Collaborative Project

ASPIRE
Advanced Sensors and lightweight Programmable middleware for Innovative Rfid Enterprise applications

FP7 Contract: ICT-*215417*-CP

WP3 – RFID Middleware Infrastructure

Public report - Deliverable

Licensing and reuse model

Due date of deliverable:	M6
Actual Submission date:	

Deliverable ID:	**WP3/D3.1**
Deliverable Title:	Licensing and reuse model
Responsible partner:	INRIA
Contributors:	Nadine Paolucci – INRIA Michel Cezon – INRIA Luc Laurens - INRIA
Estimated Indicative Person Months:	6

Start Date of the Project:	1 January 2008	Duration:	36 Months

Revision:	0.4
Dissemination Level:	PU

Document Information

Document Name:	**Licensing and reuse model**
Document ID:	**WP3/D3.1**
Revision:	**0.4**
Revision Date:	**26 June 2008**
Author:	**INRIA**
Security:	

Approvals

	Name	Organization	Date	Visa
Coordinator	Neeli Rashmi Prasad	CTIF-AAU		
Technical Coordinator	John Soldatos and Albena Mihovska	AIT and CTIF-AAU		
Quality Manager	Thomas Christiansen	CTIF-AAU		

Document history

Revision	Date	Modification	Authors
0.1	7 Mar 08	First draft	Nadine Paolucci, Michel Cezon
0.2	24 Apr 08	Corrected from comment from OSI	Luc Laurens
0.3	19 Jun 08	Added recommendations based on inputs received from Partners	Nadine Paolucci
0.4	26 Jun 08	Changed IPR statement and clarified requirements for documents according to GA decision	Luc Laurens

Content

Section 1 Executive summary

This deliverable presents a State of the Art study which has been conducted to list, review and assess the available Open Source Software (OSS) and documents licenses.

A preliminary chapter introduces the context linked to the ASPIRE project.

A second chapter deals with OSS licenses; it introduces differences between proprietary software and OSS, clarifies OSS definition and hightlights issues especially on compatibility.

A third chapter deals with the document licenses and proposes a Creative Commons schema.

A last chapter summarizes finding and provides recommendations on licenses to use for ASPIRE consortium regarding documents and Open Source Software.

Section 2 Context and overview

ASPIRE will research and provide a radical change in the current RFID deployment paradigm through innovative, programmable, royalty-free, lighweight and privacy friendly middleware. This new middleware paradigm will be particular beneficial to European SME, which are nowadays experiencing significant cost-barriers to RFID deployment.
European networked enterprises in general and SME in particular are still reluctant to adopt RFID, since they perceive RFID as unprofitable or too risky. This is largely due to the fact that the adoption of RFID technology incurs a significant Total Cost of Ownership (TCO). ASPIRE will significantly lower SME entry costs for RFID technology, through developing and providing a lightweight, royalty-free, innovative, programmable, privacy friendly, middleware platform that will facilitate low-cost development and deployment of innovative RFID solutions. This platform will act as a main vehicle for realizing the proposed swift in the current RFID deployment paradigm. Portions (i.e. specific libraries) of the ASPIRE middleware will be hosted and run on low-cost RFID-enabled microlelectronic systems, in order to further lower the TCO in mobility scenarios (i.e. mobile warehouses, trucks). Hence, the ASPIRE middleware platform will be combined with innovative European developments in the area of ubiquitous RFID-based sensing (e.g., physical quantities sensing (temperature, humidity, pressure, acceleration), mobile re, low-cost), towards enabling novel business cases that ensure improved business results.
The ASPIRE RFID middleware paradigm, as well as the unique and novel characteristics of the ASPIRE middleware platform requires a set of OSS components which will be researched in existing OSS communities or developed by the Consortium.

OSS licenses and documents licenses are addressed in the following sections.

Section 3 State of the art of legal aspects for open source software

3.1 Main legal features for Proprietary Software and Open-Source Software

3.1.1 Software as Intellectual Property

- Software is valuable intellectual property
- A software license is the contract between the software owner and the licensee defining terms of use of software.
- Software owners also have enumerated rights under the law to control the use and distribution of their property .
- Software owner's rights are protected by the following key legal institutions and contract law :

- in the US : The Digital Millennium Copyright Act (DCMA)[10] ,

- in Europe : the European Union Copyright Directive (EUCD in Europe, DADVSI in France), Berne Convention for the Protection of Literary and Artistic Works, WIPO Copyright Treaty.

About Copyright and Related Rights :

The attention of the Secretariat of WIPO (World Intellectual Property Organization) has been drawn to the fact that certain organizations issue certificates purporting to grant copyright protection. It should be noted that these certificates do not create any right. The Secretariat recalls that, by virtue of the Berne Convention for the Protection of Literary and Artistic Works, works are protected without any formality in all the countries party to that Convention. This means that international copyright protection is automatic, it exists as soon as a work is created, and this principle applies in all the countries party to the Berne Convention.

3.1.2 Typical Proprietary Software License

- Fairly standard terms
- Source code availabillity

- source code is not provided – trying to figure out inner workings of software through reverse engineering or decompiling of operating mode is forbidden ; or

- source code is provided – may or may not include permission to create modifications and enhancements

3.1.3 Proprietary Software Licence terms - Licensees

- Restrictions on dissemination
- Licensee and users are strictly defined. Licensee has no right to share with those not defined as licensee users in the license
- Licensor indemnifies licensees against third party infringement claims
- Licensor have to sign a new licence each time a new licensee obtains the code

3.1.4 Proprietary Software licence terms - Warranty and support

- Warranties provided

- for defects in media and existence of viruses

- possibility to negotiate for warranties

- Maintenance and support terms included (although may be in separate document)

3.1.5 Open Source Software – Main features

- Non-proprietary software which may or may not be used commercially
- Tpically licensed under an open source licence (licence terms differ from proprietary software licence terms), licence terms are not standard
- Source code is generally made available (legal restriction on reverse engineering do not apply)

3.1.6 Open Source Software licence – Licensees

- Original software owner or developper chooses to limit the rights that he asserts over licenses
- Lcensees, subject to license terms can :
  - mke and distribute copies of software
  - build upon software to create modification or other works

3.1.7 Open Source Software licence - Source Code

- Surce code always provided (to original product)
- Licensee can modify or enhance source code (create « derivative works ») or include source code with other licence types (create « larger works »)
- licensee may be required to share modifications with the world (in source and/or binary form), but not necessarily
- licensee may be prohibited from charging royalties for derivative and larger works, but not necessarily

3.1.8 Open Source Software licence – Warranties and Support

- Generally, software are provided « AS IS » with no warranties, warranties excluded
- No indemnification
- No maintenance or support

3.1.9 Open Source Software licence

- Terms include :

- user freedom to distribute and/or modify

- « viral » license, source code is always made available to the world

- must retain copyright notices and warranty disclaimer

3.2 Open Source legal landscape

3.2.1 The Legal Nature of Open Source

Open source software can be described on many levels. On one level it is a powerful organizational tool for collaborative software development and maintenance.[1] On another it is recognized as an ideological set of beliefs regarding the availibility of source code and the commercialization of computer software.[2]

From a legal perspective open source software refers to software licensed under an « open source » licence and is distinguishable from proprietary software as well as from freeware and public domain software. Unlike freeware and public domain software, open source is subjected to a license agreement that places legal obligations on the licensee of the software.[3]

As a result, open source software should never be mistaken as being « free » of legal obligations. But, the legal obligations imposed by open source licenses are significantly different from those imposed by traditional proprietary software licenses.

Traditional proprietary software licenses generally place major restrictions on the use of software by the end user, in particular the source code of the software. In contrast, open source licenses create a more « open » legal environnement generally characterized by the availibility of both source and binary code versions of the software, the right to modify the software and the right to distribute those modifications.

The boundaries of the « open » legal environment created by open source licenses are defined by the Open Source Definition[4].

3.2.2 The Open Source Definition

The Open Source definition is promulgated by an organization called the Open Source initiative OSI. The OSI approves licenses as « OSI Certified » based on their compliance with the Open Source Definition. Once OSI certified, licenses are generally recognized as being « open source » licenses. The Open Source Definition establishes a number of criteria that a license must meet before it is considered to be an « open source » license.

3.2.2.1 The core criteria of the Open Source Definition include the following

Source code availability

The licence must allow for distribution of the software in source code (human readable) and object code (machine readable) forms. If the software is distributed only in object code form, there must be a well « publicized » means of obtaining the source code and object code for no more than a reasonable reproduction cost. The source code must be in the preferred form for modification by a programmer.

. **Free redistribution**

The licence will not prohibit a licence from further licensing the software as part of an aggregated offering containing code from other sources, nor require that a royalty or other fee be paid in exchange for the software.

Derived works

The license must allow modifications and derived works of the licensed software and must allow any modifications or derived works to be distributed under the same license terms as the original software.

3.2.2.2 Additional criteria of the Open Source Definition require

Source code integrity (the license may require that derived works of the software be distinguished from the original program)

Non discrimination (the license must not discriminate against any person or group, or against use in any specific field of endeavor)

Distribution of licence (the license must apply to all to whom the program is distributed without the need for execution of an additional license)

Non-Product specific (the rights attached to the program must not depend on the program's being part of a particular product)

Non restrictive (the licence must not place restrictions on other software that is distributed along with the licensed program)

Technology Neutral (the licence may not be predicated on any individual technology or style of interface)

3.2.2.3 Existing Open Source Licenses

The Open Source definition functions in much the same way as a technical specification by establishing a set of criteria that allow for multiple implementations that all meet the specification. The breadth of the criteria described has allowed a wide variety of licenses to be classified as « open source », often classified in two categories : historical and classical (See Appendix 1).

At the time of this document (february 2008) there are nearly 70 different open source licenses approved by the OSI as meeting the Open Source Definition. Some of the more well-known of these licenses include the General Public licence (GPL), Lesser General Public licence (LGPL), MIT, BSD and APACHE Licenses.

