



Grant agreement no.: 224170

SHARE

Sharing Open Source Software Middleware to improve industry
competitiveness in the embedded systems domain

Project type: Support Actions (SA)

Start date of project: May 22nd, 2008

Duration: 24 months

D2.3 Licenses benchmarking results

Due date of deliverable: 03/31/2009

Actual submission date: 03/31/2009

Final revision: Version 1

CRAI

Project co-funded by the European Commission within the Seventh Framework Programme (2007–2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision no.	Date of Issue	Author(s)	Brief Description of Change
0.1	11/06/2008	CRIAI	Table of contents.
0.2	12/19/2008	CRIAI	Merge of chapter 2.
0.2.1	01/12/2009	CRIAI	Merge criteria and methodology.
0.2.2	02/12/2009	CRIAI	Merge chapters 3 and 5.
0.3	02/17/2009	CRIAI	First draft (waiting for contributions).
0.4	03/02/2009	CRIAI	Apply corrections to criteria.
0.5	03/02/2009	CRIAI	Upload template and evaluations sheets to the O4S platform.
0.6	03/02/2009	CRIAI	First light lexical correction and first best practice merge.
0.6.1	03/13/2009	Critical Software	Grammar correction and hints.
0.6.2	03/25/2009	SESM	Correction and hints.
1.0	03/27/2009	CRIAI	Final Release.

Table of Contents

List of Acronyms	9
1 Introduction.....	11
1.1 Chapters overview.....	11
2 Brief history of Open Source and Free Software.....	13
2.1 Copyright and licenses.....	13
2.1.1 Brief history of Copyright.....	13
2.1.1.1 What is copyright granting to authors and publishers?.....	14
2.1.1.2 How to apply copyright to an original creation.....	14
2.1.1.3 Why copyrighting a work?.....	14
2.1.1.4 Licensing rights.....	14
2.2 Proprietary software and licenses.....	15
2.2.1 Proprietary software licenses.....	15
2.2.1.1 Microsoft Windows EULA	16
2.3 The Free Software Foundation.....	17
2.4 The Open Source Initiative.....	19
2.5 FLOSS licenses.....	20
2.5.1 Open source licenses.....	21
2.5.2 Free software licenses and copyleft.....	23
3 Emerging issues: software patents and DRM.....	29
3.1 What are software patents.....	29
3.1.1 Copyright and patents.....	31
3.1.2 Software patents and research.....	32
3.1.3 Software patents and the EU.....	33
3.2 Digital Rights Management.....	35
3.3 How licenses are addressing emerging issues.....	37
4 Licenses benchmarking methodology.....	39
4.1 About license benchmarking methodology.....	39
4.2 Key factors and criteria for FLOSS licenses.....	40
4.2.1 Generic criteria.....	41
4.2.1.1 Copy.....	41
4.2.1.2 Attribution clause.....	41
4.2.1.3 Additional restrictions.....	41
4.2.1.4 Retain Copyright.....	42
4.2.1.5 Redistribution under a different license (retain same license).....	42
4.2.1.6 Modifications.....	42
4.2.1.7 Source code access.....	43
4.2.1.8 Binaries distribution.....	43
4.2.1.9 Binary only distribution.....	43
4.2.1.10 Mixing with proprietary software.....	43
4.2.1.11 GPLv2 Compatibility.....	44

4.2.1.12 GPLv3 Compatibility.....	44
4.2.1.13 Copyleft License.....	44
4.2.1.14 Derivative works.....	46
4.2.1.15 Close original code.....	47
4.2.1.16 Close derived code.....	47
4.2.1.17 Restrictions on other software.....	47
4.2.1.18 Linking from code with different licenses.....	47
4.2.1.19 Fee Charging.....	48
4.2.1.20 License popularity.....	48
4.2.1.21 Patents awareness.....	48
4.2.1.22 Protection against patents.....	48
4.2.1.23 DRM awareness.....	48
4.2.1.24 Discriminations against groups or people.....	49
4.2.1.25 Discriminations against fields of endeavour.....	49
4.3 O4S.....	49
5 OS business models and licenses.....	53
5.1 Selling technical support services.....	53
5.2 Distributing Software.....	55
5.3 Funding with donations.....	57
5.4 Build and sell hardware and software configurations.....	58
5.5 Release proprietary versions of the software.....	60
5.6 Release proprietary add-ons, components.....	60
5.7 Dual licensing software.....	61
5.8 Offer each new version of a software only to paying customers.....	63
6 Licenses benchmarking results.....	64
6.1 Licenses identity cards.....	65
6.1.1 Academic Free License 3.0 (AFL 3.0).....	65
6.1.2 Adaptive public license.....	66
6.1.3 Affero GNU GPLv2.....	67
6.1.4 Apache License v2.0.....	68
6.1.5 Artistic License 2.0.....	69
6.1.6 BSD License.....	70
6.1.7 Common Public License.....	70
6.1.8 Eclipse Public License.....	71
6.1.9 GNU GPLv2.....	72
6.1.10 GNU LGPLv2.....	73
6.1.11 GNU GPLv3.....	73
6.1.12 GNU LGPLv3.....	74
6.1.13 Microsoft Public License.....	75
6.1.14 Microsoft Reciprocal License.....	75
6.1.15 MIT License.....	76
6.1.16 Mozilla Public License.....	77
6.1.17 Nokia Open Source License.....	77
6.1.18 Open Software License.....	78
6.1.19 Python Source Foundation License.....	79
6.1.20 QT Public License.....	79
6.1.21 SleepyCat License.....	80

6.1.22 Sun Public License.....	81
6.1.23 Sybase Open Watcom Public License.....	81
6.1.24 Vovida Software license.....	82
6.1.25 W3C License.....	83
6.1.26 WxWindows License.....	83
6.1.27 X.net License.....	84
6.1.28 Zlib License.....	85
6.2 Results.....	85

Illustration Index

Illustration 1: The Free Software Foundation logo.....	17
Illustration 2: The Open Source Initiative logo.....	19
Illustration 3: Open Source, Copyleft and Free Software families.....	20
Illustration 4: Copyleft logo: all rights "reversed"	26
Illustration 5: A classical DRM content distribution scheme.....	33
Illustration 6: Microsoft Silverlight DRM contents distribution.....	34
Illustration 7: The O4S set weighting	47
Illustration 8: Selection of a previously saved weightset on O4S.....	48
Illustration 9: Three popular licenses compared.....	49
Illustration 10: A view of some popular Open Source projects.....	54
Illustration 11: Dual licensing development model.....	59

LIST OF ACRONYMS

BSD	Berkeley Software Distribution
CSS	Content Scrambling System
DRM	Digital Rights Management
DVD	Digital Versatile Disc
DVI	Digital Visual Interface
EPC	European Patent Convention
EPO	European Patent Office
EULA	End-User License Agreement
FFII	Foundation for a Free Information Infrastructure
FLOSS	Free/Libre Open Source Software
FSF	Free Software Foundation
GATT	General Agreements of Tariffs and Trade
GNU	Gnu is Not Unix
GPL	General Public License
HDMI	High-Definition Multimedia Interface
IP	Intellectual Property
MPL	Mozilla Public License
OSD	Open Source Definition
OSI	Open Source Initiative
OSL	Open Software License

PCT	Patent Cooperation Treaty
QSOS	Qualification and Selection of Open Source
TRIPS	Trade-Related Aspects of Intellectual Property Rights
WIPO	World Intellectual Property Organisation
WTO	World Trade Organisation

1 INTRODUCTION

Choosing a license is a critical task for every public project. Choosing a free software or an open source license can be even more difficult. Recently Free/Libre Open Source Software (FLOSS) has been having an ever greater impact worldwide, becoming the favourite choice for many individuals and organisations. This popularity led to the phenomenon of license proliferation. The birth of many new licenses is due to two main factors:

- A scarce knowledge of already existing licenses
- The need for additional terms not covered by existing licenses

License proliferation is becoming a real issue when choosing a license under which release a new software.

In this document it will be proposed a complete overview of free and open source licenses and of best practices that should be followed to successfully accomplish software licensing.

The document will mainly deal with technical problems related to licenses. Legal issues are out of the scope of this document. Nevertheless, it is impossible to ignore some threats to FLOSS introduced and/or derived by software patents and Digital Rights Management (DRM) technologies, it is possible that these threats introduce ethical, social and political issues. But, once again, it will be proposed just an overview and an in depth discussion is left to other documents.

1.1 CHAPTERS OVERVIEW

Chapter 2 will offer a brief history of copyright and licenses, introducing the concept of proprietary EULA and a practical case study of the most popular EULA worldwide, the one from the Microsoft Windows operating system. In the second part of the chapter we will look at the origins of FLOSS, drawing a brief history of the Free Software Foundation and the Open Source Initiative and the respective definitions for free software and open source software.

Chapter 3 will deal with emerging issues like software patents and patentability of software, Digital Rights Management, copy and content protection technologies and how licenses are dealing these issues and evolving in order to protect FLOSS users and developers from these threats.

In chapter 4 the benchmarking methodology for license applicability will be briefly summarized: each criterion will be described and explained. Scoring to given criteria is briefly discussed too. The O4S application is then quickly presented as methodology applying is covered in deliverable D2.1 of the SHARE project.

Chapter 5 will deal with the most popular open source and free software business models. Success stories and each business model's applicable licenses will be shown.

Chapter 6 will introduce identity cards for each license and results of the benchmarking will be showed together with some best practices in licensing. Part of

the results will be shown via a web-application, the O4S tool, that will be showing comparative matrix and kiviatt diagrams of the evaluation.

Chapter 7 will contain conclusions of the document.

2 BRIEF HISTORY OF OPEN SOURCE AND FREE SOFTWARE

2.1 COPYRIGHT AND LICENSES

Understanding FLOSS licenses and licensing issues requires a more general knowledge of the concept of copyright and how licenses are related to it.

Licenses and copyright are different entities, one tightly tied to the other, that's why it is fundamental to understand what copyright is, how it was conceived, how it has been changing during years.

2.1.1 Brief history of Copyright

Copyright is a legal concept introduced at the beginning of the eighteenth century in Britain and then internationally spread with the Berne Convention¹, whose international agreements are still in place. This first formal act granted some general rights to authors of books, photographs, motion pictures. All the countries adhering to the convention must recognize the rights of authors from other signatories' countries and consider them as they were published by national authors.

The agreement fixes minimum terms of validity for each kind of work, but each country is free to provide its own terms at the end of which the rights of the author expire.

At the time of the Berne Convention, Information Technology was far away to come: in 1996 a new treaty, disposing additional protections to copyright, was signed. This is known as the World Intellectual Property Organization Copyright Treaty². The treaty establishes that computer software is protected in the same way literary works are. Even more, the treaty states *"compilations of data or other material, in any form, which by reason of the selection or arrangement of their contents constitute intellectual creations, are protected as such"*. These were two fundamental introductions, but the treaty goes further and requests signatories' countries to take *"legal remedies against the circumvention of effective technological measures that are used by authors in connection with the exercise of their rights"*

As stated by the WIPO, actual copyright agreements cover *"literary works such as novels, poems, plays, reference works, newspapers and computer programs; databases; films, musical compositions, and choreography; artistic works such as paintings, drawings, photographs and sculpture; architecture; and advertisements, maps and technical drawings"*. [WIP08]

These two treaties constitute the base of national copyright laws all around the world and are the main sources of emerging issues, such as software patents and DRM (Digital Rights Management). More specifically WIPO states in their Frequently Asked Questions *"In the 1970s and 1980s, there were extensive discussions on*

¹Berne Convention for the Protection of Literary and Artistic Works was held in in Berne, Switzerland, in 1886.

²Full text of the agreement can be found at <http://www.wipo.int/documents/en/diplconf/distrib/94dc.htm>

whether the patent system, the copyright system, or a sui generis system, should provide protection for computer software. These discussions resulted in the generally accepted principle that computer programs should be protected by copyright, whereas apparatus using computer software or software-related inventions should be protected by patent...Copyright protection of computer software is established in most countries and harmonized by international treaties to that effect. The law relating to the patentability of software is still not harmonized internationally, but some countries have embraced the patentability of computer software and others have adopted approaches that recognize inventions assisted by computer software". [WIP08] This describes pretty well the European situation, where software patents are being actively discussed by the European parliament, but not yet recognized and regulated.

2.1.1.1 What is copyright granting to authors and publishers?

Copyright laws are thus enforced by each country with national laws based upon the above mentioned international agreements. Nevertheless, some of the author's and publisher's rights are common to all countries and the copyright owner can:

- Reproduce the work
- Produce derivative works
- Distribute copies of the work
- Publicly perform and display the work
- Translate and/or adapt the work

2.1.1.2 How to apply copyright to an original creation

- Copyright is naturally acquired once an original creation has been made, it does not depend upon a registration. Anyway national offices for registration of original creation can exist; the WIPO suggest that submission of a work and its registration can be useful in case of legal dispute over the work itself.

2.1.1.3 Why copyrighting a work?

The WIPO states that copyrights “*..are essential to human creativity, by giving creators incentives in the form of recognition and fair economic rewards. Under this system of rights, creators are assured that their works can be disseminated without fear of unauthorized copying or piracy. This in turn helps increase access to and enhances the enjoyment of culture, knowledge, and entertainment all over the world...*”. [WIP08]

Copyright does not force the author to prohibit the above mentioned actions; it only states that those rights are recognized to the copyright's owner. This means that the copyright's owner can prohibit or allow people to perform one or all of these actions.

2.1.1.4 Licensing rights

This is the point where licenses come in: licensing some of the rights held by the copyright owner to users (individuals or organisations), allowing or disallowing certain actions. A common misconception about FLOSS works is that they do not fall

under copyright laws. This is totally untrue: those are copyrighted works like any other and a copyright owner exists for each of them. The difference is made by the license terms under which they are released, granting most of the above mentioned rights to users.

This is what licenses are: legal terms established by the copyright owner to impose restrictions, allow actions, transfer rights to the user. Licenses can deal with each right or restriction in detail.

2.2 PROPRIETARY SOFTWARE AND LICENSES

Proprietary software and proprietary software licenses are out of the scope of this document. Though, the understanding of the definition of proprietary software can ease the understanding of Free and Open Source Software.

Copyrighted software does not necessarily mean proprietary software, while it is almost sure that proprietary software is protected by copyright. Restrictions are not what make the software being proprietary. Proprietary software is characterised by a simple rule: the unavailability of the source code, in a human-readable programming language. This is a common definition of proprietary software. Notably, this definition does not make any assumption about the cost of the software: free of charge software can be proprietary software, according to the above given definition.

In conclusion, proprietary software is characterised by the following properties:

- **Use is restricted.** Examples of common restrictions can be fees, agreements restricting the use to some scopes.
- **Copy is prohibited or restricted.**
- **Redistribution is prohibited or restricted.**
- **Modification is prohibited or restricted** and the software is distributed only in binary form (non human-readable language), there is no access to source code.

