to distribute products based on modified sources for non-commercial

purposes

5. The ability to commercialize products built on modified source code

Using these attributes, Microsoft has carefully tailored a number of software licenses

that grant more or less restricted access to the source code for many of its software

products, depending upon a number of variables such as what country the licensee

resides in; how important the product is to Microsoft’s core business; and whether

the software is being used for commercial purposes, charitable use, or academic

research. For some products, such as Windows, there may be literally dozens of different

licensing options.

Because the Microsoft Shared Source Initiative is so complex, and each license is the

result of relatively laborious negotiation within Microsoft and between Microsoft

and its users and developers—with product and location specificity built into each

license—the project has none of the simplicity or transparency of open source and

free software licenses. It is, at least at this time, little more than a branded extension

of Microsoft’s current commercial licensing practices

In terms of placing various licensing models on a spectrum, the GPL or the BSDmodel

license would fall on one end, depending on the nature of the “freedom”

being measured; obviously, the classic proprietary license would fall upon the other,

in terms of the restrictions imposed on licensees. In the continuum would fall the

Perl, the MPL, the SCSL and the other licenses already described. The Microsoft

Shared Source Initiative falls quite near the classic proprietary model in its function:

not a surprising result, considering that Microsoft is by far history’s largest beneficiary

of the proprietary software licensing model. But, nonetheless, it has already, at

least with regards to some applications, moved closer to a true open source model,

and the Initiative is a project worth watching.

Free vs. Open-Source vs. Proprietary Software Licenses

In this report, we refer to “free,” “open-source” and “proprietary” software. Definitions vary from source to source, but we define them as follows:

Free: Software that is licensed for use free of charge. This includes free versions of proprietary software.

Open-Source: Software that usually is licensed for use free of charge, and with source code that is made available for modification and distribution. While the license cost is often free, this software usually incurs implementation, development and other costs.

Proprietary: Software that incurs either one-time (perpetual) or recurring (subscription) fees for the license to use the software. Users cannot modify, distribute or resell the source code.

The table below lists the top advantages and disadvantages of each license type:

Advantages Disadvantages

Free

No license fees

No contract

Limited feature set

Limited or no customer support

Open-Source

Customizable

Low or no license fees

Moderate to high TCO

Limited or no customer support

Proprietary

Industry-specific feature set

Dedicated customer support

Upfront or recurring license fees

Contract usually required

For commercial open-source and proprietary products, vendors might also offer more specific license models. Under these models, fees are based on certain characteristics of your company or projects, such as:

The number of projects the software will support in a certain time frame

The number of employees who will use it; a project’s total value

Your company’s annual revenue

Now that you know the basics, you might be wondering: Does the type of license users choose correlate to their overall satisfaction with the product? According to our analysis, the answer is “yes.”

Proprietary Users Are 165 Percent More Satisfied

Overall, significantly more construction project managers who use proprietary software are satisfied with their product than their counterparts who use free and open-source products: 69 percent say they are “completely satisfied” with their proprietary software, while 31 percent are “mostly satisfied.”

User’s Overall Satisfaction, by Software License Type

User’s Overall Satisfaction, by Software License Type

Conversely, among free and open-source software users, 26 percent are “completely satisfied,” 71 percent are “mostly satisfied” and 3 percent are not satisfied at all.

While project managers in our sample are more likely to be satisfied with a proprietary product, there are still many satisfied free construction project management software users. Clearly, such products are still a viable option for some construction companies.

Drilling into qualities of the products themselves, we again find that higher numbers of proprietary software users are satisfied than are free and open-source users. The spread is much lower for cost, however, where only 10 percent more proprietary users are satisfied.

User’s Satisfaction With Software, by Category

User’s Satisfaction With Software, by Category

That proprietary users are only marginally more satisfied on cost could be explained by the simple fact that no one enjoys paying invoices, but TCO is more likely the reason.

Just as the name implies, “total cost of ownership” means the total cost paid over time to own, use and maintain the software itself, including related hardware, customer support, training, management, downtime and more. Interestingly, whether you’re using an off-the-shelf proprietary product with a subscription license or an open-source product with a free license that incurs custom development costs, TCO tends to equalize over time—so companies end up paying about the same amount for either license type if they own the software long enough.

(This, of course, does not apply to software that is completely free to use and requires no additional hosting, customization, support and maintenance.)

Integration Is Top Challenge for All Software Users

Project managers in our sample experience some challenges with their construction software, no matter how satisfied they are or how it is licensed. Paid-license users experience fewer challenges, however, and for all users, integration with other systems is the top challenge: reported by 57 percent of free and open-source users and 23 percent of proprietary users.

Users’ Top Software Challenges, by License Type

Users’ Top Software Challenges, by License Type

Choose the License Type That Best Fits Your Business

It is important to understand software licensing, because the license type you choose impacts both costs and the way your company can use the software. There is no ideal license type for all users, however. The one that is best for your company depends on your requirements and resources. Buyers should consider the following:

⇒ Talent. Large construction companies with in-house IT teams typically have the resources to implement, configure and maintain an open-source system, while small companies often need to make room in the budget to outsource such tasks to a qualified consultant.