In addition, other open source licenses exist and have not been approved by OSI. Nevertheless, some of these licenses appear to contain terms that comply with the Open Source Definition. Among them we can identify the CeCILL family Licenses[5] (CeCILL, CeCILL-A, CeCILL-B) compatible with GNU GPL, aims to be better suited for French laws[6]. Others, however, contain terms that diverge significantly from the requirements of the Open Source Definition.

Although plenty of free software licenses exist already, they are mostly written in English, from the point of view of the U.S. legal system, which can pose a problem in countries where the legal system is based on different assumptions.

> *Example of CeCILL license*
>
> *Therefore, the new license, known as CeCILL, is intended to make free software more compatible with French law in two areas where it differs significantly from U.S. Law : copyright and product liability.*
>
> *The name CeCILL is derived from the names of the three research institutes that created it -- the French Atomic Energy Commission (CEA), the National Center for Scientific Research (CNRS) and the French National Institute for Research in Computer Science and Control (INRIA) -- and the French terms for free software is « logiciel libre ».*

Given the variety of open source licenses, the potential implications of using, modifying and distributing open source software varies greatly from license to license. While a detailed examination of the variety of open source licenses is needed, an illustrative example is the distinction between copyleft and non-copyleft open source licenses.

> *« Copyleft is a general method for making a program free software and requiring all modified and extended versions of the program to be free software [...] Copyleft says that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it »*[7]

Much has been made of the so called « viral » effect of copyleft licenses such as the GPL. These licenses contain terms that can obligate a licensee to make publicly available the source code of proprietary software that is distributed with open source software licensed under these licenses.[8]

Non- copyleft licenses, such as the BSD and MIT licenses do not contain viral provisions and generally only obligate the licensee to provide certain attributions and notices when redistributing the software.[9]

These differences demonstrate that the terms of open source licenses are not created equal and reveal the importance of understanding the terms of any particular open source license before using software obtained under that license.

3.2.3 Ownership and Licensing Issues

Intellectual Property Rights and licensing can be considered as a postulate to the compatibility question : copyright, licensing, exploitation, complex ownership, open source licensing, contract, license compatibility should be subjected to further development.

Here is a brief summary of the those key elements :

1. Software is property

2. Software is protected under copyright law

3. The ownership of software can be determined by a technical legal examination of any contract under which it was produced, and other legally relevant circumstances

4. Copyright law says that by default only the owner of software may copy, adapt or distribute it

5. The owner of software can agree to let another person copy, adapt or distribute the code – this agreement is called a license

6. Open source licenses grant these rights to anyone who chooses to take them up, with certain conditions

7. Open source licenses aim to create a community of contributors who will fix and develop the software

8. Combining two pieces of software code under different licenses can be complex

9. All projects which produce software need to keep complete, detailed records of the licensing and ownership of contributions to that software

3.2.4 Typology of the most commons OS licenses

3.2.4.1 The Open Source Initiative

http://opensource.org/licenses/category

Open Source Licenses which have successfully gone through the approval process and comply with the Open Source Definition are listed here :

- Licenses by name
- Licenses by category

3.2.4.2 Selected extract from « Open Sources : Voices from the open source revolution » (Bruce Perens)

Analysis of Licenses and Their Open Source Compliance

To understand the Open Source Definition, we need to look at some common licensing practices as they relate to Open Source.

Public Domain

A common misconception is that much free software is *public-domain*. This happens simply because the idea of free software or Open Source is confusing to many people, and they mistakenly describe these programs as public-domain because that's the closest concept that they understand. The programs, however, are clearly copyrighted and covered by a license, just a license that gives people more rights than they are used to.

A public-domain program is one upon which the author has deliberately surrendered his copyright rights. It can't really be said to come with a license; it's your personal property to use as you see fit. Because you can treat it as your personal property, you can do what you want with a public-domain program. You can even re-license a public-domain program, removing that version from the public domain, or you can remove the author's name and treat it as your own work.

If you are doing a lot of work on a public-domain program, consider applying your own copyright to the program and re-licensing it. For example, if you don't want a third party to make their own modifications that they then keep private, apply the GPL or a similar license to your version of the program. The version that you started with will still be in the public domain, but your version will be under a license that others must heed if they use it or derive from it.

You can easily take a public-domain program private, by declaring a copyright and applying your own license to it or simply declaring "All Rights Reserved."

Free Software Licenses in General

If you have a free software collection like a *Linux* disk, you may believe the programs on that disk are your property. That's not entirely true. Copyrighted programs are the property of the copyright holder, even when they have an Open Source license like the GPL. The program's license grants you some rights, and you have other rights under the definition of *fair use* in copyright law.

It's important to note that an author does not have to issue a program with just one license. You can GPL a program, and also sell a version of the same program with a commercial, non-Open-Source license. This exact strategy is used by many people who want to make a program Open Source and still make some money from it. Those who do not want an Open Source license may pay for the privilege, providing a revenue stream for the author.

All of the licenses we will examine have a common feature: they each disclaim all warranties. The intent is to protect the software owner from any liability connected with the program. Since the program is often being given away at no cost, this is a reasonable requirement--the author doesn't have a sufficient revenue stream from the program to fund liability insurance and legal fees.

If free-software authors lose the right to disclaim all warranties and find themselves getting sued over the performance of the programs that they've written, they'll stop contributing free software to the world. It's to our advantage as users to help the author protect this right.

The GNU General Public licence

The provisions of the GPL satisfy the Open Source Definition. The GPL does not require any of the provisions permitted by paragraph 4 of the Open Source Definition, *Integrity of the Author's Source Code*.

The GPL does not allow you to *take modifications private*. Your modifications must be distributed under the GPL. Thus, the author of a GPL-ed program is likely to receive improvements from others, including commercial companies who modify his software for their own purposes.

The GPL doesn't allow the incorporation of a GPL-ed program into a proprietary program. The GPL's definition of a proprietary program is any program with a license that doesn't give you as many rights as the GPL.

[There are a few loopholes in the GPL that allow it to be used in programs that are not entirely Open Source. Software libraries that are normally distributed with the compiler or operating system you are using may be linked with GPL-ed software; the result is a partially-free program. The copyright holder (generally the author of the program) is the person who places the GPL on the program and has the right to violate his own license. This was used by the KDE authors to distribute their programs with *Qt* before Troll Tech placed an Open Source license on *Qt*. However, this right does not extend to any third parties who redistribute the program--they must follow all of the terms of the license, even the ones that the copyright holder violates, and thus it's problematical to redistribute a GPL-ed program containing *Qt*. The KDE developers appear to be addressing this problem by applying the LGPL, rather than the GPL, to their software.]

The GNU Lesser General Public licence

The LGPL is a derivative of the GPL that was designed for software libraries. Unlike the GPL, a LGPL-ed program can be incorporated into a proprietary program. The C-language library provided with Linux systems is an example of LGPL-ed software--it can be used to build proprietary programs, otherwise Linux would only be useful for free software authors.

An instance of an LGPL-ed program can be converted into a GPL-ed one at any time. Once that happens, you can't convert that instance, or anything derived from it, back into an LGPL-ed program.

The rest of the provisions of the LGPL are similar to those in the GPL--in fact, it includes the GPL by reference.

The X, BSD, and Apache Licenses

The X license and its relatives the BSD and Apache licenses are very different from the GPL and LGPL. These licenses let you do nearly anything with the software licensed under them.

The most important permission, and one missing from the GPL, is that you can take X-licensed modifications private. In other words, you can get the source code for a X-licensed program, modify it, and then sell binary versions of the program without distributing the source code of your modifications, and without applying the X license to those modifications. This is still Open Source, however, as the Open Source Definition does not require that modifications always carry the original license.

Many other developers have adopted the X license and its variants, including the *BSD (Berkeley System Distribution)* and the *Apache* web server project. An annoying feature of the BSD license is a provision that requires you to mention (generally in a footnote) that the software was developed at the University of California any time you mention a feature of a BSD-licensed program in advertising.

The Artistic licence

Although this license was originally developed for Perl, it's since been used for other software. It makes requirements and then gives you loopholes that make it easy to bypass the requirements. Perhaps that's why almost all Artistic-license software is now dual-licensed, offering the choice of the Artistic License or the GPL.

Section 5 of the Artistic License prohibits sale of the software, yet allows an aggregate software distribution of more than one program to be sold.

The Artistic License requires you to make modifications free, but then gives you a loophole (in Section 7) that allows you to take modifications private or even place parts of the Artistic-licensed program in the public domain!

The Netscape Public License and the Mozilla Public licence

NPL was developed by Netscape when they made their product *Netscape Navigator* Open Source. Actually, the Open-Source version is called *Mozilla*; Netscape reserves the trademark *Navigator* for their own product.

An important feature of the NPL is that it contains special privileges that apply to Netscape and nobody else. It gives Netscape the privilege of re-licensing modifications that you've made to their software. They can take those modifications private, improve them, and refuse to give you the result. This provision was necessary because when Netscape decided to go Open Source, it had contracts with other companies that committed it to provide *Navigator* to them under a non-Open-Source license.

Netscape created the *MPL,* or *Mozilla* Public License, to address this concern. The MPL is much like the NPL, but does not contain the clause that allows Netscape to re-license your modifications.

The NPL and MPL allow you to take modifications private.

[Many companies have adopted a variation of the MPL for their own programs. This is unfortunate, because the NPL was designed for the specific business situation that Netscape was in at the time it was written, and is not necessarily appropriate for others to use. It should remain the license of Netscape and Mozilla, and others should use the GPL or the or X licenses.]

3.2.4.3 Choosing a License

The table below gives a comparison of licensing practices

License	**Can be mixed with non-free software**	**Modifications can be taken private and not returned to you**	**Can be re-licensed by anyone**	**Contains special privileges for the original copyright holder over your modifications**
GPL				
LGPL	X			
BSD	X	X		
NPL	X	X		X
MPL	X	X		
Public Domain	X	X	X	

3.2.5 Compatibility

3.2.5.1 Meaning of compatibility

As seen above, within the Legal Aspects of Open Source Software, Compatibility means License Compatibility.