Another category of software is commercial software. Commercial and proprietary software must not be confused: most of commercial software is proprietary software, but there exists commercial software which is *free software*³ and/or open sourced.

All of the above listed restrictions are established by licensing terms, imposed by the owner of the copyright (an individual or a company). Restrictions are then enforced by the way of use and copy protection technologies. License terms and national laws can protect the copyright's owner against all those technologies or applications forged in order to by-pass those protections.

2.2.1 Proprietary software licenses

The use of one or more copies of proprietary software is possible only if the user accepts the terms of the software license (commonly referred by some publishers as End User License Agreement). One common characteristic of proprietary software

³Free software definition is given in following paragraphs. Free is not intended as free of charge.

licenses is that the license must be accepted, otherwise the software cannot be used at all. Accepting the license means accepting all the restrictions it imposes. Heavy restrictions are usually imposed in regard to copy, modification and redistribution, but many refer to the use of the software, establishing conditions under which the computer program can be used. It must be observed that these restrictions can go far beyond those established by national laws.

Why the license must be accepted by the end-user to use the software? The reason lays in a subtlety: the end user gets some rights licensed, but the ownership of software and software copy still remains to the publisher. The user paid for the software or he got it for free, this stands true.

2.2.1.1 Microsoft Windows EULA

An example of proprietary software license is the one under which the popular Microsoft Windows operating system is released. Different versions of Microsoft Windows come with different licenses. This happens both for major versions of the operating system (so, e.g., Windows Vista and Windows XP are released under different licensing terms), but even for different editions of the same major version (e.g. Vista Home Basing and Vista Business ships with different licenses). Being by far the most popular proprietary software, it is interesting to have a glance at its licensing schema.

As mentioned in the previous section, the EULA states “*..by using the software, you accept these terms. If you do not accept them, do not use the software. Instead, return it to the retailer for a refund or credit..*”.

It is stated that “*Before you use the software under a license, you must assign that license to one device (physical hardware system). That device is the “licensed device.” A hardware partition or blade is considered to be a separate device.*”. [MVE08]

The use is clearly allowed only to users accepting the EULA. The user is also accepting that the aforementioned use will happen only on one physical device, followed by the definition of what should be considered physical device. This is explicit with the a and b clauses of point 2:

- a) *Licensed Device. You may install one copy of the software on the licensed device. You may use the software on up to two processors on that device at one time. Except as provided in the Storage and Network Use (Ultimate edition) sections below, you may not use the software on any other device.*
- b) *Number of Users. Except as provided in the Device Connections (all editions), Remote Access Technologies (Home Basic and Home Premium editions) and Other Access Technologies (Ultimate edition) sections below, only one user may use the software at a time [MVE08].*

Article four of the EULA states: “*Activation associates the use of the software with a specific device. During activation, the software will send information about the software and the device to Microsoft. This information includes the version, language and product key of the software, the Internet protocol address of the device, and information derived from the hardware configuration of the device...You*

will not be able to continue using the software after that time if you do not activate it.” [MVE08].

Activation is defined as mandatory in the EULA: this is the mechanism to enforce the use of the software on a single device. Other editions of the same software allows for installations and use on multiple device.

Clearly the end user could have the need to reinstall the software on a different machine, due to hardware failures or a simple hardware renewal. In this scenario the user needs to re-activate the software. Regarding this issue, the Windows Vista EULA states, at point 15:

- a) Software Other than Windows Anytime Upgrade. You may uninstall the software and install it on another device for your use. You may not do so to share this license between devices [MVE08].*

This clause has lately changed: a previous version of the agreement imposed a limit of one reassignment at most. As this was source of a considerable controversy, it has since been changed.

Recalling what was observed in the previous paragraph:

“The software is licensed, not sold. This agreement only gives you some rights to use the software. Microsoft reserves all other rights. Unless applicable law gives you more rights despite this limitation, you may use the software only as expressly permitted in this agreement. In doing so, you must comply with any technical limitations in the software that only allow you to use it in certain ways....You may not

- work around any technical limitations in the software;*
- reverse engineer, decompile or disassemble the software, except and only to the extent that applicable law expressly permits, despite this limitation;*
- use components of the software to run applications not running on the software;*
- make more copies of the software than specified in this agreement or allowed by applicable law, despite this limitation;”[MVE08]*

Thus, the license explicitly disallows modifications, reverse engineering and copy of the software (one backup copy is admitted by the license though).

The Vista Home Basic license also covers the virtualization topic, stating that the software cannot be run inside a virtualization environment: *“You may not use the software installed on the licensed device within a virtual (or otherwise emulated) hardware system..” [MVE08].* Licenses covering other editions of this software do not include this clause. This point gives an idea of how licenses change and actualize, covering very specific issues that emerge while new technologies advance.

2.3 THE FREE SOFTWARE FOUNDATION

FSF history is tightly tied to the history of the GNU project, this is where the idea of a funding tax-exempt foundation was conceived. While the FSF is oriented to educating people to free software use and user rights, the GNU foundation is mainly devoted to support free software development.

GNU stands for GNU is Not Unix and was founded in 1984 by Richard Stallman (founder of FSF too) in order to develop a free replacement of the UNIX Operating System. The GNU community, during years, has grown and many developers joined it collaborating in developing replacements for many UNIX common tools. Stallman's initial idea was to develop a free operating system which was UNIX compatible to replace all the proprietary operating systems targeted at workstations and mainframe platforms that were appearing in the early eighties. In the Stallman's view, the operating system was not only the kernel, but the whole set of kernel, compilers, tools, graphical server. This is why the GNU operating system still lacks an operating system kernel: the GNU HURD project (this is the name of the kernel) is alive but somewhat advancing slowly. Stallman work initially concentrated on writing a new complete, portable and multi-target compiler for the C programming language and the GNU Emacs text editor: this was the first popular software of the GNU project (the GCC compiler will make its public appearance only almost ten years later). The success of the text editor encouraged other people to work on other free software tools and a community began to grow: applications and source code were shared among users and released under licensing terms that allowed for copy, modification, redistribution.[RMS98]

In the meantime, starting from the early nineties, the efforts of many developers were focusing on the now popular Linux kernel operating system. Linux has certainly been, during last years, the main way to spread free software and the one developed by the GNU community. The name GNU/Linux, often used by the FSF, explicitly emphasize that the so-called “Linux distributions” are made of the union of Linux operating system (kernel) and the GNU userspace tools.

The Free Software Foundation (FSF) is a donor supported charity born in 1985 with the objective of funding the GNU project, promoting, supporting and spreading the free software, promoting computer user freedom and defending the rights of free software users.



Illustration 1: The Free Software Foundation logo

Introducing the FSF foundation permits us to introduce the concept of Free Software, which is something different from the simple Open Source definition. According to the definition given by the FSF, “Free software is software that gives you the user the freedom to share, study and modify it. We call this free software because the user is free...”. This means that the word “free” has nothing to do with the existence of a fee for the acquisition of the software and related costs. A simple way used by the FSF to

explain the meaning of this world has always been “free as in free speech, not as in free beer”.

FSF supports the choice of free software as a political and ethical choice, the choice to learn and share what has been learnt with others: to learn how software works, modify it and share it again with other people that will perform the same actions. It is clear that this objective is in strong contrast with the clauses of proprietary software licenses we observed in the previous paragraph. The FSF also states that *“the corporations behind proprietary software will often spy on your activities and restrict you from sharing with others. And because our computers control much of our personal information and daily activities, proprietary software represents an unacceptable danger to a free society...”* [FSF08]. This is the main reason why FSF regularly and actively conducts campaigns against all those technologies that constitutes a threat to freedom and privacy of end-users (the one against DRM is probably the best known one).⁴

FSF is the publisher of the GNU General Public License, the most popular free software license under the terms of which are estimated to be released about the 80% or more of the free and open source softwares around the world. Individuals and societies around the world are using this license (GNU GPL v2 is the most popular and widespread, but a new version of the license, the GNU GPLv3, is rapidly spreading too). It is a common practice to make FSF as the copyright holder as this helps when getting in trouble with other people infringing the license.

Introducing the FSF is thus fundamental for the role it played and still plays into software development (through the GNU project) and of the most important and used Free Software Licenses.

2.4 THE OPEN SOURCE INITIATIVE

The Open Source Initiative (OSI) is a public benefit corporation founded in 1998 and is mainly involved in Open Source community building and education, spreading Open Source as a method for developing software.



Illustration 2: The Open Source Initiative logo

⁴<http://www.fsf.org/campaigns> for all the ongoing campaigns.

The history of OSI is tied to a strong personality of the Open Source world, Eric Raymond: *“the formation of OSI began with the publication of Eric Raymond's paper The Cathedral and the Bazaar in 1997. In this paper, Raymond pioneered a new way of understanding and describing the folk practices of the hacker community. His analysis, which centered on the idea of distributed peer review, had an immediate and strong appeal both within and (rather unexpectedly) outside the hacker culture..”* and *“OSI was jointly founded by Eric Raymond and Bruce Perens in late February 1998, with Raymond as its first president and an initial Board of Directors including Brian Behlendorf, Ian Murdock, Russ Nelson, and Chip Salzenberg. Raymond served as president until 2005 and afterwards remained a nonvoting observer active in the Board's work; Perens, originally Vice-President, resigned in 1999 over policy differences with the Board and distanced himself from the organization..”*⁵.

The OSI is responsible for the Open Source Definition⁶ and for compliance of the Open Source licenses: a license can be defined as Open Source Licence only if it respects the Open Source Definition made by the OSI. Licenses can be submitted to the OSI for review of Open Source Definition compliance.

The OSI proposes a different approach to software than the one proposed by the FSF: Open Source is mainly a method, not a political or ethical choice. This method can substantially be resume by ten rules that OSI wrote for people interested in the definition of Open Source.

According to the OSI, in fact, Open Source does not only mean access to the source code, but free redistribution with no restrictions, possibility for modification and derived works with no discrimination against groups of people and restrictions to fields of endeavour, technological neutrality⁷.

Some of the ideas behind these rules are common to those proposed by the FSF, but they have different goals: for the OSI it is all about how the software is developed, with no assumptions about social and ethical problems or about threats to user freedoms. This is the reason why often Free Software licenses are Open Source approved licenses but Open Source licenses are not Free Software licenses too: the ideas behind these two families of licenses are really different.

2.5 FLOSS LICENSES

This section will offer a general overview of Open Source and Free Software licenses. It will not deal with each license, but with the ideas and constraints that hold behind two big families of licenses: open source and free software licenses.

⁵<http://www.opensource.org/history> for a full history of the Open Source Initiative.

⁶The definition will be discussed later in this document.

⁷Ten rules of Open Source Definition at <http://www.opensource.org/docs/osd>

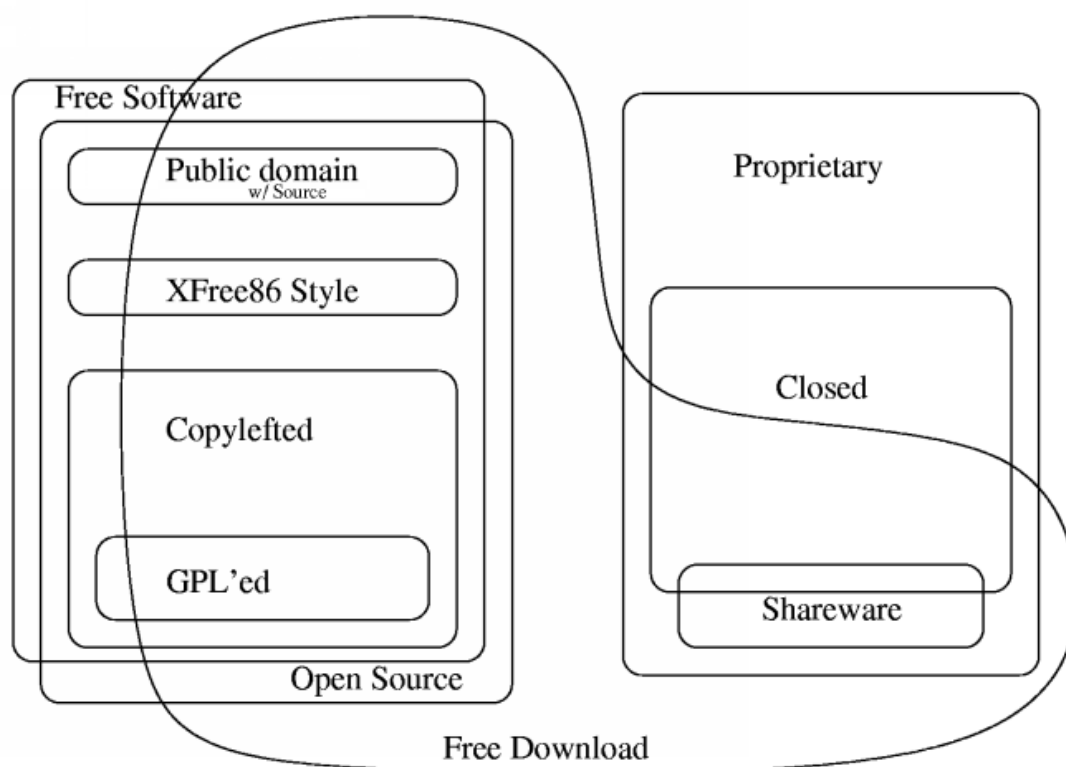


Illustration 3: Open Source, Copyleft and Free Software families

2.5.1 Open source licenses

As previously observed, Open Source licenses are all those licenses approved as such by the Open Source Initiative. The criteria used by the OSI to estimate the “open sourcedness” of a license are well described in the following ten rules by OSI which forms the complete Open Source definition:

1. **Free Redistribution.** *The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.*
2. **Source Code.** *The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.*
3. **Derived Works.** *The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.*

4. ***Integrity of The Author's Source Code.*** *The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.*
5. ***No Discrimination Against Persons or Groups.*** *The license must not discriminate against any person or group of persons.*
6. ***No Discrimination Against Fields of Endeavor.*** *The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.*
7. ***Distribution of License.*** *The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.*
8. ***License Must Not Be Specific to a Product.*** *The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.*
9. ***License Must Not Restrict Other Software.*** *The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.*
10. ***License Must Be Technology-Neutral.*** *No provision of the license may be predicated on any individual technology or style of interface [OSD08].*

One of the principles behind these rules is evolution. Evolution of software is possible only when people are free to use the software, modify it and having it work better than the original version, adding new features, using it as the base upon which building a completely new project. This is why open source licenses must necessarily force the distribution of the source code and, more precisely, it is ensured that the source code must be human-readable and not obfuscated in any way. Supporting a *peer-to-peer* and rapid evolutionary based model of the software, cannot live just by releasing the source code and have it available. It is fundamental that the source code can be modified and redistributed freely with just one limitation: the license must ensure that all modifications must be clearly marked with the indication of who modified what.