⇒ Customizations. If you have complex workflows and processes—and the budget to pay for custom development—an open-source product may be the right choice, since the code and functionality can be modified as needed. Proprietary software is typically more limited in its customization options.

⇒ Functionality. For project managers who need all the bells and whistles—such as punch lists, submittal management and drawing markups—a proprietary application is probably the better choice, because it is more likely to have industry-specific functionality built right in.

⇒ Total cost of ownership. It is a common misconception that “open-source” means “zero cost.” When you factor in implementation, customization and other costs, open-source products have costs comparable to proprietary software. Don’t choose open-source software if the only reason is to save money.

2 Licenses and the Definition of Free Software - FreeHelia

Based on the level of limitations or fees on use, study and distribution of intellectual property, licenses are often categorized as

proprietary

free (beer)

Free (speech)

A product that is “free as in beer” does not cost anything to use for some purpose, but may contain restrictions, for example on distribution or modification. Proprietary means that use is strictly limited by license and usually there are fees on use. Free as in speech, or “Free” with capital F, means that the license meets the strict freedom criteria as defined by various free software organizations.

For example, license of Microsoft Word is proprietary, license of Adobe Acrobat Reader is free (as in beer) and the license of Linux is Free (as in speech). The somewhat funny English terms Free speech and free beer are to make distinction between no cost software and Free software movement. Those words are also the ones used by the Free Software Foundation.

Free Software Foundation, the organization behind the most popular Free software license (GPL), defines Free Software as software that user can

Run for any purpose (without costs or limitation)

Study (requires source code)

Distribute (and keep the money)

Modify (and distribute modified)

(Free Software Foundation 2003)

Open Source Initiative, whose criteria is often implicitly seen as the definition of Free by categorizing licenses to OSI approved, gives a ten point criteria for license evaluation (Open Source Initiative 2005). In content, it is very similar to Free Software Foundations criteria.

Much of what Helia produces and uses is not software. Teachers create materials for courses, students write reports of solved problems. Books and materials made elsewhere are used by courses.

Many free licenses are made for software, and even though they can be used for other intellectual property, their text can become confusing. For example, what is the source code of book, mentioned by GPL and other licenses. Because of this confusion, many non-software licenses have been created lately. Most popular Free non-software licenses are GNU Free documentation license (FDL) and Creative Commons licenses. Creative Commons, in addition to FDL, is now suggested by the Free Software Foundation (Free Software Foundation 2003).

2.1 Popularity of Licenses

Popularity of software licenses can be compared by looking at the total number the license is used in any project, or we can concentrate just on the most important and most popular projects.

License distribution of open source projects

Illustration 1. License distribution of open source projects according to freshmeat.net.

An important Free software directory freshmeat.net keeps statistics of licenses used. The most used license is the GNU General Public License (GPL), which is more used than all the other licenses together. GPL is used by 70% of projects. All ten most popular projects in freshmeat.net use the GPL. (OSDN 2003)

GNU Lesser General Public License (LGPL) and the BSD license are next in popularity. Almost 10% of projects use either LGPL or the BSD license. (OSDN 2003) These licenses are similar in spirit so that they grant freedoms much like GPL, but don’t put many restrictions to protect those freedoms. For example, you can take BSD licensed source code, put it in your proprietary, non-free product and never release the source. (FreeBSD Team 1994-2004) Free Software Foundation (gnu.org), the organization behind LGPL suggests this license for some software libraries. BSD license is used by a whole unix-like operating system, BSD.

LGPL and (clarified) BSD license are compatible with GPL. This means that you can take an LGPL program and include it in your fully GPL licensed work. This reduces license complexity even further. As the list made by GNU shows, many popular open source licenses are compatible with GPL.

Other licenses, such as the Artistic license or Mozilla Public license, have less than 2% share each. In addition to this, some licenses are either GPL-compatible or author as dual licensed the work with GPL. Thus, these other licenses have very little significance. (Illustration 1)

2.2 Freedom vs Protection of Rights

Illustration 2. Freedom for users versus protection of intellectual property.

In my view, the most important criteria for choosing a license are freedoms as a user and protection for intellectual property. If we accept the basic assumption of free market, greedy consumers, any user would probably have a program for no cost, with source code and unlimited redistribution rights. When distributing the software, the user would become a vendor. Many vendors would like to limit the use of software to get maximal profits for selling licenses to use that software.

In the illustration, I have categorized proprietary and some of the most common Free software licenses according to this criteria. Typically, proprietary software licenses limit user rights as much as law permits and even more – often any resale is prohibited and vendor could deny use at any time without reason. On the other hand, BSD (Berkley System Distribution License) and LGPL (Lesser GPL) allow unlimited use with almost no restrictions. Contrary to a typical initial impression, software licensed this way does not usually end up free. Instead, someone exercises the rights given by the license and redistributes the software under his own, more restrictive license, making his version proprietary. Apple OSX is a good example of this, as it is a highly popular proprietary operating system and a distribution based on BSD software. GNU General Public License (GPL) grants users many rights, but states rules to protect those rights. It states that when redistributing software under the GPL, all recipients must be given the same GPL rights that redistributor got when he received a copy of the software. (Illustration 2)

2.3 Licensing risks

Using a licensed product is a legal agreement and not without risk. Risks common to Free and non-free licenses include trademarks, license conflicts and patents.