One of the consequences of licensing is that care must be exercised when combining source code from two or more different projects which are licensed under different licenses.

For example : if two software products, one of which is licensed under the MPL, the other under the GPL, are combined, the resulting product must be licensed consistently with the requirements of both the MPL and the GPL. If the requirements of these licenses are per se (by itself) inconsistent then there is no legal basis on which the output product can be licensed.

Compatibility between two licenses refers to whether or not software licensed under those licenses may be combined to produce a work which can be licensed consistently with requirements of both licenses. Typically it is important to know whether a license is « GPL compatible» so that software the subject of it (of the said licence) can be combined with software the subject of the GPL.

3.2.5.2 License Compatibility terms

Even if Rights are basically the same in all licenses according to the Open Source Initiative conditions (see above), and despite the commonalities, there is not one, but multiple license models that are usually not compatible because of the differences related to the permission to redistribute.

Open source licenses include a variety of terms that give rise to different obligations on the open source licensee. These obligations are based on the terms themselves and on the scenarios in which the particular open source software is being used.

As a result, no single analysis is sufficient for all open source licenses or for any one open source license in all scenarios.

Instead it is necessary to carefully analyze the terms of the particular open source license and the scenario in which the open source software is being used. This analysis is possible when both the open source license and the scenario in which the open source software is being used are known.

Without this knowledge a licensee is exposed to an increased risk of breaching the terms of an open source license[1].

For example, certain open source licenses, including the GPL, state that a breach of the license triggers an immediate termination. See the GPL, Version 2 §4 « You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. »

3.3 Practical Difficulties

3.3.1 Why different OSS licenses matter?

Licensing issues are important to developers, but for majority of users in many circumstances they don't actually matter. OSS licenses are far more different from typical proprietary licenses than from each other. All OSS licenses permit users to use the software, modify it, and redistribute the original or modified version as much as they like. Since most users don't modify their software, and since the difference between OSS licenses primarily affect developers, users don't notice the difference in most cases.

However Licenses are an important issue to consider for OSS, because there are cases where they definitely do matter.

In writing code, a programmer may find that he wants to merge elements from two or more programs into a new program. The two programs are under different licenses. The question arises : is it possible to take this code, under different licenses, and combine them in one work without violating the terms of either of the two licenses ?

Distribution or modification of programs, including incompatibility licensed code will result in copyright infringement.

3.3.2 Copylefting vs non-copy-lefting?

There are fundamentally two kinds of OSS licenses : « copylefting » licenses and « non -copylefting » licenses. A program release under a copylefting license allows anyone to change the program – but those changes must provided to recipients under exactly the same conditions as the original. In other words, an OSS program released under a copylefting license cannot be later turned into a proprietary program by a third party. Most OSS software is released under copylefting licenses, such as the GPL and the LGPL.

There is a long argument about the advantages of copylefting vs. non copylefting licenses. Some view non copylefting licenses as « more free » because recipients of that code can do anything they want with the code -including making a modification and producing proprietary version. However, many others view copylefting licenses as « freer » than non- copylefting ones, because they ensure that all later recipients of modified versions can also modify and maintain the code. Another way of looking at this is that copylefting licenses ensure that all users and developers have more freedoms, at a cost of giving fewer freedoms to the immediate recipients.

Non-copylefting licenses allow proprietary vendors to easily incorporate the code into their products, and modify it any way they like. Thus, non copylefting licenses are often used when the goal is to promote the adoption of a standard.

This issue of which license is « better » is a long-standing debate. Indeed, sometimes the same developer will choose different licenses for different products, depending on that developer's motivations which impact on how the program can be used and supported.

3.3.3 Computer Libraries

The biggest issue with copylefting licenses involves computer libraries. For a better understanding, it is often said that some computer libraries act like foundation of a house, and anything that affects a foundation can affect everything else. Any developer of a program must ensure that all the libraries they use have compatible licenses. In particular, if a computer library is covered by the GPL, then any program that uses that library must also be released under the GPL thus, if you are intending to build a proprietary system and wish to use a library, you generally can not use a library under the GPL.

3.3.4 Patent Defense

Another relevant issue about OSS software is patent defense. Some OSS licenses require that, if you modify the program in a way that it implements a patent, and you own the patent, then you must grant all recipients the rights to use the patent. Obviously, if you own any patents, you should ensure that you wish to give this grant before ou make changes to programs covered by patent defenses.

3.3.5 Legal uncertainties and challenges

For several reasons all relevant licenses are written under US Law, where the OSS movement was born 20 years ago. This does not undermine their validity : The Munich court enforced a specific application of the GPL in 2004. However it makes legal uncertainties

- Copyright law and author rights are not applied the same way, in particular concerning specific provisions related to « communication to the public » and moral rights (right to withdraw, to modify, and to remain anonymous)

- Applicable contract Law (often US law) is difficult for European judges to appreciate, and does not comply with mandatory European provisions concerning, for example, data protection and warranty or liability clauses

- The determination of the competent jurisdiction ignores the European context

- Texts are in English only and their authors refuse for integrity reasons to give value to translations

- Question of interpretation : uncertainty surrounding the new terms of open source software licenses. Beyond the general observations, it is difficult, if not possible, to provide precise guidance about what licenses may or may not be compatible with each other. Many licenses are subjected to significant variations in their terms in pratice.

- Uncertain rights due to undocumented chain of title (=lack of detailed records of the licensing and ownership of the contributions related to a software developed by a project)

- No warranties regarding title, or indemnification against third party Intellectual Property infringement claims

Today, the CeCILL Family License and the EUPL, respectively published by CEA-CNRS-INRIA et IDABC, tackle these issues in order to facilitate OSS licensing of code produced by institutions of European Union. It reduces legal flaws in the european context and highlights the contribution of european parties in an area previously dominated by US lawyers.

Apart from license issue, there are **additional legal challenges** :

- Exercise of copyright is more difficult when a work is developed by an unstructured group of persons. Some project structures, such as the « bazaar » structure (described by Eric S. Raymond in the « the cathedral and the bazaar » 1999, based on his observations of the linux kernel development process and his experiences managing an open source project) permit input into projects by hundreds and possibly thousands of programmers.

In those situations, cross- licensing is simply not practical.

Further more, some open source groups will not cross-license works copyrighted by them. The Apache Software Foundation, for example, does not cross-license its works.

- Developers « during free time » have often a second life as employees. How far are the two activities separated ?

- OSS developers feel threatened especially by software patents, because of high patent costs (seen as a monopoly for rich entreprise) and perceived restrictions of freedom to express ideas in a visible and incremental way

- The relationship between Open Source developers and their « clients » in terms of granting support or maintenance is often experimental

- The impacts of warranty and liablility, for example in the case of patent infringement, and the possibility of insurance are not easy to forecast

Section 4 License for Documentation: Creative Commons?

(Creative Commons was founded in 2001 by cyber-law and intellectual property experts James Boyle, Michael Carroll, Eric Saltzman and Lawrence Lessig, together with MIT computer scientist Hal Abelson, and publisher Eric Eldred)

Lessig has drawn much inspiration from Richard Stallman and the Free Software Foundation,

"I think that the big lesson from Open Source Software is that you can support a platform of software development that provides great positive externalities to the world because it carries source code that people know how to change. But on the other hand you can still make money from it by bundling it or tying it into some other suite of services.

There is an equivalent struggle that is going to be necessary in the context of Open Access publishing." L. Lessig

4.1 What is Creative Commons?

Creative Commons is a non-profit organization that promotes the creative re-use of intellectual works, whether they are owned or public-domain. Creative Commons has created a set of copyright licenses that are available free of charge. These licenses indicate that copyrighted works are free for sharing, but only on certain conditions.

The Creative Commons licensing tools allow authors to define the nature of the agreement in terms of attribution (giving credit to the original source material), commercialization, derivative works, and distribution. Creative Commons enables authors and creators to label their work "Some rights reserved" or even "No rights reserved."

The original non-localized Creative Commons licenses were written with the U.S. legal system in mind, so that the wording could be incompatible within different local legislations and render the licenses unenforceable in various jurisdictions. To address this issue, Creative Commons International has started to port the various licenses to accommodate local copyright and private law. Creative Commons International is dedicated to the drafting and eventual adoption of jurisdiction-specific licenses.

As of July 2007, there are 38 jurisdiction-specific licenses (including China and Brazil), with 9 other jurisdictions in drafting process (including Greece).

4.2 Why does Creative Commons exist ?

As a matter of fact, many creators have come to prefer relying on innovative business models rather than full-fledged copyright to secure a return on their creative investment. For whatever reasons, it is clear that many users of the Internet want to share their work - and the power to reuse, modify, and distribute their work - with others on generous terms. Creative Commons is a way for people to express this preference for sharing by offering a set of licenses.

www.creativecommons.org

4.3 What is the scope of the Creative Commons licences?

Creative Commons license are based on copyright. So it applies to all works that are protected by copyright law. The kinds of works that are protected by copyright law are books, websites, blogs, photographs, films, videos, songs and other audio & visual recordings, for example.

Creative Commons licenses give you the ability to dictate how others may exercise your copyright rights—such as the right of others to copy your work, make derivative works or

adaptations of your work, to distribute your work and/or make money from your work. They do not give you the ability to restrict anything that is otherwise permitted by exceptions or limitations to copyright—including, importantly, fair use or fair dealing—nor do they give you the ability to control anything that is not protected by copyright law, such as facts and ideas.

It is important to indicate that Open Access publishing is not publishing without copyright. It is publishing with copyright exercised in a way that makes material open and available for others to build upon. It still uses copyright, but for a reason different than the reason used by proprietary

publishers who exclude people from getting access to the content.