The OSI not only provides these rules, it offers a public review process to companies and individuals that are interested in receiving the Open Source label for their own license. This public review proves aim at several objectives:

- **Assures the compliance of the license to the OSD:** the license is reviewed and the compliance to the criteria above described is evaluated. This step is

fundamental, obviously, for the acceptance of the license in the family of Open Source licenses.

- **Assigns the license to an appropriate license proliferation category:** when a license is accepted and labelled as Open Source, the next step is its categorization. OSI currently distinguishes the following families:
 - Licenses that are popular and widely used or with strong communities
 - Special purpose licenses
 - Other/Miscellaneous licenses
 - Licenses that are redundant with more popular licenses
 - Non-reusable licenses
 - Superseded licenses
 - Licenses that have been voluntarily retired
 - Uncategorized Licenses
- **Discourages duplicate licenses:** one of the worst problems with open source licenses is related to an excessive license proliferation during last years. Most popular licenses (such as the GPL or the BSD license) have been often forked so to be adapted to one project's needs. OSI tries to avoid this proliferation of forks refusing to admit duplicate licenses (or twin licenses, e.g. same licensing terms but different names).
- **Provides a timely review process (It is fixed in 60 days)**

Individuals and companies can access the review process submitting the license via mail and then following a few simple steps⁸.

After the submission, a first discussion takes place in the License Review Community, a mailing list, for at least 30 days. The inscription to this mailing list is free and the archives of the discussion threads are publicly accessible via web⁹. Starting from the discussion, “tickets” will be filed and the License Review Chair will summarize the discussion in order to report to the OSI Board. The OSI Board is responsible for the final decision (the admittance of the license). Once the board takes this decision, the License Review Chair is notified and can report the decision back to the mailing list. If the process succeeds, the OSI site is updated adding the new license, its full text in the appropriate proliferation category.

2.5.2 Free software licenses and copyleft

Free software is a sub-family of open source. All applications released under the terms of a free software license are therefore considered “free software”.

⁸For a full process description refer to <http://www.opensource.org/approval>

⁹More infos at <http://www.crynwr.com/cgi-bin/ezmlm-cgi?17>

The keyword is “free”: as observed, this word is used by FSF to mean “freedom” as in “freedom of speech”. The definition of “free software” was firstly formulated and published by Richard Stallman in a 1986 essay and is nowadays available on the GNU project site, philosophy section¹⁰.

The following text constitutes the free software definition accepted by the FSF and GNU projects.

“Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer.

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- ***The freedom to run the program, for any purpose (freedom 0).***
- ***The freedom to study how the program works, and adapt it to your needs (freedom 1).*** Access to the source code is a precondition for this.
- ***The freedom to redistribute copies so you can help your neighbour (freedom 2).***
- ***The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits (freedom 3).*** Access to the source code is a precondition for this.

A program is free software if users have all of these freedoms. Thus, you should be free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere. Being free to do these things means (among other things) that you do not have to ask or pay for permission.

You should also have the freedom to make modifications and use them privately in your own work or play, without even mentioning that they exist. If you do publish your changes, you should not be required to notify anyone in particular, or in any particular way.

The freedom to run the program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job and purpose, without being required to communicate about it with the developer or any other specific entity. In this freedom, it is the user's purpose that matters, not the developer's purpose; you as a user are free to run a program for your purposes, and if you distribute it to someone else, she is then free to run it for her purposes, but you are not entitled to impose your purposes on her.

The freedom to redistribute copies must include binary or executable forms of the program, as well as source code, for both modified and unmodified versions. (Distributing programs in runnable form is necessary for conveniently installable free operating systems.) It is ok if there is no way to produce a binary or executable form for a certain program (since some languages don't support that feature), but you must have the freedom to redistribute such forms should you find or develop a way to make them.

¹⁰For more informations: <http://www.gnu.org/philosophy/>

In order for the freedoms to make changes, and to publish improved versions, to be meaningful, you must have access to the source code of the program. Therefore, accessibility of source code is a necessary condition for free software.

One important way to modify a program is by merging in available free subroutines and modules. If the program's license says that you cannot merge in a suitably-licensed existing module, such as if it requires you to be the copyright holder of any code you add, then the license is too restrictive to qualify as free.

In order for these freedoms to be real, they must be irrevocable as long as you do nothing wrong; if the developer of the software has the power to revoke the license, or replace it with a different license (since this implies revoking the old license), without your doing anything wrong to give cause, the software is not free.

However, certain kinds of rules about the manner of distributing free software are acceptable, when they don't conflict with the central freedoms. For example, copyleft (very simply stated) is the rule that when redistributing the program, you cannot add restrictions to deny other people the central freedoms. This rule does not conflict with the central freedoms; rather it protects them.

Free software does not mean non-commercial. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important. You may have paid money to get copies of free software, or you may have obtained copies at no charge. But regardless of how you got your copies, you always have the freedom to copy and change the software, even to sell copies.

Whether a change constitutes an improvement is a subjective matter. If your modifications are limited, in substance, to changes that someone else considers an improvement, that is not freedom.

However, rules about how to package a modified version are acceptable, if they don't substantively limit your freedom to release modified versions, or your freedom to make and use modified versions privately. Rules that if you make your version available in this way, you must make it available in that way also can be acceptable too, on the same condition. (Note that such a rule still leaves you the choice of whether to publish your version at all.) Rules that require release of source code to the users for versions that you put into public use are also acceptable. It is also acceptable for the license to require that, if you have distributed a modified version and a previous developer asks for a copy of it, you must send one, or that you identify yourself on your modifications.

In the GNU project, we use copyleft to protect these freedoms legally for everyone. But non-copylefted free software also exists. We believe there are important reasons why it is better to use copyleft, but if your program is non-copylefted free software, it is still basically ethical.

See Categories of Free Software for a description of how free software, copylefted software and other categories of software relate to each other.

Sometimes government export control regulations and trade sanctions can constrain your freedom to distribute copies of programs internationally. Software developers

do not have the power to eliminate or override these restrictions, but what they can and must do is refuse to impose them as conditions of use of the program. In this way, the restrictions will not affect activities and people outside the jurisdictions of these governments. Thus, free software licenses must not require obedience to any export regulations as a condition of any of the essential freedoms.

Most free software licenses are based on copyright, and there are limits on what kinds of requirements can be imposed through copyright. If a copyright-based license respects freedom in the ways described above, it is unlikely to have some other sort of problem that we never anticipated (though this does happen occasionally). However, some free software licenses are based on contracts, and contracts can impose a much larger range of possible restrictions. That means there are many possible ways such a license could be unacceptably restrictive and non-free.

We can't possibly list all the ways that might happen. If a contract-based license restricts the user in an unusual way that copyright-based licenses cannot, and which isn't mentioned here as legitimate, we will have to think about it, and we will probably conclude it is non-free.

*When talking about free software, it is best to avoid using terms like give away or for free, because those terms imply that the issue is about price, not freedom. Some common terms such as piracy embody opinions we hope you won't endorse. See *Confusing Words and Phrases that are Worth Avoiding* for a discussion of these terms. We also have a list of translations of free software into various languages.*

Finally, note that criteria such as those stated in this free software definition require careful thought for their interpretation. To decide whether a specific software license qualifies as a free software license, we judge it based on these criteria to determine whether it fits their spirit as well as the precise words. If a license includes unconscionable restrictions, we reject it, even if we did not anticipate the issue in these criteria. Sometimes a license requirement raises an issue that calls for extensive thought, including discussions with a lawyer, before we can decide if the requirement is acceptable. When we reach a conclusion about a new issue, we often update these criteria to make it easier to see why certain licenses do or don't qualify..." [GFD08].

It is quite clear, as it is even explicitly stated, that this definition requires careful reading and thought to be fully understood. Comparing this definition to the Open Source one reported in the previous paragraph is difficult, as the approach is completely different: the open source definition targets at several "technical" licensing points that must be covered by the license to be labelled as open source. The free software definition asks something more: user rights are somewhat superseded by user freedoms and the points that the license must cover are in the shape of freedoms that must be granted to the user. This definition also asks to understand all those ethical problems that stand behind human works and creativity (and this does not apply just to software). Free software versus proprietary software is not to be considered then just as a matter of technical solutions and possibilities. It is a social and ethical matter because it is related to freedoms, not just rights: freedoms are intrinsic to the human nature; rights (possibly based upon these freedoms) are usually enforced by an authority using laws. Using rights to preserve

freedom is the final objective and free software licenses are the mean to enforce these rights.

Free Software licenses are strongly characterized by the Copyleft idea, which represents the main and very strong difference with the more generic “open source” world. It is of big relevance, thus, to introduce the concept of copyleft briefly.



*Illustration 4: Copyleft
logo: all rights
"reversed"*

The word Copyleft is clearly a pun around the word “copyright”. A wrong but common thought, is that copyleft stands for “not copyrighting”, uncopyrighted. This assumption is completely wrong, because the spirit of copyleft is to make an alternative use of copyright rules.

A software work can be published with no copyright notice at all, thus released to the public domain uncopyrighted, and source code can be released. This is very easy way to release a project: in this way people can take the code, modify it, compile it and redistribute it, with no restriction at all. There is no need for a license, in this scenario, because no one owns the copyright. Having no restrictions, people could also take the released code, modify it and redistribute in binary form, without disclosing the source code of the modifications: acting this way “..people who receive the program in that modified form do not have the freedom that the original author gave them; the middleman has stripped it away” [GCL08].

Copyleft says that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it. Copyleft guarantees that every user has freedom [GCL08].

Copyleft was conceived in order to grant and retain those freedoms that were originally licensed to users by the author. This mechanism, which is at the base of free software licenses, is often referred as “viral”: the terms and the freedom of the original license must be retained in derivative works. There is no identity between copyleft licenses and “viral” licenses: some “viral” factors characterize many other licenses, some Creative Commons licensing schemes being the most notable and popular examples. It must be observed that the term “viral” is almost often used with a bad touch or even derisively and the FSF does not support or like the use of this term.

As FSF states “..Copyleft is a general concept, and you can't use a general concept directly; you can only use a specific implementation of the concept. In the GNU

Project, the specific distribution terms that we use for most software are contained in the GNU General Public License...”

Often, free software applications have a copyright notice stating that all rights are reserved to the Free Software Foundation, not to the authors. According to the FSF there is a valid reason to do so and assign the copyright to the foundation and this has to do with GPL enforcement in case of disputes:

“Under US copyright law, which is the law under which most free software programs have historically been first published, there are very substantial procedural advantages to registration of copyright. And despite the broad right of distribution conveyed by the GPL, enforcement of copyright is generally not possible for distributors: only the copyright holder or someone having assignment of the copyright can enforce the license. If there are multiple authors of a copyrighted work, successful enforcement depends on having the cooperation of all authors.

In order to make sure that all of our copyrights can meet the recordkeeping and other requirements of registration, and in order to be able to enforce the GPL most effectively, FSF requires that each author of code incorporated in FSF projects provide a copyright assignment, and, where appropriate, a disclaimer of any work-for-hire ownership claims by the programmer's employer. That way we can be sure that all the code in FSF projects is free code, whose freedom we can most effectively protect, and therefore on which other developers can completely rely..” [MOG08].

Just like the OSI, FSF and GNU projects offer collaboration and help to individuals and organisations involved in the process of writing a new license, analysing its compliance to the free software definition. As excessive license proliferation can be a real problem for users and developers, the FSF and GNU organisations, before considering a new license, will help people finding an already existing license that can fit users needs.

FSF is the publisher of the by far most used and popular free and open source license, the GNU General Public license (GNU-GPL, but we will call it GNU-GPL from now on for the sake of brevity, referring to GPLv2, the second version of this license). Just to have an idea of how spread is this license, statistics¹¹ show that about a 70% of the free and open source software is licensed under the terms of the GPL family of licenses¹².

¹¹Daily statistics update is provided by Blackduck software <http://www.blackducksoftware.com/oss>

¹²The family includes GPLv2, LGPLv2, GPLv3 and LGPLv3

3 EMERGING ISSUES: SOFTWARE PATENTS AND DRM

This chapter will deal with recently emerging issues like software patents and copy and content protection technologies often referred as Digital Rights Management (DRM). A full analysis of these issues could take (and actually takes) entire books. It is out of the scopes of this document to give an in-depth look about these themes. Thus the reader should consider this chapter as a starting point to look at this issue which could be best understood if analysed in a social and ethical perspective.

3.1 WHAT ARE SOFTWARE PATENTS

Before trying to give a definition of software patents, it is important to make one point clear: they are a reality in the United States of America, but not in Europe. The European Parliament has been working for years to a law that legalises patents in the European Community too, but, up to now, every proposal has failed due to a strong resistance made by some European governments and a big number of organisations.

Each European country still holds its own patent office and its own patents laws. There is an ongoing effort to uniform all patents laws in all the so-called “developed” countries. This effort is mainly made by trade organisations (WTO and WIPO among them). As for copyright, these organisations developed a common set of rules which constitutes a minimal standard that national laws should enforce. Two main treaties must be recalled: Patent Cooperation Treaty (PCT), dated year 1970, and Trade-Related Aspects of Intellectual Property Rights (TRIPS), signed in year 1995.

The simplest definition of software patents that can be given is “*a patent intended to prevent others from using some programming technique or algorithm*”. Another possible definition can be as a “*patent on any performance of a computer realised by means of a computer program*”¹³. Finally, a third definition as a “*software patent is a patent for a computer program claimed as such, or an algorithm or computer-implemented business method that make no technical contribution...*”.

There is not a single definition of software patents, and each of the above given definitions can easily lead to misunderstandings. Furthermore, patents, much like as for copyright, are enforced by the mean of national laws (at least for those countries where patents are admitted). Each national law could admit a certain family of patents and not admit some other families. For instance, in the UK patents cannot describe algorithms or mathematical methods, in accordance to what prescribed by the European Patent Convention (EPC):

Article 52: Patentable inventions

(1) European patents shall be granted for any inventions which are susceptible of industrial application, which are new and which involve an inventive step.

(2) The following in particular shall not be regarded as inventions within the meaning of paragraph 1:

- *(a) discoveries, scientific theories and mathematical methods;*

¹³In the definition of Foundation for Free Information Infrastructure <http://www.ffii.org/>.

- *(b) aesthetic creations;*
- *(c) schemes, rules and methods for performing mental acts, playing games or doing business, and programs for computers;*
- *(d) presentations of information* [EPO09].

Paragraph 1 of the above article refers to “innovative” and new inventions, where the meaning of innovative can be, at least, subjective. How does this apply to software? How can be distinguished an innovative software by a non-innovative one?

Article 54 gives an answer to this point: it is innovative an invention the does “not form a part of the state of the art...”.

Paragraph 2 exclude from patentability “programs for computers”.