Trademarks can limit what you can call a product. With some licenses, you can have the program and source free, but if you distribute it, you must come up with another name. Examples of trademark protected Free software include the most popular database management system MySQL and the most popular Linux distribution Red Hat (Freshmeat 2003, MySQL AB 2003, Red Hat Inc 2003). It is not fully clear if this kind of limitations to GPL licensed software are valid. Theoretically, this could be changed by Free Software Foundation (gnu.org) for GPL licensed products. The GPL usage notice allows use under GPL version 2 or any later version (Free Software Foundation 2003). In practice, most organizations will likely want to by on the safe sides to enjoy all public relations benefits of Free software, and because removing trademarks from software is often quite straightforward.

Releasing other’s property could result in the products license to be invalid. For example, if I sold you the right to broadcast next summer Olympics, you still would not have the rights, because I can not sell something I don’t own. At the time of writing, a failed Linux distributor SCO claims to own parts of Linux kernel. As they have failed to provide any proofs, many consider their claims false but harmful to Linux image. (Reuters 2003, Taylor 2003) BSD, a free operating system similar to Linux, won a similar case with a symbolic settlement in 1995 (McKusick 1999).

License conflict is just another form of somebody releasing code he does not own. Licenses often put some restrictions on what you can do with the software. For example, most licenses demand you don’t claim credits for other’s work. Parts of software with this restriction cannot be mixed with software whose license allows removal of credits. GPL requires modified software to be released with the same Free GPL license as the original. Because one cannot just take GPL code and put restrictions on it, Microsoft has called GPL a “viral license” and “cancer” (Microsoft 2004, BBC 2003, ZDNet UK 2005). In my opinion, the restrictions provided by GPL benefit Helia - when we develop and publish new features for software, we expect other organizations using our modifications give their improvements back to us.

Software patents are allowed in many countries, such as USA and most parts of Europe. In Finland, patenting a mathematical method (software) is not allowed by law, but is beginning to be possible because of the surprising new policies in National Board of Patents and Registration of Finland (Electronic Frontier Finland 2003, National Board of Patents and Registration of Finland 2003). Patenting software has lead to many ridiculous cases in USA and other countries. Examples include a patent for electronic commerce (Amazon 1997) and a patent for hyperlink (Sargent 1980). Also many formats are patented, such as GIF (images) and mp3 (music). Usually, a better patent-free format exists, in this case PNG for images and OGG for music.

2.3.1 Risks of Proprietary Licenses

Proprietary licenses, such as the Microsoft End User License (MS EULA), are created to by vendor, to protect vendors interest. Proprietary licenses can contain high risks for client companies. Vendors of proprietary software often aim for customer lock-in, so that even if customer is not satisfied with service or product quality, changing has become too expensive. (McHugh 1999)

The big risk is one-sided change of conditions whenever vendor wants. This is achieved through a combination of software features and licensing terms. License usually allows vendor to change conditions of a license, or terminate users right to use the program. A more used method of forcing users to accept one-sided license change is to drop support for older operating system version and have a new license for the new version. As the code is proprietary, there are no third party vendors to provide critical security patches, which makes it impossible to use the program in business environment. This kind of forced change was used to get people over to Windows XP, whose license has changed a lot from previous versions of Windows. Reverse engineering is often not allowed by the license, but this kind of limitation does not necessarily apply to Finland. Sometimes licenses contain impossible terms to reduce vendor responsibility.

Licenses are not just a cost by itself. To follow license agreements with vendor, IT support must be continuously counting licenses. To avoid buying unused licenses, IT support is forced to create many configurations for workstations. When workstations are managed by imaging (as in Helia), this creates a lot of unneeded manual work.

Multiple license agreements create a hard to manage legal portfolio. In my personal experience consulting companies, even large firms often solve this problem by forgetting it. Little effort is made to read agreements, maybe because client companies feel they could not choose vendors or negotiate the terms anyway. Many proprietary software licenses are very long, and purposefully made hard to read. Some software vendors go as far as using a tiny font for printing the agreement and using a matchbox sized window to display a ten-page agreement on computer screen.

Many of the problems of proprietary licenses don’t exist with Free licenses. A lot of Free software uses compatible licenses, and most of the software use the same license, namely the GPL. This reduces costs of managing the legal portfolio. Not only there are less licenses for company lawyers to be familiar with, but also there is an ongoing public review of the widely used licenses, especially the GPL. This results in less costs and fewer risks.

Free software gives huge benefits to users of software. What is their downside? More equal stand to negotiate price and service terms between customers and vendors makes it very hard to have artificially high profits based on customer lock-in or dominating position on the market. What some vendors may find threatening, most customers and users find beneficial.