The creative commons license enable copyright holders to grant some or all of their rights to the public while retaining others through a variety of licensing and contract schemes including dedication to the public domain or open content licensing terms.

4.4 How a Creative Commons license is constructed?

The original set of licences all grant the "baseline rights". The details of each of these licences depends on the version, and comprises a selection of four conditions :

1. • **Attribution** (by): Permit others to copy, distribute, display and perform the work and make derivative works based upon it only if they give the author or licensor the credits in the manner specified by these.

2. • **Noncommercial** or **NonCommercial** (nc): Permit others to copy, distribute, display, and perform the work and make derivative works based upon it only for non commercial purposes.

3. • **No Derivative Works** or **NoDerivs** (nd): Permit others to copy, distribute, display and perform only verbatim copies of the work, not derivative works based upon it.

4. • **ShareAlike** (sa): Permit others to distribute derivative works only under a license identical to the license that governs your work.

Mixing and matching these conditions produces 16 possible combinations, of which eleven are valid Creative Commons licenses.

4.5 Typology of the Creative Commons licenses regularly used

The license is a statement as to what others may do with your work, so you should select a license that matches what you agree for others to do with your work.

Here are the six regularly used licenses :

1. Attribution alone (by)
2. Attribution + Noncommercial (by-nc)
3. Attribution + NoDerivs (by-nd)
4. Attribution + ShareAlike (by-sa)
5. Attribution + Noncommercial + NoDerivs (by-nc-nd)
6. Attribution + Noncommercial + ShareAlike (by-nc-sa)

The following describes each of these six licenses, starting with the most restrictive license type you can choose and ending with the most accommodating license type you can choose :

Attribution Non-c ommercial No Derivatives (by-nc-nd)

Choose by-nc-nd license

This license is the most restrictive, allowing redistribution. This license is often called the “free advertising” license because it allows others to download your works and share them with others as long as they mention you and link back to you, but they can’t change them in any way or use them commercially.

Read the Commons Deed | View Legal Code

Attribution Non-commercial Share Alike (by-nc-sa)

Choose by-nc-sa license

This license lets others remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms. Others can download and redistribute your work just like the by-nc-nd license, but they can also translate, make remixes, and produce new stories based on your work. All new work based on yours will carry the same license, so any derivatives will also be non-commercial in nature.

Read the Commons Deed | View Legal Code

Attribution Non-commercial (by-nc)

Choose by-nc license

This license lets others remix, tweak, and build upon your work non-commercially, and although their new works must also acknowledge you and be non-commercial, they don't have to license their derivative works on the same terms.

Read the Commons Deed | View Legal Code

Attribution No Derivatives (by-nd)

Choose by-nd license

This license allows for redistribution, commercial and non-commercial, as long as it is passed along unchanged and in whole, with credit to you.

Read the Commons Deed | View Legal Code

Attribution Share Alike (by-sa)

Choose by-sa license

This license lets others remix, tweak, and build upon your work even for commercial reasons, as long as they credit you and license their new creations under the identical terms. This license is often compared to open source software licenses. All new works based on yours will carry the same license, so any derivatives will also allow commercial use.

Read the Commons Deed | View Legal Code

Attribution (by)

Choose by license

This license lets others distribute, remix, tweak, and build upon your work, even commercially, as long as they credit you for the original creation. This is the most accommodating of licenses offered, in terms of what others can do with your works licensed under Attribution.

Read the Commons Deed | View Legal Code

4.6 Elements to be considered when applying a Creative Commons license to a work

4.6.1 Does the work falls within the Creative Commons license ?

Creative Commons licenses apply to works that are protected by copyright. Generally, works that are protected by copyright are: books, scripts, websites, lesson plans, blogs and any other forms of writings; photographs and other visual images;

films, video games and other visual materials; musical compositions, sound recordings and other audio works.

Creative Commons licenses do not apply to things such as ideas, factual information or other things that are not protected by copyright.

4.6.2 Who can decide to apply a Creative Commons license ? (the one who hold the ownership of the rights in the work)

Before applying a Creative Commons license to a work, you need to make sure you have the authority to do so. This means that you need to make sure that the person who owns the copyright in the work agree to have the work made available under a Creative Commons license.

The creator of a work is generally the owner of copyright and so can license the work how he wishes.

When the work is made as part of an employment, then the employer probably owns the rights to the work and so only the employer can decide to apply a Creative Commons license.

When the work is made under an agreement, you need to check the terms of that agreement (for example, to see if the rights to the work were transferred to someone else).

When the work is combining pre-existing works made by different people or is made in conjunction with different people to produce something, you need to make sure that you have express and explicit permission to apply a Creative Commons license to the end result.

When the work is combining a work that is already Creative Commons-licensed then you will also have the rights provided your use is consistent with the terms of that license.

4.6.3 How does a Creative Commons license operate?

Creative Commons licenses is attached to the work and authorize everyone who comes in contact with the work to use it consistent with the license.

The license is expressed in three ways :

- a Commons Deed (a simple, plain-language summary of the license),

- a Legal Code (to ensure that the license will stand up in court),

- and a Digital Code (a machine-readable translation of the license that helps search engines and other applications identify the terms of use).

You don't need to sign anything to get a Creative Commons license—just select your license at :

http://creativecommons.org/license/

then frame your metadata and legal notice accordingly, eg. “this document is licensed under a Creative Commons [insert description] 2.5 license.”

All Creative Commons licenses are non-exclusive. This means that you can permit the general public to use your work under a Creative Commons license and then enter into a separate and different non-exclusive license with someone else, for example, in exchange for money.

REMARKS

An important point to consider is that Creative Commons licenses are non-revocable. This means that you cannot stop someone, who has obtained your work under a Creative Commons license, from using the work according to that license. You can stop offering your work under a Creative Commons license at any time you wish ; but this will not affect the rights with any copies of your work already in circulation under a Creative Commons license.

« Work » means the creative work offered under the terms of the license.

« Derivative work » means a work based on the Work or created upon the Work and other pre-existing works.

Except the uses that are authorized under the licenses, any other use of the Work remains submitted to the author’s law or any applicable law.

The exercise of any right on the Work provided by the license is a tacit consent.

« Public Domain » means Creative works in relation to which no person or other legal entity can establish or maintain proprietary interests within a particular legal jurisdiction.

This work is considered to be part of a common cultural and intellectual heritage, which, in general, anyone may use or exploit, whether for commercial or non-commercial purposes. Here, work is offered without conditions.

Section 5 Synthesis and recommendations

Further to the discussions held by the Consortium and the requirements expressed, this document relates materials collected, requirements expressed in terms of licensing for software and documentation, and gives some key about the main features of various Open Source licenses, as well as recommendations according to those requirements, in respect of the applicable Law in the framework of the ASPIRE project.

5.1 Introduction

Recommendations for licensing can not be considered without giving first a few elements of Intellectual Property Law, especially related to some rights which are part of what is generally known as Intellectual Property Rights.

5.1.1 Difference in Intellectual Property Laws

There are different applicable Governing Laws:

- Berne Convention (copyright/droit d'auteur)
- European Patent convention
- EU Software directives and member nation laws, e.g. Code de la propriété intellectuelle

They present many similarities, especially in copyright, but some significant differences including:

- software patent validity
- questions of transfer of copyright assignment
- joint copyright assignment

5.1.2 Copyright

✓Covered Action

- reproduce
- create derivative works
- distribute
- publicly display
- publicly perform

(terms are from US Copyright Act but concepts are same)

✓Key requirements

- original expression
- with some minimal amount of creativity
- fixed in a tangible medium

✓ Neither registration nor notices is required

✓Not every thing is protected by copyright law

- idea of expression merger
- minimus work

✓ Indications of copyright infringement

- substantial similarity, and
- access to infringed work

✓ Bottom line

- must have a license from the author/owner to take any of the covered actions (beyond fair use)

5.1.3 Patents

✓Covered actions

- Make / Have made
- Use / Import
- Sell / offer to sell

✓Key requirements
Novel
Non-obvious
Has utility
Described in detail

✓Required patent application and approval by patent authority (e.g. EPO or USPTO)

✓Limited monopoly
Provides right to exclude others from the above actions for a limited time
Covers processes, designs, machines, article of manufacture (differs across international laws)

✓Bottom line

- independently created inventions still require a license from any valid patent before use

5.1.4 Trademarks

- Identifies the origin of product or service
- Generally distinctive symbols, pictures or words
- Registration is required

The present document will not address trademarks.

5.2 Collected materials

5.2.1 Consortium agreement

It provisions related to IPR and rules defined by the consortium.

Article 4 “IPR and Access Right”

Article 4.2.8.3 “Open Source Software” defines that the parties acknowledge that the use within the project of software that is “Open Source”, and/or the release of Foreground upon license terms associated which such software, may have benefits for the conduct Project and promote the use and the dissemination of the resulting Foreground.
Thus the Parties agree …/… they are committed part of or all of their Foreground in the form of software as “open source” as a way of collectively contributing to the creation of a Standalone Software product …/…”

The Annex 6 gives a list of the software that is background or sideground in respect of which the parties are willing when granting Access Rights to grant access to source code for use as referred to in section 2 4.2.7.1

The consortium agreed that the rules defined for licensing will not be applicable to MELEXIS.
Article 4.2.8.1 a) “The parties explicitly agree that the Access Rights do not apply to the software developed by MELEXIS to design Integrated Circuits”

5.2.2 Partners requirements (License forms)

License forms have been submitted to partners in order to centralize their needs and define a licensing schema applicable to the whole consortium for the Results of the Aspire project.

See below (in Appendix A) the synthesis of the requirements expressed by 6 of the Aspire project’s partners.

5.2.3 Specific questions from Aspire partners

- **GNU General Public license v2 and V3**

Members have asked what the typical differences between those licenses are.

- **GNU Lesser General Public License v 2.1**

Members have expressed their preference to choose this license and argue that LGPL v2.1 would fit in with their own strategy for business or research purpose.