The above article constitutes the point around which a first main branch of the debate started: what is patentable and what is not? While it is explicitly stated that programs for computers cannot be patented, it seems that software patents are going to be accepted by the European Union, following the U.S.A. road to patentability of software with a main difference: in the U.S.A. mathematical algorithms and business methods (but not mathematical formulae). Another requirement is that the patents can be applied only to ideas that have an industrial applicability, where “industrial” stands to every kind of industry. Software patentability is somewhat admitted by the third clause of article 52:

“The provisions of [clause] 2 shall exclude patentability of the subject-matter or activities referred to in that provision only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such...” [EPO09].

This clause can be interpreted by the union of the following points:

- A technical invention can be patentable even if it includes non-technical aspects whose patentability is excluded by the above mentioned article 52 of the EPC. As an example, software can be patentable, even if it includes mathematical algorithms and the application of mathematical formulae.
- At the same time, a non-technical invention cannot be patented as such, if its patentability is excluded by article 52, even if we add some technical features to it. The sole addition of technical features to a business method, as an example, does not make it patentable.

The second main branch is about patent effects on society and industries: how are patents hitting research, innovation and growth. According to some groups this point should be extended to how patents are hitting freedom because of their pervasiveness.

Of interest, the debate around these themes is not limited to Europe, but is particularly active in the U.S.A. where software patents existed for more than two decades and their effects are visible.

3.1.1 Copyright and patents

Copyrights and patents (and even trademarks) are often confused and mixed together, while they are totally different entities. The expression that is often used when this confusion happens is “intellectual property”. To understand the difference between the three, it can be useful to cite an enlightening article¹⁴ by Richard Stallman:

“According to Professor Mark Lemley, now of the Stanford Law School, the widespread use of the term “intellectual property” is a fashion that followed the 1967 founding of the World “Intellectual Property” Organization, and only became really common in recent years. (WIPO is formally a UN organization, but in fact represents the interests of the holders of copyrights, patents, and trademarks.)

The term carries a bias that is not hard to see: it suggests thinking about copyright, patents and trademarks by analogy with property rights for physical objects. (This analogy is at odds with the legal philosophies of copyright law, of patent law, and of trademark law, but only specialists know that.) These laws are in fact not much like physical property law, but use of this term leads legislators to change them to be more so. Since that is the change desired by the companies that exercise copyright, patent and trademark powers, the bias of “intellectual property” suits them.

The bias is enough reason to reject the term, and people have often asked me to propose some other name for the overall category...

The term “intellectual property” is at best a catch-all to lump together disparate laws. Non-lawyers who hear one term applied to these various laws tend to assume they are based on a common principle, and function similarly.

Nothing could be further from the case. These laws originated separately, evolved differently, cover different activities, have different rules, and raise different public policy issues.

Copyright law was designed to promote authorship and art, and covers the details of expression of a work. Patent law was intended to promote the publication of useful ideas, at the price of giving the one who publishes an idea a temporary monopoly over it—a price that may be worth paying in some fields and not in others.

Trademark law, by contrast, was not intended to promote any particular way of acting, but simply to enable buyers to know what they are buying. Legislators under the influence of “intellectual property”, however, have turned it into a scheme that provides incentives for advertising.

Since these laws developed independently, they are different in every detail, as well as in their basic purposes and methods. Thus, if you learn some fact about copyright law, you'd be wise to assume that patent law is different. You'll rarely go wrong!...hus, any opinions about “the issue of intellectual property” and any generalizations about this supposed category are almost surely foolish. If you think all those laws are one issue, you will tend to choose your opinions from a selection of sweeping overgeneralizations, none of which is any good.

¹⁴The full article is at <http://www.gnu.org/philosophy/not-ipr.html>

If you want to think clearly about the issues raised by patents, or copyrights, or trademarks, the first step is to forget the idea of lumping them together, and treat them as separate topics. The second step is to reject the narrow perspectives and simplistic picture the term “intellectual property” suggests. Consider each of these issues separately, in its fullness, and you have a chance of considering them well.”

Let's try to put it in even simpler terms: copyrights were conceived to protect details of an authored work, copyrights protect them from being copied by some other author in some other work.

Patents deal with something completely different: patents do not cover details, patents do cover ideas. One would expect that there are well defined limitations on what is patentable, but, as observed in the previous section, this is not the case. The norms proposed to the European parliament to legalise the software patents were considered very broad. Just to have an idea, the EPO has already accepted patents for ideas like the progress bar¹⁵.

3.1.2 Software patents and research

As far as observed in the previous paragraph, patents should promote innovation by enabling revenues from inventions. But access to patents application is somewhat restricted by costs. This is a first factor that should be considered: whenever costs are high, individuals are not allowed to access the patent submission process.

Software programmers are involved in a continue process of reinventing. Patenting ideas is like stopping them from reinventing a new algorithm to accomplish the idea itself. A clear example could be the one constituted by the popular MP3 compression algorithm, which is patented: many open source and free software are leaving MP3 encoders out of their work cause of patent issues. Multimedia encoders/decoders suite are a large field of research and the MP3 case is a good example to understand how patents and related royalties are used and how they can be a total “show-stopper” for enhancements, innovation and research.

Research and development of the MPEG format is partially open: people can submit for joining the group through local MPEG contacts. This takes to a big number of people (individual, mostly paid by big companies) working on the different MPEG standards, the MP3 format being one of them. In 1998 the Fraunhofer institute began to ask for fees to license the MP3 encoding to free software and open source encoder developers. The MP3 format requires a certain format (bitstream syntax) to comply with the ISO standard. The standard does also offer sample source code for encoding and decoding routines, but this is just published for educational purpose and cannot be used freely. It should be possible to freely implement own algorithms for encoding and decoding MP3, but that's where patents come in: you must pay a fee to patent owners to do so. There are many patents for the MP3 audio compression format, Fraunhofer and Thomson Multimedia are the only two companies enforcing the patent asking for a fee¹⁶. Other companies that take part into the MPEG development and hold patents on some MPEG format could behave in

¹⁵Although this is a common example often used in articles and papers, it is used here because it is a real example of what is patentable.

¹⁶Royalty rates for the mp3 format are available at <http://www.mp3licensing.com/royalty/>

the same way in the future. Fees do apply in terms of number of copies shipped for software or unit shipped (in case of an embedded device like a portable MP3 player).

Another popular case, is the one involving the GIF format which used the Lempel-Ziv-Welch (LZW) compression algorithm, patented to Unisys. Unisys even decided to sue after all those web-sites that were using images encoded in the GIF format (so they were not only after those software developers using this format into their products).

Considering that software development (and open and free software development even more) evolves rapidly by its nature and it has evolved in the last 50 years at an impressive rate, how can patents be an incentive to innovation? Aren't them more likely to be a stop to human inventive? If one has an innovative idea building up from the MP3 format, what can be done to avoid patents? Reinventing the wheel each time is not reasonable and cannot be the right solution. Deutsche Bank Research in year 2004 released a paper about software patents and their influence on research. The following excerpt explains their position over this issue:

"A growing number of R&D-intensive businesses realises that licensing out their IP (intellectual property) can constitute a substantial share of their revenues, which in turn encourages innovation efforts. Bearing this in mind, one could be tempted to consider ever stricter IP protection regimes to provide ever more stimuli for innovation.

This conclusion is wrong, however. A prime example is patents on software, which might at first sight be seen as a logical expansion of the classic technology patent. But creating software differs markedly from creating machinery and the like: MIT researchers Bessen and Maskin argue that innovation in software is both strongly sequential (one invention building on a previous one) and complementary (thriving on parallel approaches to the same problem), far more so than in other technology fields. In fact, they found empirical evidence that software patenting substitutes R&D activity, rather than encouraging it, and conclude: "For industries like software or computer, there is actually good reason to believe that imitation becomes a spur to innovation, while strong patents become an impediment" 2. In accordance with other academics, they strongly favour copyright over patent protection for software. Copyrighting provides both adequate leeway for sequential innovation and enough protection for marketable software products..." [DBR04].

On the other hand, software patents supporters still argue that making patents easier to claim should allow for more revenues for patents holders and this should incentive a major effort into the cycle Research-Development-Patent.

3.1.3 Software patents and the EU

As previously observed, under article 52 of the EPC, programs of computer are completely out of the scope of patentability. The article, according to patents supporters, must be read as "patents on computer programs must not apply to software as such". The software must then provide some new, innovative technical and technological idea to be patented.

Software patentability is treated as a case needing a new detailed legislation, this is the main reason behind the decision to extend existing laws about patents with software patents related new laws.

Several tries to extend the existing laws were stopped by the vote made by the European Parliament on July the 6th 2005: the European commission that wrote the text of the law did not take into account the opinion of the Parliament and presented a text that did not include several changes requested by the Parliament itself. This led to a first vote against the proposal with quite a net result (648 votes against 14, the whole parliament is made of 680 members). In the meantime, outside the European Parliament, movements against software patentability, made of individuals, associations and industries, gained popularity and consensus with a campaign that lasted seven years and that had its own victory in 2005. The text of the law was somewhat restarted so to be rewritten by the dedicated European Commission. This does not mean that software patents are not accepted in the EU, as the EPO is accepting them. Examples of accepted licenses:

- **Webshop:** Selling things over a network using a server, client and payment processor, or using a client and a server.
Patents no. EP803105, EP738446, EP1016014
- **Order by cell phone:** Selling over a mobile phone network.
Patent no. EP1090494
- **Shopping cart:** Electronic shopping cart.
Patent no. EP807891
- **[CDs] [Films] [Books]:** Tabbed palettes and restrict search.
Patents no. EP689133, EP1131752
- **Picture link:** Preview window.
Patent no. EP537100
- **Get key via SMS:** Sending key to decrypt bought data via mobile phone network.
Patent EP1374189
- **View film:** Video streaming (segmented video on-demand).
Patent no. EP633694
- **Copy protection:** Encrypt file so it can only be played on authorised devices.
Patent no. EP1072143
- **Credit card:** Pay with credit card on the Internet.
Patent no. EP779587
- **Adapt pages:** Generate different web page depending on detected device.
Patent no. EP1320972 [NSP06]

3.2 DIGITAL RIGHTS MANAGEMENT

Under the name Digital Rights Management (DRM) fall all those new technologies ideated, developed and deployed in order to:

- protect copyrights
- limit what users can do with a product
- impose rigid business models

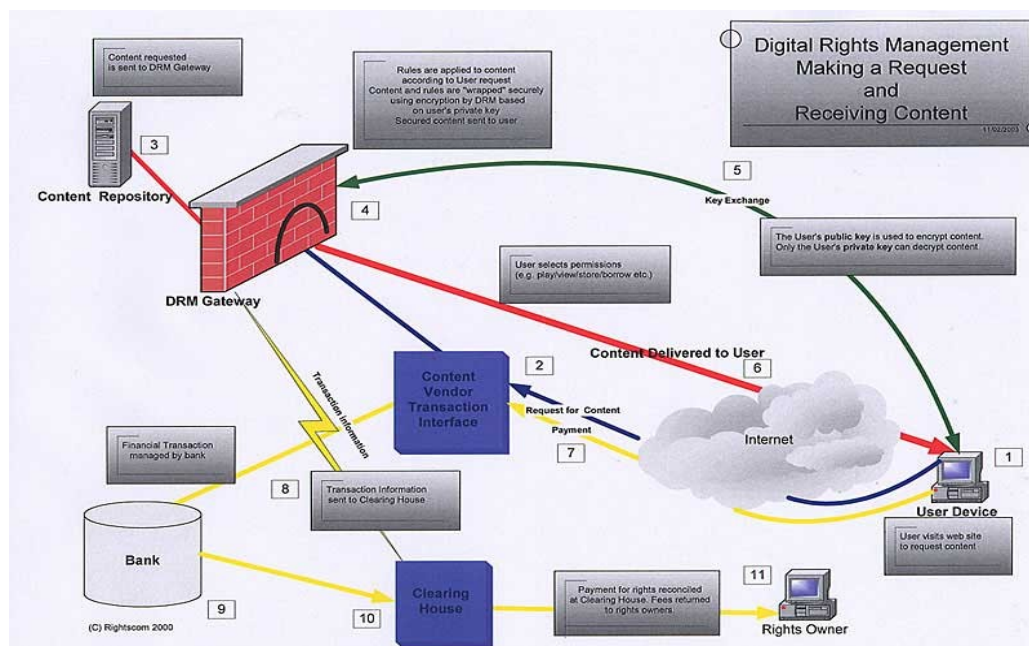


Illustration 5: A classical DRM content distribution scheme

It is common to read and hear of DRM when referring to multimedia content, because this is one of the fields where protection technologies are widely used. With the widespread of digital multimedia devices and file sharing, many DRM technologies found their way into market.

Among the most common options, it is worth of note citing the Content Scrambling System (CSS), used for the encryption of multimedia DVDs. CSS was based on a 40 bit encryption scheme. Content was encrypted and could be decrypted with the use of keys. These keys were distributed by the DVD Forum to licensed software and hardware producers, so that these products could reproduce the media.

The technology permitted to deny:

- playback of the content
- copy operations

But, as many other DRM technologies that would have come later, CSS lasted the time of a few months: during October 1999, three programmers distributed a software program, called DeCSS, that enabled decryption of DVD content. The source code for this program was then released, most of the code was written following a

reverse engineering process. Many national laws still forbid reverse engineering if used to circumvent and disable DRM technologies (and more generally copy protection technologies). This was the main reason for DeCSS to be explicitly prohibited in many countries, with providers hosting the files raided by local authorities and the single publicly known developer persecuted by the police.

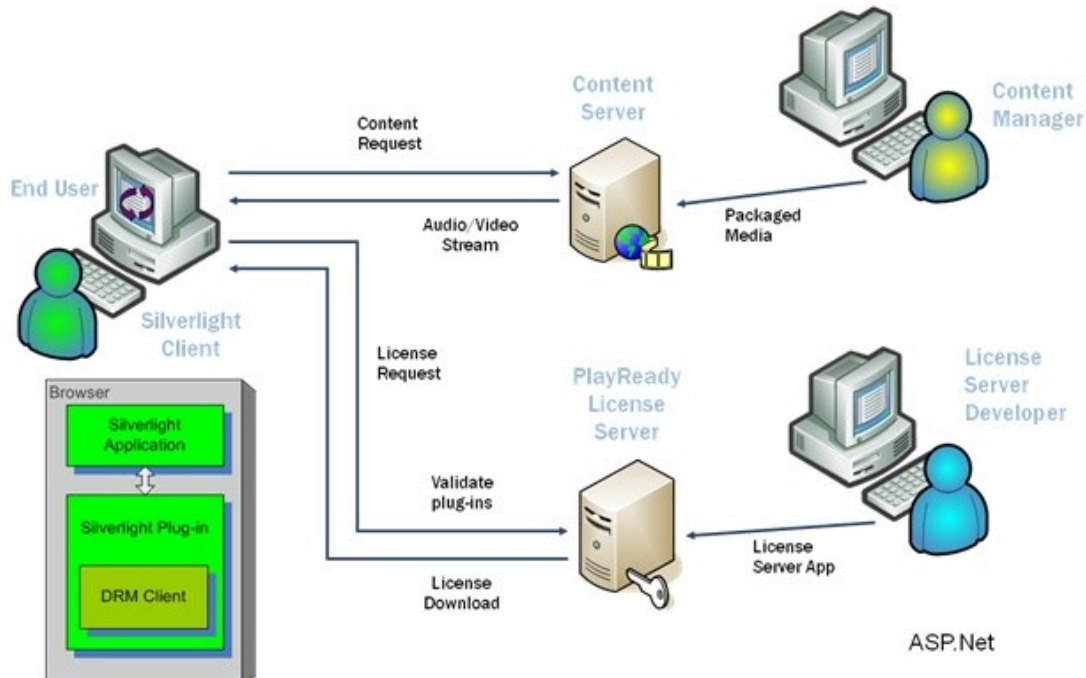


Illustration 6: Microsoft Silverlight DRM contents distribution

During years, many other DRM technologies have been “cracked” by users, developers, researchers or just curious people and really often this happened in response to corporations proposing stronger and stronger protection technologies which badly hit user's freedom, even when enjoying paid contents.