See below (in section 5.3) an Overview of the LGPL v 2.1, GPL V2 & V3.

- **Dual Licensing**

Some members would be interested in dual licensing schema for dissemination of Aspire project results.

5.3 Overview of the licenses : LGPL v 2.1, GPL v2, GPL v3

5.3.1 The GNU Lesser General Public License v2.1 (LGPL v2.1 for short)

The GNU Lesser General Public License v 2.1 is a variation of the regular GNU General Public License (GPL). Originally known as the GNU Library General License, it was drafted by Free Software Foundation to provide a weaker (or Lesser) form of copyleft for use in certain specific circumstances.

A brief background of the LGPL

In computing terminology, the word library can be used to describe a grouping of software functions for use by other programs. In this way the program code to undertake common tasks can be placed in the library, and programmers who wish their programs to perform these tasks can take advantage of the library's code in order to avoid the redundant work of writing their own version. The library's functions can either be copied into the program when it is compiled, or alternatively the program can access the library, if necessary, when it is being executed. Having your program use code from someone else's library requires that you have a licence from the library's owner to do so - after all, your program is incomplete without the library's functions, and will only function correctly with the addition of those functions. If the library were licensed under the terms of the GNU General Public License, your program would become a work derived from the library when it makes use of the library and thus the requirement would be that you release your program under GPL.

This fact means that anyone who writes a program that uses a GPLed library must either never distribute the program, or agree to license and distribute their program under the GPL as well. As a result, no closed source program can ever be distributed with a GPLed library that it uses. This is, of course, a desired effect of the GPL.

Nevertheless, sometimes a developer of a library might want to ensure that the library itself remains free while permitting non-free software to make use of it. This might happen if the author is trying to create a standard implementation of a particular software solution, and wants the resulting library to be used as widely as possible, while still being protected from relicensing and closing of its source. It is for these purposes that the GNU Lesser General Public License v2.1 was created

Main features of the LGPL v2.1

The LGPL v2.1 is identical to the GPL in many of its provisions. Nevertheless, there are some points in which the LGPL v2.1 differs from the GPL :

- Where the GPL mandates that all derivative works be distributed under the GPL, if at all, the LGPL v2.1 defines a separate class of works which may be derivative but which nevertheless can be licensed in any way. These are referred to as *works that use the library*. These are, essentially, programs that have been written to take advantage of the LGPL-licensed library but contain little or no actual code from the library in their uncompiled form. Such works may be distributed with the LGPL-licensed library and need not themselves to be distributed under the LGPL.

The exact extent to which the programs in question may contain code from the library is not precisely defined by the licence, although some guidelines are given.

- The LGPL v2.1 also differs from the GPL in placing restrictions on the variety and nature of derivative works that it allows. Licensees may modify an LGPL-licensed library, but if they wish to distribute their modified version it must also be a library. Modifications to LGPL-licensed libraries should not impair the library's ability to work with a wide range of programs.

  Provided that a work that uses the library meets these conditions, it can be distributed with the LGPL-licensed library in a number of ways. The aim of the licence here is to preserve the ability of recipients to modify the LGPL-licensed library and still have it work with the (possibly closed source) work that uses it. Any distribution must include the source code to the library, and prominent statements of the ownership of the library.
  It must also either

    - include the source code of the *work that uses the library*
    - include a facility which permits the *work that uses the library* to work with modified versions of the library, provided of course that the modified library retains its interface

  The second option there is most easily accomplished by having the *work that uses the library* dynamically access library functions when it is executed, rather than have it copy the library functions into its own code at compile time, and the LGPL v2.1 explicitly suggests this as a way of fulfilling its requirements.

- Finally, the LGPL v2.1 permits a programmer to distribute a hybrid library, which contains functions from the LGPL-licensed library and other functions which are not LGPL-licensed. However, a copy of the library with no LGPL-licensed code inserted must also be provided, and a notice of where the uncombined LGPL-licensed library may be obtained.

What does the LGPL v2.1 do ?

The points below are intended to summarise what is distinct about the LGPL v2.1. They are not intended as a full description of its features. The GNU Lesser General Public License v2.1

- keeps modified versions of the library itself open source
- allows non-open source software to use the library, and be distributed with it

5.3.2 GNU General Public license v 2 & v3

The GNU General Public License v2 - (GPL v2 for short)

The GPL v2 is the most commonly used open source licence. Approximately 70% of the projects in the software repository Sourceforge use the GPL v2. This document attempts to draw together the main features of this License into a friendly and comprehensible digest and, in addition, to note some details about its history and usage.

a. History

Although the basic principles of free software were established early on in the GNU project, it was not until 1989 that they were distilled into a licence that could be easily taken up and applied to any piece of software by its owner. Up to that point licences for GNU software had been written on an ad-hoc basis for each software release, and were often peppered with direct references to the software they licensed. This made them troublesome to reuse. The GNU General Public License version 1 solved this problem by simply referring to the *Program*. In 1991 the GPL v2 was revised to version 2, although the changes made were entirely in phraseology rather than legal effect. Also in 1991, the Library (or lesser) General Public License (LGPL) v2.1 was released by the FSF, to deal with special cases in which it might be desirable for free software to interact closely with software released with a licence that is incompatible with the GPL v2, such as a proprietary licence.

b. Main features of the The GPL v2

Like nearly any licence, grants rights under certain provisos. These are briefly listed here. A licensee of GPL v2-licensed software can:

- copy and distribute the program's unmodified source code (Section 1)
- modify the program's source code and distribute the modified source (Section 2)
- distribute compiled versions of the program, both modified and unmodified (Section 3) provided that:
    - all distributed copies (modified or not) carry a copyright notice and exclusion of warranty (Section 1 and 2)
    - all modified copies are distributed under the GPL v2 (Section 2)
    - all compiled versions of the program are accompanied by the relevant source code, or a viable offer to make the relevant source code available (Section 3)

The licence insists that the program itself and all programs based on it must be made available under the GPL v2 if they are made available at all. The source to the program, and all modified versions, must also be made available; if not, the granted right to modify is impossible to exercise.

Every new recipient of a GPL v2-licensed piece of software receives their licence from the original licensor (or licensors, if the work has been modified by one or more people). There is no sub-licensing of the rights granted from one recipient to another - anyone who cares to make use of the licensed program can get their own licence from the owners. This is stated explicitly in Section 6.

It follows from this, and the grants quoted above, that no-one can place additional restrictions on a GPL v2-licensed piece of software. If you choose to pass on the software to a third party, they will get the same licence that you did. If you have modified the software, you have already agreed to release the changes you make under the GPL v2, if they are released at all. As a result of this fact, it can be difficult to combine code that you receive under the GPL v2 with code that you receive under another licence. If the other licence contains any restrictions that are not present in the GPL v2, then the combination cannot be legally distributed. This licensing issue means that only a small set of very permissive licences are actually compatible with the GPL v2.

Section 7 of the GPL v2 explicitly spells out another consequence of the licence's grants and conditions. If a court rules that someone distributing GPL v2-licensed software must do so with an additional restriction - for example a charge for use of a patent belonging to someone else - then this means that the distributor must stop distributing the GPL v2-licensed software entirely.

c. What does the GPL v2 do ?

The points below are intended to summarise what is distinct about the GPL v2. They are not intended as a full description of its features.

- it ensures that modified versions of the code it covers remain free and open source
- it attempts to spread copyleft by mandating the use of the GPL v2 for distributed adaptations of GPL v2-licensed code

What's new with the GNU General Public License v3 - (GPL v3 for short)

On 28 June 2007, the Free Software Foundation (FSF) finally published the third version of their GNU General Public License (GPL). Over the previous eighteen months the Foundation had engaged in an unprecedented public consultation exercise, publishing four discussion drafts on the web and gathering opinions via a specially written web application that allowed anyone to flag sections of the drafts with their comments or concerns. In addition to this, the FSF formed four discussion committees designed to represent the wide range of interested parties from enterprise to private users.

This part of the document attempts to describe the major elements of the GPL v3, how it differs from its predecessor and some of the reasons for these changes.

a. What's wrong with GPL v2 ?

Over sixteen years separate the GPL v3 and its predecessor. In terms of information technology and software development, they have been extremely eventful years. Despite having been created for a world in which the internet was an academic curiosity and the phrase 'open source' was a reference to journalistic practice, the GPL v2 has generally coped well with the changing world of IT. It is by far the most commonly applied free and open source software licence, and probably the best known.

Nevertheless, by late 2005 Richard Stallman and Eben Moglen (respectively the founder and the lawyer of the Free Software Foundation) had decided that an update was necessary. Stallman had conceived the GPL as a legal tool for protecting what he termed the 'Four Freedoms' (perhaps in reference to President Roosevelt's famous 1941 State of the Union address). The FSF's Four Freedoms relate exclusively to software usage, and comprise the freedom to run, study, redistribute and improve the software. Although the GPL v2 had spectacularly succeeded in fostering a huge corpus of software that was usable under these freedoms, Stallman and Moglen saw some technological and legal developments that had occurred over the intervening years as potentially very harmful to software freedom. A new licence could perhaps build on the enormous success of the GPL v2 while combating these new perceived threats.

So what were these threats? Broadly speaking they can be summed up as follows

- 'Tivoisation' and Technological Protection Methods

- unintentional incompatibility with some open source licences
- US-specific legal terminology
- the rise of the web application as a means of realising value from software
- (emerging long after drafting and consultation had begun) software patent non-enforcement covenants as a means of dividing the free and open source software community

In the next sections we will describe these issues in more detail, and discuss the approaches taken by the GPL v3 to resolving them.

“Tivoisation” and Technological Protection Methods

Named after the popular digital video recorder marketed by TiVo Inc, 'tivoisation' refers to the distribution of free software in a device which cannot execute modified versions. The TiVo video recorder uses software distributed under the GPL v2 to perform some of its functions, and TiVo Inc. abide by the conditions of the GPL v2 by making the source code to this software available via their website. However, crytpographic signing is required to make the TiVo unit execute software, and this makes it effectively impossible for those who want to adapt the software and reinstall it on their TiVo unit to do so. TiVo are by no means the only company to do this, although they are probably the most successful, hence the coinage. To prevent software licensed under the GPL v3 being subject to the same kind of restriction, the FSF drafted a fairly complex addition to the terms which govern distribution of code in a non-source form. In early drafts of the GPL v3, the FSF simply added a requirement that - if a cryptographic key or some other vital 'Installation Information' were needed to modify and run the code on a device with which it was distributed - then the supplier had to hand that information over along with the source code. As a result of the public consultation, however, the FSF came to accept that it was not necessarily desirable for all categories of GPL-software-bearing devices to be user-modifiable in this way. Cardiac pacemakers were given as an example of something that might prevent user modification for perfectly good reasons of safety. As a result, the category of devices that must be accompanied by a usable signing key was narrowed to 'User Products', essentially meaning devices whose primary application is not industrial.

Another concern raised by the FSF related to the legal controls on circumvention of 'effective Technological Protection Methods' introduced in many world-wide jurisdictions through the adoption of the 1996 WIPO (World Intellectual Property Organization) Copyright Treaty. The treaty and its resulting embodiment in national legislation created a new way of violating someone's copyright: cracking their copy protection. It is worth noting that you do not actually have to copy or distribute a protected work in order to fall foul of this new provision; just removing a form of protection that previously functioned is enough. The FSF decided that they were against legislation that prevented what they saw as legitimate tecnological research. As a result, the GPL v3 contains a declaration by the licensor that no code distributed under it can be considered an 'effective Technological Protection Method', and thus that no licensor can act against someone modifying their code under the WIPO-based legislation.

Unintended incompatibilities

The fact that free and open source software is so readily available to all sometimes leads people to make incorrect assumptions about what they can do with it. It seems natural to assume that if one can use and distribute it freely, one must also be able to combine it freely. Unfortunately this is not the case. The GPL v2 stipulates that code which it covers must be distributed (if it is distributed at all) under the GPL v2, with absolutely no additional restrictions. This also applies to any modified versions of the code, which would include software products made from a combination of GPL'd code and some code obtained under a different licence. The upshot of this stipulation is that it is only possible to combine GPL'd code with other code obtained under a small set of other open source licences. To be part of the set, a licence must *only* contain restrictions that are present in the GPL. In other words, the GPL is only compatible with licences whose restrictions are subsets of those in the GPL. (It should be noted that these problems only arise when combining other people's code - the restrictions do not apply to taking GPL'd code and making new adaptations oneself).

Now many open source licences contain restrictions that the GPL does not. In some cases this is exactly what the licence's authors intended. In others, though, the authors did not want code covered by their licence to be eternally separate from GPL'd code. Both the Apache License 2 and the Eclipse Public License fell into this category. In both cases, the unintentional incompatibilities arose due to clauses relating to patents. So-called 'patent retaliation clauses' in these licences dictated that - if a licensee started patent litigation against any of the licensors - then the licence would be automatically withdrawn. The FSF were not against these kind of clauses in principle, although Eben Moglen had publicly questioned their efficacy. So, with the GPL v3, additional restrictions such as patent retaliation clauses became an optional extra for the GPL itself. If one wanted to combine code from an Apache 2-licensed project or an Eclipse-licensed project, it would be necessary to add such a patent retaliation clause to the GPL v3 that covered the eventual release. In that way the distributor could satisfy their responsibilities to the Apache licensor and the GPL licensor, and distribute without violating either licence.

US-specific legal terminology

The success of the GPL v2 outside its birthplace in the US meant that its roots in American law became increasingly problematic. To alleviate this, the GPL v3 was drafted using terminology that does not spring from any specific legal tradition. As a result, far more space in the GPL v3 is devoted to definitions of terms, and this has attracted some criticism from lawyers. After all, with greater complexity comes a greater potential for miscommunication and using terminology that no lawyers are familiar with could be seen as an invitation to an argument. It remains to be seen if the great effort spent in divorcing the GPL v3 from its origins in American law will lead to greater or lesser clarity.

In addition to the definition of simple terms, some larger sections of the GPL v2 were tailored to be effective in the US but not necessarily anywhere else. For example, the GPL v2's Limitation of Liability section was drafted in a way that was - arguably - entirely ineffective in the UK, due to its attempting to entirely exclude liabilities that can't be excluded here. Thus the desired effect, which was to protect the licensor, was entirely reversed. In reaction to this problem the GPL v3 permits licensors to redraft these sections of the licence to better suit conditions under local law.

The rise of the web application as a means of realising value from software

With the rise of the internet as a place to do business, more and more open source software is being adapted by companies to run large-scale web services for payment. Under the terms of the GPL v2, this does not count as distribution, and thus the companies in question are under no responsibility to publish the source code to their modifications. Some commentators saw this as a 'bug' in the GPL v2 - after all, surely the ethos of the free software community demanded that those who gain greatly from it also contribute back? Others were less convinced, and argued that the activities of web application providers were in accord with both the letter and the spirit of the GPL v2.

It was an issue that the FSF itself could not decide upon, so a sample restriction was included in the first draft of the GPL v3 for public discussion. The clause in question allowed a licensor to incorporate a specialised function into their code which returned the source code of the software if a user requested it over a network. If the licensor decided to include this kind of function, the draft restriction prevented any subsequent modifiers from removing it - indeed they were obliged to maintain it so that it always returned or 'spewed' the most up to date version of the source code.

In fact, a San Francisco-based company called Affero (who provide a system for rating and assessing online volunteers) had already drafted a variation on the GPL v2 back in 2002, specifically to include a 'code spew' clause. This proved to a neat way of handling the dilemma; those who wanted the restriction could use the Affero version of the GPL on their code, and the final version of the GPL v3 would be adapted to make code it covers combinable with code released under the Affero GPL. The FSF is now in the process of helping Affero create v2 of their licence.

Software Patent Wars

The FSF's original plan of having the entire GPL v3 composed and agreed upon in twelve months was always extremely ambitious. On 2 November 2006, Novell and Microsoft announced a complex reciprocal deal that many saw as a deliberate attempt to frustrate the aims of the FSF while remaining technically in accordance with the GPL v2. As the free and open source software community began to form a strongly negative view of the deal, it became inevitable that a lot more work would be needed to make the GPL v3 a suitable vehicle for discouraging such deals in the future.

Microsoft and Novell's collaboration agreement is broad and complex, and its specifics are not yet available publicly in their entirety. The main irritant for the free and open source software community was the mutual agreement between Novell and Microsoft that they would not use their patent portfolios to litigate against each other's customers. This had two important implications for free and open source software. Firstly, it gave strength to Microsoft's oft-stated but never demonstrated assertion that Linux violates Microsoft patents. After all, why would Novell make the deal if they did not believe that their SUSE Linux users were at risk? Secondly, it created a division within the Linux user community. On the one side were those covered by Microsoft's promise to Novell, and on the other was everyone else. If the belief that protection from Microsoft's patent litigation was a necessary component of any Linux distribution became widespread, it would create a severe limitation to the free use and adaptation of Linux - effectively driving users to install SUSE or face the wrath of Microsoft. Of course, if that view did become widespread it would be very likely to benefit Novell financially.

The fact that the patent 'promise' was directed at the companies' customers rather than the companies themselves was widely interpreted to be a deliberate evasion of the terms of the GPL v2. Had Microsoft provided an indemnification to Novell itself, Novell would have been prevented from distributing Linux, as their acceptance of what is essentially an additional requirement imposed by Microsoft (to not sue Microsoft's customers) would have violated the terms of GPL v2 section 7.

To combat what was seen as a new method of suppressing software freedom, the GPL v3 includes two new stipulations. Firstly it dictates that anyone who distributes GPL v3-licensed code and provides a patent licence to some group of recipients must automatically extend that licence to all recipients. This would have an effect on the Novell-Microsoft deal as it includes an agreement to distribute and support each other's operating systems. Secondly GPL v3 draft 3 includes a stipulation that you cannot distribute covered code if you enter into a deal with another software distributor that involves your paying that software distributor to not sue your customers (in this case the 'payment' would be in the form of an undertaking not to litigate). Some doubt remains over the workability of these provisions, and whether they may trap beneficial patent-cross-licensing deals.

As a conclusion, Opinion continues to be strongly divided on the new GPL. Linus Torvalds, originator of Linux and chief maintainer of the Linux kernel, has stated his dissatisfaction with it, and intends to keep on using the GPL v2. On the other hand Jeremy Allison the chief developer of Samba (the widely used free software that provides networking compatibility between Linux and Microsoft's Windows platform) has announced that he fully approves of the new GPL and will release all future versions of Samba under v3 only. Success or failure for the GPL v3 will be judged on both its resilience as a legal document and the number of software authors who choose it to protect their code, whatever its virtues as a legal document.

5.4 Dual licensing

Simply, dual licensing describes a situation where the same piece of software can be obtained under two different software licences. Usually one of these licences is an OSI approved open source licence and the other is a proprietary licence.

The licence fee is generally only one way that a software vendor makes its money; they often also offer additional chargeable services such as support and consultancy. These supporting services are often available from third parties. For example, an institution may purchase some proprietary software by paying a licence fee to that software's vendor but may then negotiate a support contract with a third party that may or may not be affiliated in some way with the vendor.

5.4.1 Dual licensing as a business model

Supporting services are not the only way that an open source business can make money, they can make money from licensing. Clearly, if software is released under an open source licence it is not practicable to charge for the software as your neighbour can give it away for free. However, an open source software vendor may choose to dual license its software. This means that its software is made available both under an open source licence and under a different licensing scheme that may incur a licence fee. But why would anyone choose the chargeable licence? There are many reasons why this might happen and the most common by far is that the open source software is to be re-used within a proprietary software product.