The growth of on-line music stores has seen a widespread diffusion of DRM, but, at the same time, has permitted music producers and distributors to really think about new business models not relying on freedom restrictions and copy protection: it is recent news that most popular on-line music stores are beginning to offer DRM-free high quality media contents for at an increased price and some other music *majors* admitting that the costs due for the deployment of DRM technologies are not worth the final result (with regard to the quick reverse engineering and “cracking” processes).

A popular story in the open source and free software community is the one regarding the TiVO set-top box. This digital multimedia device makes use of the Linux kernel and GPLed software for the user-space. As requested by the GPL license, all of the modifications made by the producer to the original code was made available to the public and, where required by the terms of the GPLv2 license, released under the same license. This basically implies that people could modify the software so to better fit their own needs, recompile it and load it on the device.

This was not the case for TiVO: the producer of the device implemented a hardware-based mechanism that impedes the execution of binaries that do not bring a private key based digital signature. This stops people to have full control on the device and apply modifications, customisations or add enhancements. So, once one buys the device, he has limited freedom to decide over its features. This is exactly the kind of freedom that the GPL tries to preserve but that, in the TiVO case, could not legally preserve. TiVO producer was somewhat able to circumvent all of the legal clause of the GPL limiting the user freedom [ORI06].

The TiVO case opened a new front of discussion: during the last few years, multimedia entertainment has become more and more a field for experimenting for DRM technologies. Not only is high definition content encrypted (this is true for the new HD-DVD and BluRay formats too), but some hardware protection is implemented using DRM enabled hardware components. To make it clearer, it is possible to look at the High Definition Multimedia Interface (HDMI), which is becoming common on relatively new consumer-market hardware. This interface is used to transmit an audio and video stream, via a digital cable, from a graphical adapter to a video device. The device has an additional functionality: it supports some private key based protection schemes, namely High-Bandwidth Digital Content Protection (HDCP): when streaming high definition (or high bandwidth) media streams through two nodes (e.g. your BluRay equipped laptop and a television) via HDMI, both of them must support HDCP and “authenticate” showing digital signature based on a private key based encryption scheme. If one of the two ends do not support HDCP or if a protection violation of the content is detected, the interface can refuse the stream (so the target device will not display the content) or degrade the quality of the content (the target device will display the content at a low bitrate) [FIS05].

All of this can be seen as an essay of the so-called Trusted Computing platform: hardware devices that interoperate correctly (or as they are expected to interoperate) only with other trusted devices and refuse to work correctly (or refuse to work at all) with non trusted devices and contents. The *tivoisation* process opens another battlefield, where hardware vendors could decide about what a user can execute or not: a media player, an operating system or the user favourite text editor. In the perspective of a trusted platform it seems that users are the only ones who are never trusted.

3.3 HOW LICENSES ARE ADDRESSING EMERGING ISSUES

This section will briefly deal with how licenses are addressing the issues exposed in previous sub-sections. Before doing so, it must be observed that the majority of available open source and free software licenses are not even aware of the problems related to software patents and DRM, probably because they are not felt as a threat to the whole free software community or are somewhat considered as socio economic and political problems that software licenses should not address.

The Free Software Foundation recently released the GNU GPLv3 license. The following quote explains perfectly what is changed between GPLv2 and this last version:

“We update the GPL to protect its copyleft from being undermined by legal or technological developments. The most recent version protects users from three recent threats:

- *Tivoization: Some companies have created various different kinds of devices that run GPLed software, and then rigged the hardware so that they can change the software that's running, but you cannot. If a device can run arbitrary software, it's a general-purpose computer, and its owner should control what it does. When a device thwarts you from doing that, we call that tivoization.*
- *Laws prohibiting free software: Legislation like the Digital Millennium Copyright Act and the European Union Copyright Directive make it a crime to write or share software that can break DRM (Digital Restrictions Mismanagement; see below). These laws should not interfere with the rights the GPL grants you.*
- *Discriminatory patent deals: Microsoft has recently started telling people that they will not sue free software users for patent infringement—as long as you get the software from a vendor that's paying Microsoft for the privilege. Ultimately, Microsoft is trying to collect royalties for the use of free software, which interferes with users' freedom. No company should be able to do this...”* [SMI08].

4 LICENSES BENCHMARKING METHODOLOGY

This section will deal, in first instance, with what is meant for license benchmarking.

In second instance this section will deal with benchmarking criteria elicitation. As observed in previous chapters, there are different conceptions of what a FLOSS license is. The two main branch of thought are those represented by the OSI and the FSF: if for the former an Open Source license is mostly a technical issue, for the Free Software Foundation introduces some ethical and social factors. Shall social and ethical factors be introduced in the process of benchmarking? Is it possible and is fair to declass ethical and social factors to technical matters?

So, what are the key factors for license “benchmarking”?

4.1 ABOUT LICENSE BENCHMARKING METHODOLOGY

The idea proposed in this document is to deploy a benchmarking process that enables:

- license elicitation
- license key factors and criteria enumeration
- context-based evaluation

The attended outputs are:

- a comparative view of most (or not so) popular licenses
- a many-to-many match between licenses and the most popular FLOSS business models
- best practices in licensing

The most difficult part in the whole process is constituted by key factors and criteria elicitation and their subsequent definition. For the sake of clearness: are ethical and social factors to be considered key factors and their related criteria included among the benchmarking ones? There is a huge debate about this argument. Emerging issues, like DRM and Software Patents, can be viewed certainly in an ethical perspective. Their acceptance, at least, is tightly tied to the debate about intellectual property.

Although interesting, the whole debate is out of the scope of this document, but DRM and Software patents can be key factors for a license, technical factors and not only ethical factors, once they influence and hugely impact the development of free and open source software.

Operatively, the idea is to apply and exercise the Qualification and Selection of Open Source (QSOS) methodology to licenses.

We believe it is possible to individuate some critical criteria for free software and open source licenses and to work on them as if they were requirements for the software (or the research) project.

This process will lean on a customized QSOS methodology and the custom O4S tool developed in the scope of SHARE Project.

The evaluation of licenses can vary much according to the context: there are environmental and economical factors that can interfere with

- Weighting of criteria
- Scoring

Two main contexts were identified, following a common naming convention that is really popular among FLOSS projects:

- Community context
- Enterprise context

This distinction must not lead to a misconception: both the “community” and “enterprise” contexts are communities in its common meaning, but their scopes, organisational structures, targets and, more generally, characteristics can be very different (e.g. a non-profit organisation and a big corporation). Several subgroups can be identified too inside the above two main families, up to a very fine-grained group distinction. The two terms “community” and “enterprise” were thus selected due to the fact that many big FLOSS projects often follow a dual release scheme that usually presents a fork between a “community supported” edition and an “enterprise edition”. These distinctions are usually made on the level of services offered by the enterprise itself to customers, but often implicate two separate and distinct business models to which different licensing schemes best fit too.

4.2 KEY FACTORS AND CRITERIA FOR FLOSS LICENSES

In the process of evaluating the license applicability, criteria have been split into some macro-areas:

- **Generic criteria:** these are criteria applicable to all floss licenses.
- **Software patents related criteria:** these are all the criteria related to software patents emerging issue. Only a few licenses cover this topic.
- **DRM related criteria:** these are all the criteria related to Digital Rights Management protection technologies. As for software patents, only a few licenses are currently covering this topic.
- **Discriminations:** these are criteria to evaluate whether the license contains restrictions against a group of users or a field of endeavour.

Criteria clustering, as enabled by the QSOS methodology, apply to the Generic criteria macro-area. It serves as a container area for other macro-areas:

- **Distribution terms:** these are all the criteria related to the distribution of third parts original software.
- **Derivative works related criteria:** these criteria deal with derivative works distribution. There is a huge debate about what is to be considered as a derivative work. Different licenses make different assumptions about this.
- **License compatibility:** compatibility analysis can be quite an hard task, but is a necessary step when mixing up licenses (e.g. different components of the same software licensed under different licenses) or when dual licensing a part of the code.

4.2.1 Generic criteria

4.2.1.1 Copy

Does the license allow copy of the software? This criterion does not refer to the case of “backup copies”, it is instead meant as unlimited copies and transfers by the mean of some media (cd, portable storage device and internet).

Scores:

- 0 If the license does not allow copy
- 1 If copy is restricted by some terms
- 2 If there are no limits to copy

Redistribution

Does the license permit redistribution of the software? This criterion is intended to cover one particular redistribution case which is when modifications to the original software have been applied. Source and binary form redistribution are covered by other following criteria.

4.2.1.2 Attribution clause

Does the license request an attribution clause? This means that, after applying modifications, an attribution clause to authors of the original unmodified software must be attached.

Scores:

- 0 if the license does require an attribution clause
- 1 if the license does optionally require an attribution clause
- 2 if the license does not require an attribution clause

4.2.1.3 Additional restrictions

Does the license permit to specify additional restrictions? They are meant to be additional user-made clauses to add restrictions. This means that users can modify the license adding clauses to it.

Scores:

- 0 the license permits additional restrictions
- 1 the license permits limited additional restrictions
- 2 the license does not allow for additional restrictions

4.2.1.4 Retain Copyright

Does the license require the software to retain a copyright notice? This means that, after having applied modifications to a software, we must retain the copyright notes of the original software.

Scores:

- 0 the license requires to retain the copyright notes
- 1 the license requires to retain the copyrights under some terms
- 2 the license does not require to retain the copyright

4.2.1.5 Redistribution under a different license (retain same license)

Does the license allow redistributing the software under a different license? This is sometimes referred as “reciprocity” clause: modify the software and redistribute it under the same original license.

Scores:

- 0 yes
- 1 yes under restrictive terms
- 2 no

4.2.1.6 Modifications

Does the license permit modifications to the code? A free software or open source license should allow for modifications but certain licenses can be very restrictive about modifications.

Scores:

- 0 no
- 1 yes, under some restrictions
- 2 yes

Distribution terms

Under which form can the software be distributed? Binary and source forms (machine-readable and human-readable) are the two possible forms.

4.2.1.7 Source code access

Does the license provide source code access? Three possibilities are provided: no access to the sources, access under some terms, free and unlimited access. Releasing a program under a license that requires source access without releasing the source code results in a violation of the license itself. There are, on the other hand, free license that are really permissive and allow the user to release binary-only forms of the software.

Scores:

- 0 if the license does not guarantee access to the sources
- 1 if the license guarantees access under some restrictions
- 2 if the license guarantees free and unlimited access

4.2.1.8 Binaries distribution

Does the license permit distribution in binary form? This is intended not as binary-only, but binaries with source access. This may for example affect major Linux distributions where binaries are released and installed but source code access is possible.

Scores:

- 0 no
- 1 optionally
- 2 yes

4.2.1.9 Binary only distribution

Does the license permit binary only distribution? This equals to the license permitting to close the source of an application.

Scores:

- 0 no
- 1 under given circumstances
- 2 yes

4.2.1.10 Mixing with proprietary software

Does the license allow to mix released code with proprietary (close-source) code?

Scores:

- 0 yes
- 1 yes, under some terms
- 2 no

Compatibility

License compatibility is a fundamental cluster of criteria when mixing licensing schemes in different software components which are part of the same whole work. On this point, again, there are licenses that are very permissive and licenses that are restrictive.

4.2.1.11 GPLv2 Compatibility

Is the license compatible with the GNU GPLv2 license?

Scores:

- 0 no
- 1 partially
- 2 yes

4.2.1.12 GPLv3 Compatibility

Is the license compatible with the GNU GPLv3 license?

Scores:

- 0 no
- 1 partially
- 2 yes

4.2.1.13 Copyleft License

Is the license copyleft? Copyleft definition was given in chapter 2 of this document. As many license features, this too could be sometimes unclear. Besides that, it was made a distinction among no copyleft at all, weak copyleft and strong copyleft in order to reflect the terms used in the widespread debate over free licenses.

Scores:

- 0 no copyleft
- 1 weak copyleft
- 2 strong copyleft

Derivative works

The real issue about derivative work is constituted by the answer to the “simple” question: “what is a derivative work?”.

As observed when introducing the copyright concept, it is up to national laws to enforce copyright. This stands true in this case too: explicitly defining what is to be considered a derivative work is a matter delegated to national laws. Nevertheless it is possible to refer to the United States Copyright Act (title 17 of United States Code), as national copyright laws are quite standardized, at least for those countries that adhere to the Berne convention:

A “derivative work” is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a “derivative work” [USC09].

Given this definition, it is important to understand how the copyright on the original work extends on the derived work. Referring to the United States Code again:

The copyright in a compilation or derivative work extends only to the material contributed by the author of such work, as distinguished from the preexisting material employed in the work, and does not imply any exclusive right in the preexisting material. The copyright in such work is independent of, and does not affect or enlarge the scope, duration, ownership, or subsistence of, any copyright protection in the preexisting material....

Subject to sections 107 through 122, the owner of copyright under this title has the exclusive rights to do and to authorize any of the following: (1) to reproduce the copyrighted work in copies...; (2) to prepare derivative works based upon the copyrighted work; (3) to distribute copies...of the copyrighted work to the public by sale or other transfer of ownership, or by rental, lease, or lending...” [USC09].

An example of derivative work, in this definition, is any original work that incorporates portions of material previously published. In order to acquire copyright on a new original work, it has to contain a *substantial amount of new material*. Minor changes to pre-existing work do not make your work liable to a new copyright submission.

The given definition of new work is somewhat weak though: there is no objective measure of “substantial amount”. And to make things even less clear, the above mentioned law does not explicitly mention software works, which display some peculiar characteristics that do not belong to arts like music and literature.

It is useful to briefly concentrate on the software context to best explain what is intended as derived work.

When coming to software and source code, several scenarios can be considered:

- **The source code is available, modifications are applied to this code.** This makes the new work a derivative work. Licenses like the GNU GPL or the Open Source Licenses, impose that you must publish such a work under the same license. Other permissive licenses, such as the BSD license, instead

impose no restrictions; you could even close the source code of the newly created software.

- **The original source code is incorporated in a (possibly larger or different) new work as-is, without modifications.** Before considering this scenario, it must be clear that this configures a mix of already licensed source code and new code. To do this legally, the license of the original already published code and the one under which the new code is released must be compatible. This could not result in an issue if the original code is released under a permissive license like the BSD one.
- **The original creation is linked by the new one.** This is what happens when a new software creation links to external libraries. In this case, it is even possible to detail two distinct ways of linking the software library: static and dynamic linking.

If for the first scenario it can be assumed with almost no doubts that the new work is to be considered a derivative work, the second and the last one are still open legal issues: can the new works considered as derivative?

In the Free Software and Open Source communities there are different positions about this issue. The two main positions are:

- an application linking against an external library is not a derivative work of the library itself
- an application linking against an external library is a derivative work of that library as the new code and the library together form one work

A work-around to this issue is offered by the Lesser General Public License:

A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License [LGP09].

The LGPL has born to the ultimate aim of allowing third parties closed source applications to link against a library. LGPL fits particularly well for libraries of routines, but it is deprecated by the GNU foundation and the FSF.

The GNU GPLv3, in its last revision, tries to clarify what can be defined as derivative work: for this license (as it was for GPLv2) this is a fundamental step to understand when and where the reciprocity obligation clause must be applied to the new software.

4.2.1.14 Derivative works

Does the license permit derivative work?

Scores:

- 0 no

- 1 yes, with some restrictions
- 2 yes

4.2.1.15 Close original code

Does the license permit to close original unmodified code merged into an own creation?

Scores:

- 0 yes
- 1 yes, with restrictions
- 2 no

4.2.1.16 Close derived code

Can a derived work be closed?

Scores:

- 0 yes
- 1 yes, with restrictions
- 2 no

4.2.1.17 Restrictions on other software

Does the license impose restrictions on other software shipped on the same media?

Scores:

- 0 yes
- 1 yes, under some terms
- 2 no

4.2.1.18 Linking from code with different licenses

Does the license permit to link from code licensed under the terms of a different license? Some licenses, like the GNU GPL, are source for forks to allow/disallow this point.

Scores:

- 0 yes
- 1 yes, with restrictions
- 2 no

4.2.1.19 Fee Charging

Does the license permit to charge a fee for the work? This corresponds to a commercial clause. Almost no license disallows a commercial (sell software) option. According to the FSF such licenses cannot be considered as Free Software licenses.

Score:

- 0 no
- 1 yes, with restrictions
- 2 yes

4.2.1.20 License popularity

Is the license popular and commonly accepted? A good acceptance by communities is an important step for license spread (and can avoid license proliferation). Three ranges of adoption were considered. Raw data were taken from the Blackduck Software open source survey.

Scores:

- 0 low popularity
- 1 medium popularity
- 2 high popularity

Software Patents**4.2.1.21 Patents awareness**

Is the license patent aware? This means that the license explicitly deals with software patents and related royalties.

Scores:

- 0 no
- 1 partially
- 2 yes

4.2.1.22 Protection against patents

Does the license establish any mechanism of protection against software patents and related royalties?

DRM**4.2.1.23 DRM awareness**

Is the license DRM aware?

Scores:

- 0 no
- 1 partially
- 2 yes

Discriminations**4.2.1.24 Discriminations against groups or people**

Does the license make any discrimination against people or groups?

- 0 yes
- 1 partially
- 2 no

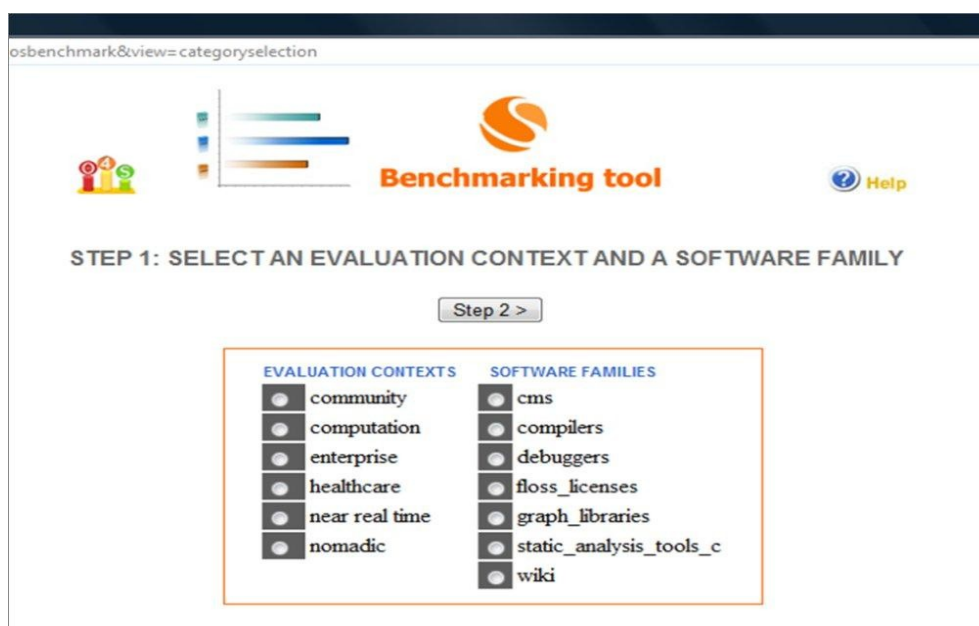
4.2.1.25 Discriminations against fields of endeavour

Does the license make any discrimination against the field of endeavour?

- 0 yes
- 1 partially
- 2 no

4.3 O4S

Considering each criteria as a requirement, the QSOS methodology was applied to licenses. This document is thus supported by all the material produced for the O4S instance that will be found at <http://www.share-project.eu>.



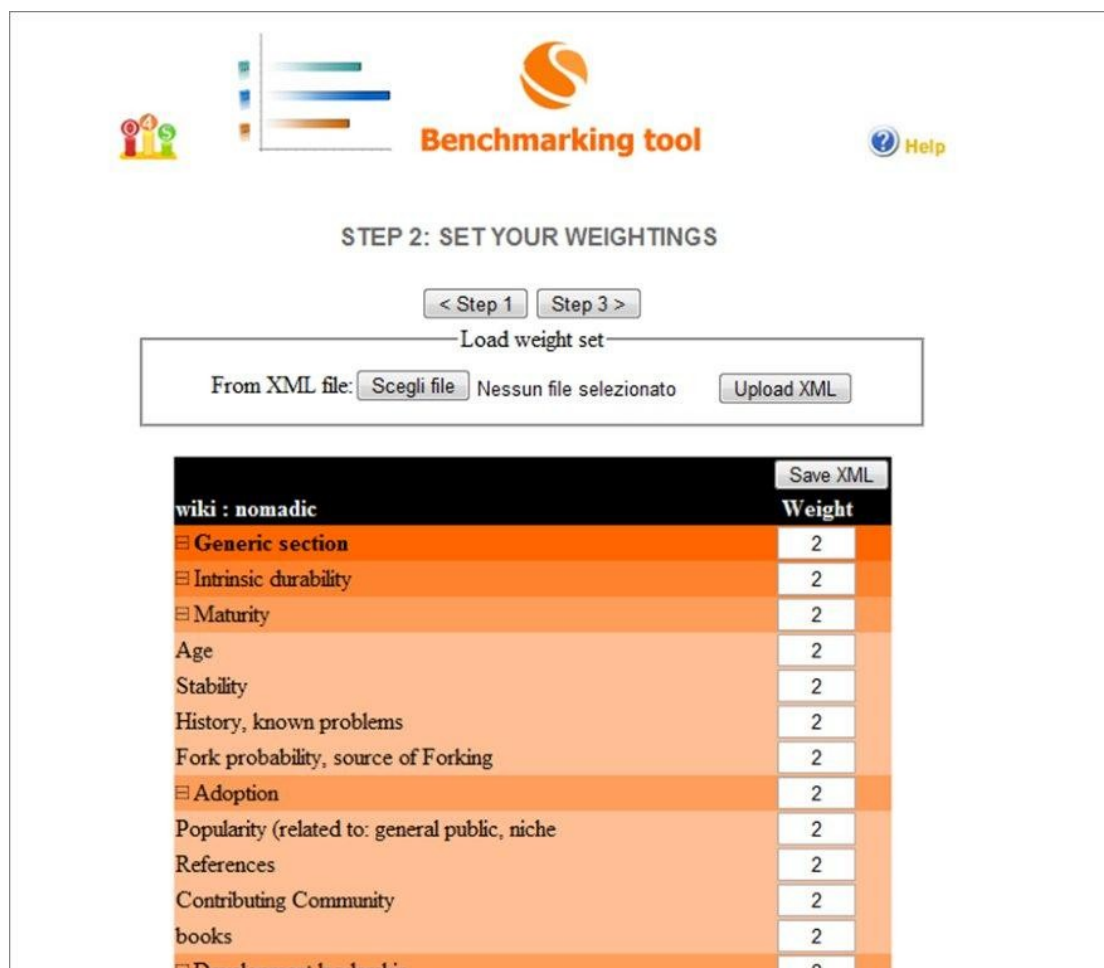
The screenshot shows the 'osbenchmark&view=categoryselection' web interface. At the top, there is a 'Benchmarking tool' logo and a 'Help' button. Below this, the instruction 'STEP 1: SELECT AN EVALUATION CONTEXT AND A SOFTWARE FAMILY' is displayed. A 'Step 2 >' button is located below the instruction. The main content area is divided into two columns: 'EVALUATION CONTEXTS' and 'SOFTWARE FAMILIES'. Each column contains a list of categories with radio buttons for selection.

EVALUATION CONTEXTS	SOFTWARE FAMILIES
<input type="radio"/> community	<input type="radio"/> cms
<input type="radio"/> computation	<input type="radio"/> compilers
<input type="radio"/> enterprise	<input type="radio"/> debuggers
<input type="radio"/> healthcare	<input type="radio"/> floss_licenses
<input type="radio"/> near real time	<input type="radio"/> graph_libraries
<input type="radio"/> nomadic	<input type="radio"/> static_analysis_tools_c
	<input type="radio"/> wiki

Illustration 7: Selection of context and family in O4S

The O4S application was developed in the scope of the SHARE project. O4S is a fork of the O3S application developed by Athos Origin. The application was redesigned to support a storage back-end so to keep memory of:

- applied weights
- applied scores



wiki : nomadic	Weight
Generic section	2
Intrinsic durability	2
Maturity	2
Age	2
Stability	2
History, known problems	2
Fork probability, source of Forking	2
Adoption	2
Popularity (related to: general public, niche	2
References	2
Contributing Community	2
books	2
Development leadership	2

Illustration 8: Weighting process

Saving this data is useful to analyse and collect statistics about the assessment: how weights changed during time or for a particular context, how scores changed for a product or for different versions of the same product.

To accomplish the assessment via the O4S web application, a template made of the above criteria was first formed. The template was then used as a base to complete the process filing an evaluation sheets with scoring for each license. For an in-depth coverage of the assessment methodology, refer to document produced as deliverable D2.1 of the SHARE project.

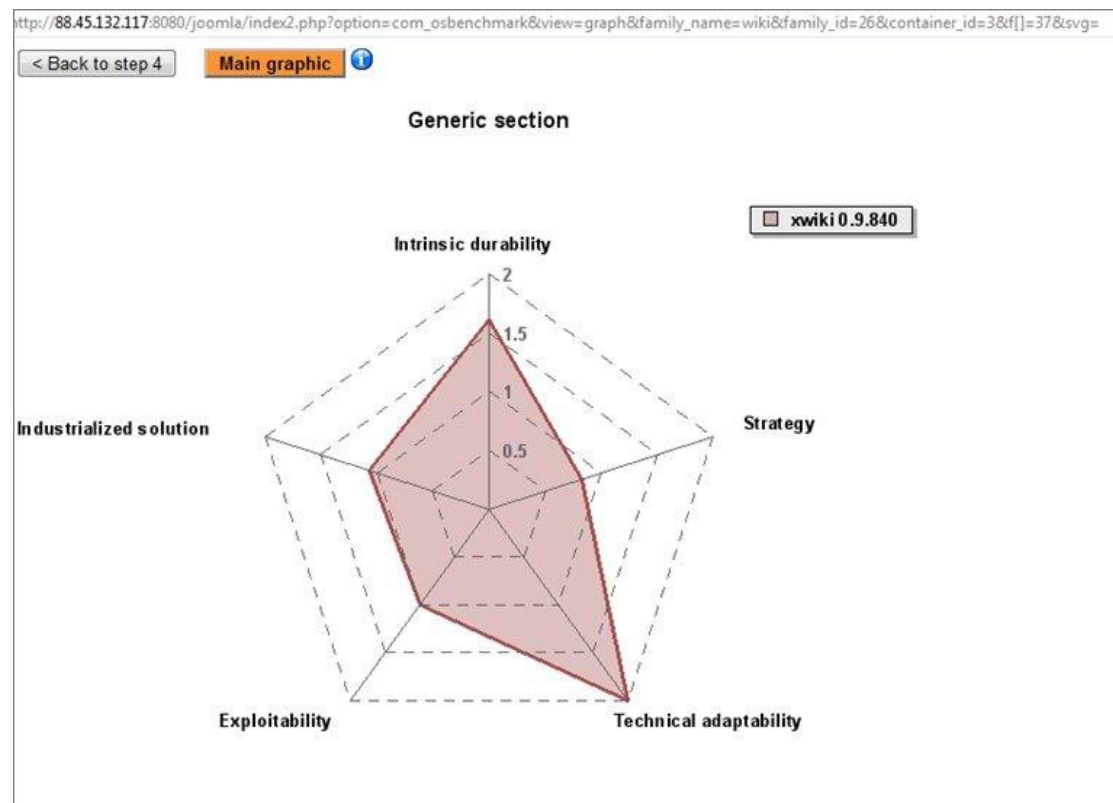


Illustration 9: O4S: radar diagram of evaluation

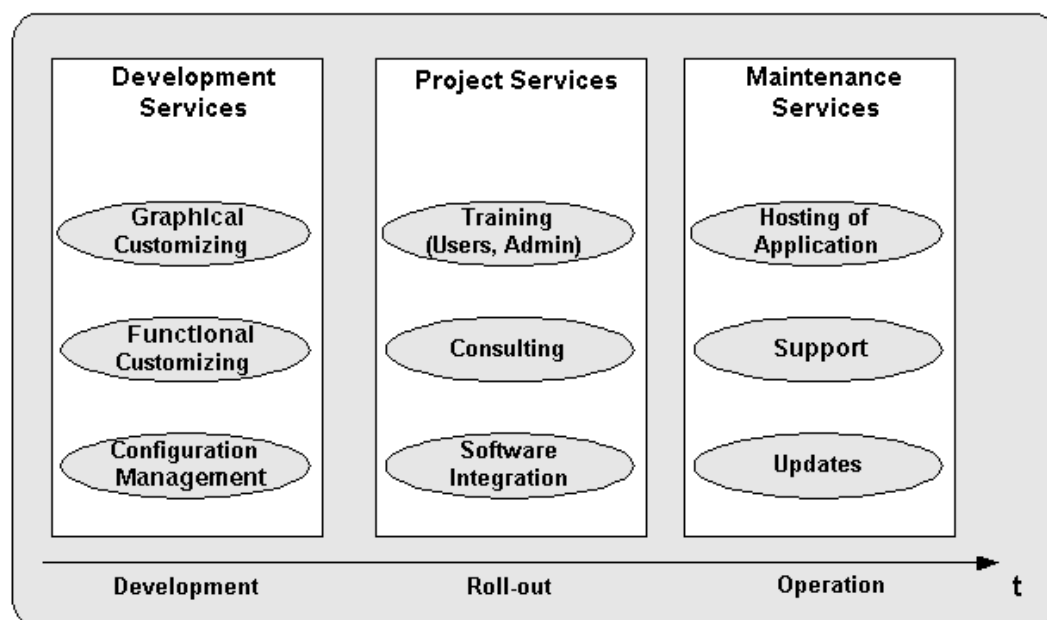
5 OS BUSINESS MODELS AND LICENSES

There is not a large amount of literature available about FLOSS business models, but one possibility is to rely on FLOSS success stories to identify them. This section will deal with some of them, but clearly many more can exist and some others are hybrid coming out by mixing between them.