5.4.2 An example

A company, Databases-r-us, is developing a database application aimed at the first time database user. They wish to develop and sell an application that consists of a database back end with a suite of easy to use tools on top that make designing, maintaining, and using a database a trivial task. They would like to use the popular MySQL database, an open source application developed and distributed by the company of the same name and released under the GNU General Public License (GPL). The terms of this licence are such that if Databases-r-us develops and releases software that contains the MySQL database code then that MySQL-based application must be redistributed in a way that the complete source code for the application is open and available for redistribution. In practical terms this means that the resulting application must also be released under the GPL. In this situation Databases-r-us do not want to do this because they feel that the software code they have developed is part of their business advantage. However, they are very keen on the MySQL database and still want to bundle it with their software. Fortunately for Databases-r-us, as the MySQL database is a dual licensed product this is indeed possible.

Under MySQL's dual licensing business model, users may choose to use MySQL products under the open source GPL or under a commercial licence. Anyone who is developing and distributing open source applications under the GPL is free to use MySQL under the GPL. Additionally anyone who is developing and distributing open source applications under an OSI approved licence that is **not** the GPL, may use MySQL under the GPL with a FLOSS exception.

For anyone who wants to develop and distribute but does not want to release the source code for their application MySQL is able to provide a commercial licence. Because MySQL has full ownership of the MySQL code it is able to tailor its commercial licensing terms to meet the unique requirements of users interested in embedding or bundling MySQL.

5.4.3 Can dual licensing benefit the open source world?

On the one hand, in such a case, the open source world benefits from the dual licensing model as the code developed is available to anyone who wishes to use it.

However, it should be noted that dual licensing may have a negative effect on community contributions to open source projects. That is, by allowing some people to keep modifications private whilst others are forced to make their changes public, the community built around your software code is likely to consist of many more users than developers.

5.4.4 Dual licensing for licence compatibility

Another reason for using a dual licensing model is to circumvent some of the incompatibilities between OSI-certified licences. For example, the Mozilla Foundation uses a tri-licensing model to license certain software under the Mozilla Public License (MPL), the General Public License, and the Lesser General Public License (LGPL) in an effort to address the issue of incompatibility with other open source licences.

5.4.5 Example of other dual licensed products

MySQL is not the only open source company providing dual licensed products. Other examples include:

- Qt, a cross platform toolkit used to develop GUIs, from Trolltech
- Berkeley DB, a database system, from Sleepycat Software
- Asterisk, an open source telecommunications software suite, from Digium
- LAMS, a learning activity management syste, from Macquarie University

amongst others.

5.5 Recommendations

The present document gives a proposal but don't take into account specific elements of the partners participating in the project.

Beyond the recommendations, each member shall validate with its own representatives and legal department the approach of the licensing schema which would be , in order to make a final choice in accordance with its legal status (universities research institute, SME's, non profit organization, etc.) and other internal rules, strategy for business or research and development, commitment with third part etc.

5.5.1 Licensing for software

Main elements of various requirements and members needs

As far as partners who have filled in the form submitted are concerned, the main points resulting from the analysis of the forms are the following:

a. concerning the activities of the partners involved in the project

There are two kinds of "participant" in the project: a part of them are engaged in development activities, others are not.

b. concerning reuse of software,

Some partners may use many 3rd part OS component available under various OS licenses and/or some proprietary software.

c. concerning future development (beyond the project life time)

Some partners would like to be authorized to do future development under both option of closed/open source (and may combine some pieces of code under OS licenses).

d. concerning future dissemination and patent policy
Some partners will probably continue the development and support of the platform/application as open source, as well as provided some commercial extension and get into commercial relationship.

Suggestions

How to decide whether to use GPL or LGPL (and more generally the best license for a project) is determined by the project's strategy, and depends on the details of the situation. Therefore the recommendations made below, and based under the requirements expressed by the partners, are given at a generic level since each member has its own specificity. (It would probably involve case by case studies)

According to the relevant elements mentioned in that document, there are several options:

a. Simple license

As far as the code concerned is a library, the LGPL v2.1 appears to fit in well with the various requirements expressed, as it

- keeps modified versions of the library itself open source
- allows non-open source software to use the library, and be distributed with it

This might happen if the aim is trying to create a standard implementation of a particular software solution, and wants the resulting library to be used as widely as possible, while still being protected from relicensing and closing of its source.

The Apache v2 and the BSD licenses might find application here, recognised by the Open source Initiative as some popular and widely used licenses with a strong community might find an application by the Aspire consortium.

Although the Apache License v2 is icompatible with the GPL v2, today it is not the case with the release of the GPL v3.

Both licenses, the BSD and the Apache v2, permits code that it covers to be mixed with closed source projects. (See in Annex 1 an overview of the Apache License v2 and the new BSD)

b. Dual licensing

As seen above, some projects try to fund themselves by using a dual licensing schema, in which proprietary derivative works may pay the copyright holder for the right to use the code, but the code still remains free for use by open source projects.

The attractiveness of dual licensing schema for the results of Aspire project is that :

- This tends to work with code libraries and standalone applications (but the exact terms differ from case to case). Often the license for the free side is the GNU GPL.
- At its best, it provides a way for a free software project to get a reliable income stream.

But, on the other side, it can also interfere with the normal dynamics of open source projects. The problem is that any volunteer who makes a code contribution would contribute to two distinct entities: the free version of the code and the proprietary version.

Moreover, the inconvenient might be exacerbated by the fact that in dual licensing, the copyright owner really needs to gather formal, signed copyright assignments from all contributors, in order to protect itself from a disgruntled contributor later claiming a percentage of royalties from the proprietary stream. The process of collecting these assignment papers means that contributors are confronted with the fact that they are doing work that makes money for someone else.

Restrictions

The choice for the licensing schema should take into account legal aspects provided by :

- the contracts in which members are already involved in, and licenses granted by third part, (that means to have potential commitments towards potential third part rights).
- the consortium agreement provisions

5.5.2 License for documentation

Requirements

The dissemination level of the project documents has been established in the Description of Work and in previous agreements.

The OSS documentation may be public.

If a further modification of these rules is required then all the partners in the consortium should agree on these matters.

Suggestion

As described above, and according to the agreement by all partners in Brussels, as well as the specific requirements of members, the recommendation given here is to enforce a Creative Commons license :

CC Attribution + Noncommercial + ShareAlike (by-nc-sa)

This license lets others remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms. Others can download and redistribute your work just like the by-nc-nd license, but they can also translate, make remixes, and produce new stories based on your work. All new work based on yours will carry the same license, so any derivatives will also be non-commercial in nature.

The reference to this license should be as follows :

"This work is licensed under the Creative Commons Attribution-Non Commercial- ShareAlike 3.0 Licence.

To view a copy of this licence, visit http://creativecommons.org/licenses/by-sa/3.0/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA."

Section 6 References and bibliography

1 United Nations Conference on Trade and Development, E-commerce and Development Report 2003, « Free and Open Source Software : Implications for ICT Policy and development »

2 « open source software : Why is it here and will it stick around ? » K. Nikulaien (2004), See also GNU Manifesto, Richard Stallman

3 « How to evaluate Open Source Software/Free Software Programs » (2005) David Wheeler

4 See Open Source Initiative, Definition version 1.9 (2006.07.07) available at http://www.opensource.org/docs/definition.php

5 See http://www.cecill.info/licences.en.html

6 See http://www.infoworld.com/article/04/07/09/HNfrancelendssupport_1.html (by Peter Sayer -IDG News Service)

7 http://www.gnu.org/copyleft/copyleft.html

8 http://www.opensource.org/licenses/gpl-license.php

9 MIT licence available at http://www.opensource.org/licenses/mit-license.php and New BSD license, available at http://www.opensource.org/licenses/bsd-license.php – Last visited july 2007

10 Digital Millennium Copyrigth Act (DMCA)

http://www.loc.gov/copyright/legislation/dmca.pdf

ORGANIZATIONS

http://www.wipo.int/sme/en/documents/opensource_software_primer.htm

http://www.epip.eu/conferences/epip02/

http://www.opensource.org/

OS DEFINITION

http://www.google.com/search?hl=en&lr=&oi=defmore&q=define:Open+source+software

http://www.opensource.org/docs/osd

http://www.fsf.org/licensing/essays/free-sw.html

COMPLIANCE

http://www.fsf.org/licensing/compliance.html

Jason D. Haislmaier – 2006 « Closing the Open Source Compliance Gap (Best practices for developping Open Source Compliance Gap) »

HISTORY

http://www.gnu.org/gnu/gnu-history.html

«Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free. The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program and distribute modified versions — but not permission to add restrictions of their own. Thus, the crucial freedoms that define "free software". . . become inalienable rights.», Stallman,*"The GNU Operating System and the Free Software Movement"*; in DiBona et al pp. 53 — 70, plus précisément p. 59.

LEGAL

Brendan Scott – 2006

« The Open Source Legal Landscape »

Jonathon J. frost, University of Washington – 2005

« Some Economic and Legal Aspects of Open Source Software »

Linda Hamel, General Counsel Information Technology Division 2004 (Massachussetts)

« Open Source Software, Legal and other Issues related to use Open Software Products »

Thierry Lafond, Tiia Tiainen - 2003

« la problématique juridique du logiciel libre »

Yannick Bailly (note de recherche CEIPI)

« La protection juridique des logiciels libres »

JB Soufron 2002 « La GPL : un système original de protection juridique pour les créations issues des systèmes de développement coopératifs »

Compatibility of various licence

Benjamin Jean – 2006

« option libre : compatibilité entre contrats » (Mémoire de recherche – Master Droit de créations immatérielles))

http://www.foo.be/indutec/

http://www.venividilibri.org/Licences/Comparatif

http://www.gnu.org/gnu/gnu-history.html

«Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free. The central idea of copyleft is that we give everyone permission to run the program, copy the program, modify the program and distribute modified versions — but not permission to add restrictions of their own. Thus, the crucial freedoms that define "free software". . . become inalienable rights.», Stallman,*"The GNU Operating System and the Free Software Movement"*; in DiBona et al pp. 53 — 70, plus précisément p. 59.

www.groklaw.net/article.php?story=20031231092027900

http://www.ipr-helpdesk.org/index.html

http://www.denniskennedy.com/resources/technology-law-central/opensourcelaw.aspx/

http://www.oss-watch.ac.uk/resources/gpl3final.xml

Appendix A Summary of the requirements

A.1 Software

Partners	Software code reuse for aspire developments	Future development (beyond project life time)	Future dissemination and patent policy	Question
A	May use many 3rd part OS component available under various OS licenses	Would like both option of closed/open source, may combine some pieces of code with Google toolkit, Jboss application server, Jonas application server	- Will probably continue development and support of the platform/application as closed source - May offer adds value consulting services using the aspire SW/ MW - Will customize the aspire MW for business with future clients	
B	Will not be engage into development activities	- Only use OS - may combine code with other OSS under Apache –type or LGPL License	- surely continue the development and support of the platform/application as open source. - commercialise a certification programme with privacy seals and auditing. - does not foresee any patent needs.	
C	- is not involved in the development of ASPIRE middleware - ASPIRE Middleware must be free, open source, and have to be improved by the whole community, without being possible to modify it and keep sources	- future development under opened or closed manner	- do not see restriction as far as the ASPIRE middleware is “open source” and allows the use of proprietary developments for low level and high level functionalities. No possibility for patents or closed developments for the ASPIRE middleware.	

D	confidential. Nevertheless, low level and high level functionalities have to be combined with ASPIRE even if they are patented, not free, closed, etc…			
D	- May use both commercial and proprietary SW tools - Flexible with respect to enter different licensing schemes	- Main activities carried out are for research purpose - Possibility to get into commercial relationships	-Commercial extension from the research project is not excluded	
E	- Both proprietary and free OSS component will be used - rules applicable are project by project	- Agree with the rules defined for aspire project for results - Re use of inventions and software for research purposes	- possibility for future development	
F				- Do only HW not SW - The consortium agreed that the rules defined for licensing are not applicable to Melexis

A.2 Documentation

Partners	Documentation license policy	Creative Commons	choice
A	-No dissemination of the documentation expect some selected part - Specific Authorization is needed	Yes	CC Attribution -Non commercial- Share Alike
B	- The documentation will be distributed with the source code. - Anyone, as long as it is distributed alongside the correspondent version of the source code. -Yes, as long as modifications are fed back into the original source. - They belong to OSI and the ASPIRE Consortium.	Yes	As agreed in Brussels by all partners
C	- The documentation will be distributed with the source code. - Anyone, as long as it is distributed alongside the correspondent version of the source code. - Yes, as long as modifications are fed back into the original source. - They belong to Traceability Centre and the ASPIRE Consortium.		As agreed in Brussels by all partner
D	Documentation may be public The responsible for dissemination have been established in the DOW & previous agreement	Yes if agreed by all partners	As agreed in Brussels by all partner

E	- Documentation will be public - Anyone from the consortium has the opportunity to distribute the documents, with the approval of the party who has made the document and project management - Authorization is needed for modification - Document belong to the consortium, the author will be mentioned in any re-publications.		As agreed in Brussels

Appendix B Overview of the Apache License v2 and the new BSD License

B.1 The Apache License v2

The Apache License is recognised by the Open Source Initiative as a popular and widely used licence with a strong community. It is used by only about two percent of the open source-licensed projects on the software repository Sourceforge. It is an interesting licence to compare with the Berkeley Software Distribution (BSD) licence, which it resembles in some ways.

B.1.1 History of the Apache License

Beginning in 1995, the Apache Group (later the Apache Software Foundation) released successive versions of their well-known httpd server. Their initial licence was essentially the same as the old BSD licence, with only the names of the organisations changed. When Berkeley accepted the argument put to it by the Free Software Foundation and retired their *advertising clause* from the BSD licence, Apache did likewise and created the Apache License v1.1 - a slight variation on the *new BSD licence*. In 2004 Apache decided to depart from the BSD model a little more radically, and produced the Apache License v2.

B.1.2 Main Features of the Apache License v2

Like all licences the Apache License v2 grants certain rights under certain conditions. In brief a licensee of Apache Licensed V2 software can:

- copy, modify and distribute the covered software in source and/or binary forms
- exercise patent rights that would normally only extend to the licensor

provided that:

- all copies, modified or unmodified, are accompanied by a copy of the licence
- all modifications are clearly marked as being the work of the modifier
- all notices of copyright, trademark and patent rights are reproduced accurately in distributed copies
- the licensee does not start legal action against the licensor(s) over patent infringement
- the licensee does not use any trademarks that belong to the licensor

This rewriting of the BSD licence tries to achieve a few things. Firstly it adds an explicit grant of patent rights where that is needed to operate the software. Some argue that such a grant is implicit in other open source licences, but the Apache License v2 spells it out. It also contains solid definitions of the concepts it uses, providing more certainty as to its intended meaning. Among these is a definition of *Contributor* that contains another interesting feature of the licence. A *Contributor*, as distinct from someone who just modifies the software, also grants a licence to their modification back to the original authors. This mechanism, if taken up, simplifies control of the code. Finally the v2 licence is usable by other projects without the need to replace wording in the licence document itself.

B.1.3 Other Features of the Apache License v2

One unintended upshot of the creation of the Apache License v2 is that it became incompatible with the GPL v2. Previous versions, being heavily based on the BSD licence, were compatible. However the restriction in v2 that terminates the grant of rights if the licensee sues over patent infringement is seen by the Free Software Foundation as a restriction that is not present in the GPL v2. This being the case, code that it licensed under the Apache License v2 cannot be combined with GPL v2-licensed code and distributed. The Apache Software Foundation itself argues that the GPL v2's section 7 (which terminates the right to distribute if an external body places additional restrictions on the distributor) is similar enough to their patent clause to make them effectively the same restriction. Unfortunately, as the licences stand, the commonly accepted view remains that they are incompatible.

However, with the release of the GPL v3, this incompatibility is no longer insurmountable. The GPL v3 allows for the addition of a patent retaliation clause whose effect is to allow code from a GPL v3-licensed project to be combined with code from an Apache 2-licensed project. So, a major milestone has been acheived and the two licences are no longer incompatible.

Like the BSD licence, the Apache License v2 permits code that it covers to be subsumed into closed source projects.

B.1.4 What Does The Apache License v2 Do?

The points below are intended to summarise what is distinct about the Apache License v2. The Apache License v2

- explicitly grants patent rights where necessary to operate the software
- permits code that it covers to be subsumed into closed source projects

B.2 The New BSD License

The Berkeley Software Distribution License (BSD for short) is one of the most commonly used open source licences. Approximately seven percent of the open source-licensed projects on the software repository Sourceforge use some form of BSD licence. Although this may sound like a small proportion, it does in fact make it the third most popular open source licence (the GPL and LGPL are ahead of it and account for almost eighty percent of open source licensing).

B.2.1 History of the BSD License

The University of California at Berkeley has a long history of pioneering software development and software distribution models. Having existed in some form since the early 1980s, the BSD licence can claim to be the oldest of the open source licences. In fact its long life has resulted in there being more than one version, and it is slightly misleading to speak of *the* BSD licence as a result. Although the history of its evolution is an interesting one, for the purposes of this document we will confine ourselves to detailing the last major revision that resulted in what is today called the *modified BSD licence* or the *new BSD licence*.

Until the late 1990s, many instances of the BSD licence included the following clause:

All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by (developer).

As reasonable as this might seem, it threatened to make the practice of aggregating open source software extremely impractical. Someone who wanted to publish such a collection might include hundreds of pieces of software, all with an adapted version of that clause. The obvious result would be that any promotional materials would have to include line after line of acknowledgements, leaving almost no room for logos, images or details. As it became clear that this was becoming a real problem, the Free Software Foundation lobbied Berkeley's legal department to reissue the licence without the *advertising clause*. This they did, creating the *new BSD licence*. Of course, there are still hundreds of pieces of software out there licensed under the old version, and effort continues to contact the authors and persuade them to reissue their work under the revised licence.

B.2.2 Main Features of the BSD License

Even before the removal of the advertising clause, the Berkeley Software Distribution License was refreshingly short. It fits easily onto one side of a sheet of paper, and is relatively free of verbiage. A licensee of BSD-licensed software can:

- use, copy and distribute the unmodified source or binary forms of the licensed program
- use, copy and distribute modified source or binary forms of the licensed program

provided that:

- all distributed copies are accompanied by the licence
- the names of the previous contributors are not used to promote any modified versions without their written consent

B.2.3 Other Features of the BSD License

When comparing the BSD licence to other open source licences such as the GPL or the MPL, it is clear that it does not try to exercise anywhere near as much control over its licensees. In consequence, a licensee can take some code that is licensed under the BSD licence and incorporate it into their closed source work. A licensee can take BSD-licensed code and add to it, safe in the knowledge that whatever they contribute can be distributed in whatever way they choose. For this reason the licence is seen as friendly to traditional software business models that depend upon keeping the source private and capitalising on the sale of licensed binaries. Code that enters a traditional software business as BSD-licensed need not be distributed that way, and thus competitive advantage in the traditional sense can be maintained.

Another result of the BSD licence's simplicity and brevity is that code licensed under it can be distributed alongside code licensed under the GPL without problems. In general this kind of distribution is hampered by the fact that the GPL demands that no additional restrictions are placed on its licensees. In practice this means that no licence which features a restriction that is not in the GPL is compatible with the GPL. The new BSD licence's only restriction - that the original authors' names not be used in promotion without their permission - is present in the GPL.

B.2.4 What Does the BSD Do?

The points below are intended to summarise what is distinct about the BSD licence. The BSD licence

- allows code licensed under it to be incorporated in closed source software
- allows code licensed under it to be incorporated in GPL-licensed software