- Sell technical support services
- Distribute software
- Fund the projects using donations
- Build and sell hardware and software configurations
- Release proprietary versions of the software
- Release proprietary add-ons, components
- Dual licensing software
- Offer each new version of a software only to paying customers

5.1 SELLING TECHNICAL SUPPORT SERVICES

Selling technical support services and, more generally, support services to customers is maybe the most popular and well-known business model available for FLOSS producers and distributors.



Most of FLOSS projects offer community-based support services via several communication instruments, like mailing lists, forums, web-sites. This kind of support service is usually completely free of charge, clearly. Along with it, many industries

offer “professional” support behind payment of a fee¹⁷. These are commonly referred as enterprise support services. This kind of technical support includes training for employees.

Among the industries adopting this kind of model the free and most popular is without any doubt the Red Hat Inc..

Red Hat has been involved in open source development since from the beginning of its history. As a success story, it is interesting to have a quick look at what Red Hat offers:

- subscriptions for the technical support
- training
- integration services to customers

Of interest, in recent years, Red Hat launched a new service, called Red Hat Exchange¹⁸, which is a portal where to resell free software and open source software by other software firms. “Why willing to pay for something that is free of charge?” could be the immediate question. Red Hat offers some extra software to paying people and support services, documentations. The RHX portal offers software from several FLOSS firms and this is perfectly legal, for example, when using the GPL License, which can seem the most restrictive license around. The General Public License, in fact, allows the developer to charge a fee for the software. It just requires that the source is always distributed and available (along with binaries or stand-alone):

“You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee...” [GPL91].

“For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights...” [GPL91].

There are licenses, like the Artistic license, where it is explicitly stated the difference between a distribution fee (the one permitted by the GPL) and a license fee: license fees usually characterise closed source licenses and are not admitted by most (if not the totality) of open source licenses.

“Distributor Fee” means any fee that you charge for Distributing this Package or providing support for this Package to another party. It does not mean licensing fees [ART09].

There are many points of the Artistic License where it explicitly denies the use of a licence fee.

¹⁷The use of the word “professional” in this document does not refer to the quality of the service, while at a common naming convention.

¹⁸See <http://rhx.redhat.com/> for more informations.

“..and requires that the Source form of the Modified Version, and of any works derived from it, be made freely available in that license fees are prohibited but Distributor Fees are allowed...” [ART09].

Notably, Red Hat also distributes a commercial version of the formerly free of fees Red Hat Linux, Red Hat Enterprise Linux, and a community supported one, the Fedora Linux project. The same business model is used nowadays by Novell Inc., that in recent years has acquired the SUSE Linux distribution which comes in a commercial flavour and in a free of charge one, exactly as it happens for Red Hat.

Novell and Red Hat are just two big industries among many others using this business model that is enabled by the fact that the most popular and adopted licenses do not deny a fee for the distribution of the software, which is something completely different from a license fee.

5.2 DISTRIBUTING SOFTWARE

Software distribution is another common business model. The expression “Linux distribution”, as an example, is something known even to not so experienced users. In this section we will deal with this kind of software distribution in order to show a real instance of the software distribution business model.

What's the exact meaning of “Linux distribution”?

Linux distribution usually refers to a distribution of software that includes:

- The Linux kernel. This is the core of the operating system, released under the GPLv2 license.
- A complete set of applications and user space tools. Most of them are coming from the GNU foundation project. It is clear that a lot of different licenses apply to the big number of included applications: GPLv2, GPLv3, BSD, MPL and many more.
- Home-made tools and scripts. Usually a packaging systems, boot and init scripts, system configuration tools and patches to third parties software. These are licensed under the terms of a license of choice for the project.

There are groups, such as the Free Software Foundation, that invite people to refer to Linux distributions as “GNU/Linux distributions”, given the strong presence of GNU project tools included.

It is out of the scopes of this document to take a position about the right terminology. It is of most interest, instead, underlining that both the Linux kernel and most of the userspace tools provided by the GNU project are released under the GNU GPL License (it does not matter here if it is the version 2 or 3). This license permits the user to redistribute the software and charge a fee for the redistribution, as observed in the previous section. It is important, when redistributing in binary form, that the source is always available. This does not mean that source must be attached to the binary. It just means there must be an easy way for users to obtain the source code.

The most popular Linux distributions:

- Offer a free download of the whole distribution. This includes downloading the software in binary and source forms
- Offer CDs and DVDs of the distribution charging a fee for it

Some other producers/distributors also offer particular versions of the distribution that are available only to paying customers (Red Hat, as observed, is among them). The fee gives access to source and binaries, according to the GPL License.

This is made possible by many other licenses too, even without some restrictions which characterise the GNU GPL. As an example, the BSD license

- does not limit the application of a fee
- does not ask to release modifications
- does not limit the license that can be used to redistribute a derived work

Among this point, the latter can imply that the user can redistribute under a closed source proprietary license and can impose a fee for the license, not for the distribution. More precisely, every referral here is to the modified BSD license, also known as version 3 of the BSD license:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- *Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.*
- *Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.*
- *Neither the name of the <organization> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission [BSD99].*

Just as a note, the one here cited is called modified BSD license because the historical BSD license came with four clauses, the third of which was completely removed: it asked to acknowledge the University of California Berkeley in derived work (it was referred as the “announcement clause”). The old four clauses version was not approved by the OSI and was not compatible with GPLv2. Removing the mentioned clause, allowed to solve both problems so that this three-clauses version is now OSI approved and GPLv2 compatible. Code released under this license can thus be mixed with GPL-ed code. This license is an open source license, a free software license and is very permissive. It is fundamental, once again, to understand the difference between being permissive and being an open source or free software license. Being a permissive license is not enough to be recognized as an open source license.

Characteristics that should be considered, in the context of this business model, are thus:

- possibility to distribute, charging a fee for the distribution

- possibility to redistribute modified versions and derived works (so to apply patches or fork new applications)
- compatibility (when mixing code in the same work)
- restrictions on software released under other licenses (the license that applies to one software should not bring restrictions on software licensed under other licenses when shipped on the same media)

5.3 FUNDING WITH DONATIONS

Funding the project with donations is a common way to bring money needed to support developers and further development of open source and free software projects.

This is a business model that is best suited to those communities consisting of volunteers, but it can also apply and work well with big commercial communities. Funding via donations is something regulated by national laws and this section will not deal with the necessary steps to accept donations or with taxation of funds made by donations.



Illustration 10: A view of some popular Open Source projects

As a success story, it can be useful to look at the funding campaigns launched by the Mozilla foundation. The Mozilla Organization was first started in 1998 as a parallel experimental project by America On Line/Netscape. Netscape was a closed source browser, while Mozilla, sharing part of the code base, was an open project that was started for testing purposes and not for end-users releases. Given the success of the Mozilla browser, plans changed during years: the netscape market share has progressively fall, while the Microsoft Internet Explorer browser continued to gain market shares, together with Mozilla. When AOL decided to scale down its effort on

Mozilla, the only way to fund the project was to constitute as a foundation and begin accepting donations.

During years the Mozilla foundation started many new projects and is now the producer and distributor of the popular Mozilla Firefox web browser (this is not the only project developed, but it is for sure together with Mozilla Thunderbird the most famous).

The Mozilla foundation has always used a donation business model: fundraising campaigns have been launched during years. Money earned was always used for two main activities:

- support the development
- advertising and dissemination initiatives

Volunteer donations are just one of the incoming voices. The Mozilla foundation in fact brings attention and receives donations from some leading industries to support the development of the Firefox Browser. Google Inc., for instance, funded the Firefox development with 180 millions of dollars distributed over a range of three years.

Mozilla Foundation has finally launched in 2005 a subsidiary, the Mozilla Corporation, which is a different entity (to be clearer a taxable entity) and has open the road to other business models¹⁹.

Far different from Mozilla Foundation, there are lots of volunteer-based projects that constituted as no profit organisation or foundations to accept donations (money and sometimes hardware too). Just to give an idea, the CentOS and Sabayon general purpose Linux distributions, the whole Debian project, the OpenWRT distribution for embedded devices are all popular projects, loved by users and enterprises, which live thanks to donations.

There are no particular notes about licenses and donations, but it is clear that the acceptance of the license among users can be a key factor for the widespread adoption of the software and consequently of its success. These are all factors that can have a large impact on a fund-raising campaign.

5.4 BUILD AND SELL HARDWARE AND SOFTWARE CONFIGURATIONS

It is possible to distinguish several options here:

- A software producer that resells hardware coming with the software installed
- A hardware producer selling the hardware with some software preconfigured on it

In the first category fall all those software producers that distribute their products either as a stand-alone option or pre-installing it on a given hardware solution. It is possible to separately sell services and technical support or purpose a full pack including hardware, configuration and technical support. The added value is given by:

- Installation and configuration services
- Technical supporting

¹⁹Cfr. <http://www.mozilla.org/press/mozilla-2005-08-03.html>

- Ad-hoc hardware configurations. It can often happen that the software producer have such an agreement with the hardware producer so that the first will get particular “limited” edition of the hardware.

There are no particular problems with licenses for this business model to be liable. There are no open source licenses that forbid selling the hardware with pre-installed software.

The second category is constituted by hardware producers and vendors shipping their own products with open source and free software. Such companies are gaining a big share of market: if in the beginning it was a niche market targeted at “hackers” and very experienced people, it is now clear that most of average experienced people like to get products that work well but at the same moment having a large amount of control over them. To consider a simple example, many big companies distribute networking devices with open source and free software firmwares pre-installed. This was done silently in a first stage. Now this products are more and more often advertised as “running Linux” or running open and modifiable firmwares [PER09].

This business model could rapidly become the most important one between those listed in this document. The widespread popularity of embedded devices for professional use and entertainment is a very big share of market. Some fields of endeavour:

- Networking devices: routers, managed switches, firewalls.
- Mobile devices: smart phones, PDAs, handhelds.
- Entertainment devices: audio and video players and recorders, cameras, video cameras, TV set-top boxes.

Making use of available licensing schemes, the producers can:

- Use ready-to-use software: even the more restrictive open and free license permits hardware vendors to ship their products with software. In case of a restrictive license, as the GNU GPL (versions 2 and 3), the vendors must additionally supply:
 - The source code
 - Each modification applied to GPL-ed software

It must be noted that this does not mean the vendor must ship in the same package an additional media with sources, distribution of the source code on a web-site is enough. The vendor must provide to the user all the information needed to get a copy of the source code. Of note, some vendors provide not only the source code, but a complete suite (made of compiler and cross-compiling toolchains) to recompile the software.

- Develop additional software to support the hardware. Along with modification to open sourced code, the vendor could decide to protect hardware implementation details by developing closed-source kernel components and tools and mix them with open source and free software components. This is made possible by many licenses. The most restrictive in

regard with this practice is the GNU GPL (v.2 and v.3) which forbids to close derived work and does not allow to mix closed source and GPLed code in a same *work*. As observed in previous chapter, there is a big debate around the definition of work and derived work.

Chapter 3 of this document dealt with a problem arisen in a context where this business model is deployed: the so-called *tivoization* proposes a case where the hardware producer ships the product with free GPL-ed software included and, accomplishing the terms of the license, makes the source code available. On the other side, the producer protected the software of the device against modifications implementing a DRM technology that blocks any attempt to change the original firmware. The whole Tivoization affair is analysed in the third chapter. In this section it is important to mention that such behaviour is now explicitly denied by a DRM-aware license such as the GNU GPLv3.

5.5 RELEASE PROPRIETARY VERSIONS OF THE SOFTWARE

This is a simple business model where original software is modified, features are added or bugs are fixed, and then the resulting software is released under the terms of a proprietary license. This applies to permissive open and free licenses, the BSD licenses, most notably, and many other BSD licenses, such as the Xiph.org license for instance.

5.6 RELEASE PROPRIETARY ADD-ONS, COMPONENTS

This business model implies selling a component (a plug-in or an extension) under a commercial proprietary license, charging a fee for the license and/or its distribution. The definition of “component” could be something like “a piece of a larger work”.

To approach this kind of business model, the “larger work”, the original software you are coding the extension for, must be released under a permissive license, such as the BSD, that permits to plug proprietary code into the work. Or you could use a much younger license, as the Mozilla Public License.

The MPL was created in order to allow third parties to design proprietary closed-source extensions which plug into the Mozilla browser pluggable architecture. MPL is an OSI approved license and is a Free Software license too, also referred as a weak copyleft license.

It is similar to the GNU GPL, but while the GPL denies the possibility to plug proprietary code in the context of a single executing process, the MPL denies this possibility in the scope of a single source code file: a file containing MPL-licensed code must be fully released under the MPL terms. This also means that modifications to MPL licensed code must be licensed under the same license. This is why this is considered a weak copyleft license.

Some excerpts from the Mozilla Public License:

1.7. "Larger Work"

means a work which combines Covered Code or portions thereof with code not governed by the terms of this License....

1.9. "Modifications"

means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

- *Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.*
- *Any new file that contains any part of the Original Code or previous Modifications....*

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

the licenses granted in Sections 2.2 (a) and 2.2 (b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

Notwithstanding Section 2.2 (b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor...[MPL95].

5.7 DUAL LICENSING SOFTWARE

Dual licensing is becoming a popular business model in the Open Source and Free Software context. MySQL AB is probably one of the best known open source companies and has been one of the first firms to pioneer the dual licensing model.

The concept behind dual licensing is quite easy to understand: the same code base is released under two different licenses: on one hand under a free software license, the GNU GPLv2; on the other hand under the terms of a commercial proprietary license. The first makes the code free for everyone to be downloaded and used, modified and

distributed. The latter is instead available only to paying customers. Paying customers gets guarantees that community users do not get. If the vendor accountability is a requirement, the commercial license is probably the right choice, but this is not the only key factor to go with a commercial license.

The dual licensing models may have issues. Consider the case the code is dual licensed under the terms of the GNU GPL and a commercial license. Until one owns the copyright of the source code, there are no problems with dual licensing it.

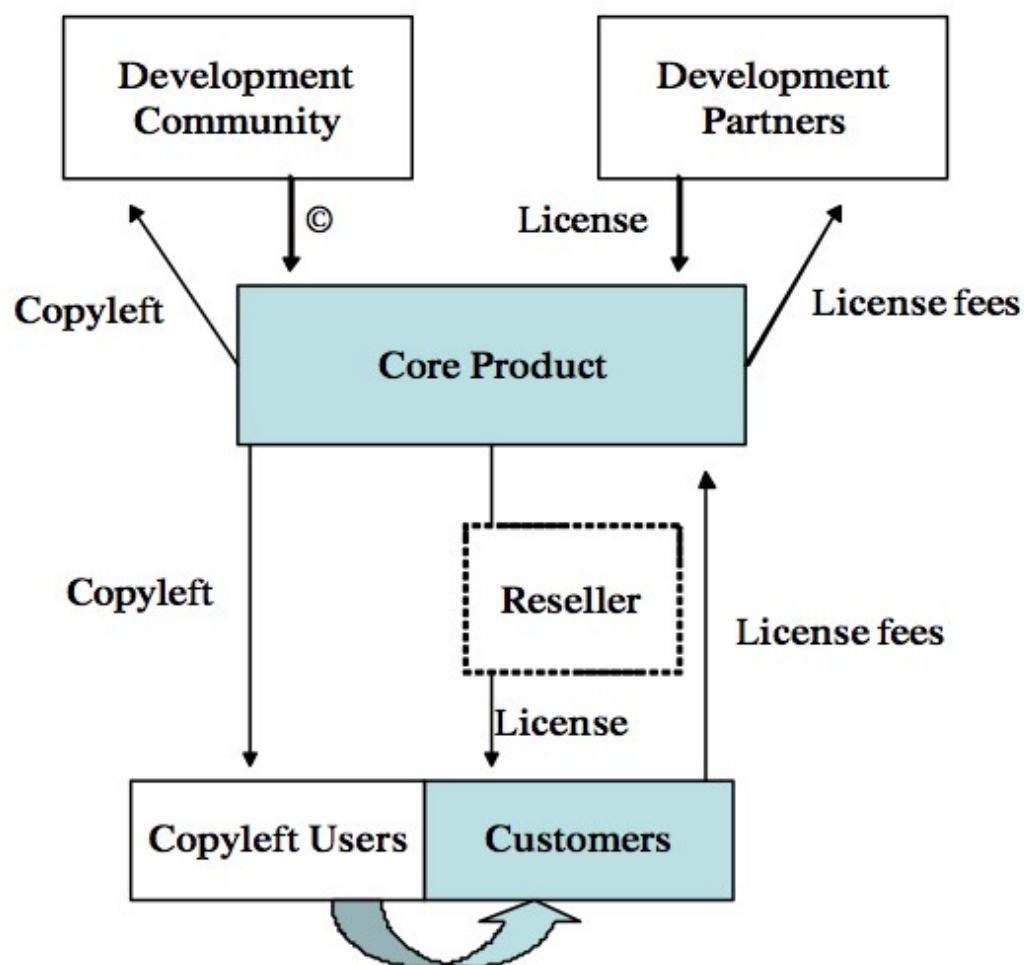


Illustration 11: Dual licensing development model

But what happens when someone, a third part, contributes some code? This contribution will be published under the terms of the GPL, the contributor is the copyright owner. Can this contribution be merged into the code and then licensed under a commercial proprietary license without any permission or revenue for the contributor? This point is still unclear and, in the doubt, contributors are required to give the copyright back to the company. This is not an unusual practice in dual licensing contexts [VAL03].

There exist licenses that are somewhat dual licensing aware; the SleepyCat license is maybe the clearest example of such licenses. In particular, it is interesting to look at the following excerpts of the license:

“The open source license for Berkeley DB permits you to use the software at no charge under the condition that if you use Berkeley DB in an application you redistribute, the complete source code for your application must be available and freely redistributable under reasonable conditions. If you do not want to release the source code for your application, you may purchase a [proprietary] license from Sleepycat Software.” [SLP99].

Thus the free license mentions that the restrictions imposed on derivative works (whose source code must be released) can be superseded by acquiring a commercial license. The Sleepycat license is clearly shaped on a particular software license (the Berkeley Database Software, formerly released under the original BSD license).

Accountability is not the only feature of the commercial license. With the commercial license, the customer has the possibility to develop customizations, apply modifications and extend the software without having to release the source code, thus making all the modifications proprietary software, this is the main reason to prefer a commercial license and this is the main source of revenue for products such as the Berkeley DataBase and the MySQL. There are many products nowadays that embed databases. The choice of such a product when selecting MySQL or BDB as a database can be:

- Using the free software license and release the source code of the product
- Paying for the commercial license and keep the source code of the product closed and proprietary

5.8 OFFER EACH NEW VERSION OF A SOFTWARE ONLY TO PAYING CUSTOMERS

A viable model is offering a new software product, released under a free and open license, for a limited range of time behind the payment of a fee. This can be worked out in two ways:

- Customers can subscribe a plan (pay for a membership) with which they get products as soon as they are released. The membership can expire after a certain number of months. The products, after some months from the release, will be widespread and accessible to anyone (even not paying users).
- Customers can buy the product as soon as it is released. The same product will be distributed for free after some time.

The key factor for this business model, which is used by a moderate number of projects, is to establish a fair price for the product: customers would not be leased to pay a big price for something that can be get for free after some time, they would be rather wait for the free distribution.

Almost any FLOSS license that allows charging a fee for the distribution should fit for this business model.

6 LICENSES BENCHMARKING RESULTS

In this section we propose an inventory of Open Source licenses. It contains most of the licenses approved as Open Source Licenses by the Open Source Initiative. A full list of the OS license can be found on the OSI web site²⁰.

All those licenses that are considered obsoleted and rarely used were not included in this list; all those licenses that were written and used for special purposes (sometimes even for a single application or a suite of tools) and that were found out not to be interesting in the scope of the OS licensing analysis are not analogously included. Finally, many of these licenses were often written as variants or exceptions of other popular licenses. In particular both the BSD and the GNU GPL licenses suffered tenth of forks by several software projects.

As stated by the OSI "open source doesn't just mean access to the source code" and licenses should comply to other principles too. You can read a quick and simple resume of the characteristic that a license should have on the open source definition page at OSI.

Each of the following licenses was submitted by proposers and passed through an approval process defined and guided by the OSI. Entities interested in submitting a new license can consult the OSI board to improve their license before it is reviewed in the formal approval process.

We compiled a simple form for every license following these guidelines:

- What you can do and what you can't do with the license. In particular, we looked at the three main aspects: copy, redistribution and modifications.
- Several characteristics that, according to us, are fundamental for the popularity and the adoption of a license:
 - Is the license copyleft? We use the "copyleft" word to mean an use of the copyright laws that allows people to copy, modify and redistribute works (software, music, documentation) granting that this freedom is preserved in subsequent modified versions.
 - GPL compatibility: recent studies show that the GNU GPLv2 license is by far the most popular and used license in the Open Source scenario. Using a GPL compatible license allows you to mix code and contributes under your license with GPL-licensed ones. Except were particularly stated, the compatibility is to be intended with GPLv2 and not with GPLv3.
 - Linking code from different licenses: some licenses, prominently the GNU GPL, see the linking of GPL-ed code from other licenses as a major problem. Some other licenses permit this practice. This is an important criterion when choosing a license: how my code (eg. library code) can be linked by third parties.

²⁰The site is at <http://www.opensource.org>

- Estimated number of projects using the license. This criteria is based on the work of Blackduck²¹ software. Blackduck has been conducting a long and accurate work on FLOSS and on FLOSS licenses usage. The adoption rate was taken consulting their top 20 OS licenses list, which is updated on a daily base. Three ranges of values were individuated:
 - **High:** licenses ranking from position 1 to 6
 - **Medium:** licenses ranking from position 7 to 11
 - **Low:** licenses falling beyond position 11

Just to get an idea of the magnitude order of the number of softwares examined by Blackduck, consider that there are about 3000 softwares licensed under GPLv3 and LGPLv3, which translates approximately into a 2% share in their usage table²².

- Main projects using the license. This list contains the most popular projects using the license itself. This gives an idea in terms of end-users usage. For instance: the Mozilla Public License is adopted only by a 0.8% of projects, but the user base of the browser is estimated into hundreds of millions of unities.

6.1 LICENSES IDENTITY CARDS

For each license an identity card was compiled. This task was accomplished before the whole evaluation process had begun: each card tries to facilitate the process of attributing scores to criteria, gathering information about each license and organizing them in an easy to read table.

Accomplishing an analysis of applicable FLOSS licenses can be a very hard task if there is a lack of facilitation, due to the technical legal language used for their redaction. Furthermore, understanding the legal language is a necessary but not sufficient condition to have a good comprehension of how a license works. A basic knowledge of national laws should be the base to exercise license enforcement.

6.1.1 Academic Free License 3.0 (AFL 3.0)

License Name	Academic Free License 3.0 (AFL 3.0)
Copy	It is possible to copy the original work either alone or as part of a collective work
Modifications	It is possible to translate, adapt, alter, transform, modify, or arrange the original work, thereby creating derivative works based upon the original work. Neither the names of Licensor, nor the names of any contributors to the Original Work, nor any of their trademarks or service marks, may be used to endorse or promote products derived from this Original Work without express prior

²¹See <http://www.blackducksoftware.com/oss> for more informations about Blackduck FLOSS knowledge base.

²²Data acquired on September 2008.

	permission of the Licensor. You must retain, in the Source Code of any Derivative Works that You create, all copyright, patent, or trademark notices from the Source Code of the Original Work, as well as any notices of licensing and any descriptive text identified therein as an "Attribution Notice."
Redistribution	It is possible to distribute or communicate copies of the original work and derivative works to the public, under any license that does not contradict the terms and conditions of the Academic Free License.
Copyleft	No
GPLv2-Compat	No (but not AFL versions 1.2 and 2.1)
Linking from code with different license	Yes
Notes	Allowed modification to the license. Licensor grants You a worldwide, royalty-free, non-exclusive, sublicensable license, under patent claims owned or controlled by the Licensor that are embodied in the Original Work as furnished by the Licensor, for the duration of the patents, to make, use, sell, offer for sale, have made, and import the Original Work and Derivative Works.
Estimated number of projects using the license	Low
Main projects using the license	CodeBeamer, Vanilla, Jaggregate

6.1.2 Adaptive public license

License Name	Adaptive public license
Copy	It is possible to copy and use the initial work.
Modifications	It is possible to reproduce and to prepare derivative works, the derivative work must contain a file documenting the changes.
Redistribution	It is possible to distribute, and sublicense any derivative works, in Source Code and executable form, either with other modifications, on an unmodified basis, or as part of a larger work when a distributor makes the licensed work, or any portion thereof, available to any person in source code form, a copy of this license must be included with each copy of the source code. The recipient shall have no obligation to distribute, in either source code or executable form, any such internal use modifications made by him. The name of a distributor may not be used by any other distributor to endorse or promote the licensed work

	or products derived from the licensed work, without prior written permission.
Copyleft	Weak Copyleft
GPLv2-Compat	NN
Linking from code with different license	NN
Notes	It is an adaptable template license, in fact it must exhibit an attached accompanying the license to determine the specific adaptive features applicable it. The initial contributor may publish revised and/or new versions of the license from time to time. Each version will be given a distinguishing version number. The license has an optional attribution clause.
Estimated number of projects using the license	Low
Main projects using the license	Mamook, JMd5Sum

6.1.3 Affero GNU GPLv2

License Name	Affero GNU/GPL
Copy	See The GNU General Public License (GPLv2)
Modifications	See The GNU General Public License (GPLv2)
Redistribution	See The GNU General Public License (GPLv2). Additionally the AGPL requires that the full source code of the networking/web application is fully available. It is explicitly requested not to remove the AGPL notice in the source files.
Copyleft	Yes
GPLv2-Compat	Yes
Linking from code with different license	No
Notes	This license is derived from the GNU GPL. It has been crafted specifically for networking software, web applications and software used on a network.

Estimated number of projects using the license	Low
Main projects using the license	?

6.1.4 Apache License v2.0

License Name	Apache License v.2.0
Copy	It allows the user of the software the freedom to use the product for any purpose.
Modifications	It is possible to modify the product.
Redistribution	It is possible to distribute the product and the modified versions of it. The Apache License does not require modified versions of the software to be distributed using the same license nor even that it be distributed as free/open-source software. The Apache license only requires that a notice is kept informing recipients that Apache licensed code has been used. Thus, in contrast to copyleft licenses, recipients of modified versions of Apache licensed code do not necessarily also get the above freedoms.
Copyleft	No
GPLv2-Compat	Only GPLv3
Linking from code with different license	Yes
Notes	Two files that must be put at the top directory of redistributed software packages: 1) A copy of the license itself 2) A "notice" text document listing the names of licensed libraries used, together with their developers.
Estimated number of projects using the license	High
Main projects using the license	Apache, Apache SpamAssassin, Sequoia, Apache Tomcat, Apache Ant, phpOpenTracker, Apache Cayenne, Tomcat Status Widget, Java Parallel Processing Framework and many others.

6.1.5 Artistic License 2.0

License Name	Artistic License 2.0
Copy	It allows the user to copy the product
Modifications	You are permitted to use the Standard Version and create and use Modified Versions for any purpose without restriction, provided that you do not Distribute the Modified Version.
Redistribution	<p>You may Distribute verbatim copies of the Source form of the Standard Version of this Package in any medium without restriction, either gratis or for a Distributor Fee, provided that you duplicate all of the original copyright notices and associated disclaimers. At your discretion, such verbatim copies may or may not include a Compiled form of the Package. You may Distribute your Modified Version as Source (either gratis or for a Distributor Fee, and with or without a Compiled form of the Modified Version) provided that you clearly document how it differs from the Standard Version, including, but not limited to, documenting any non-standard features, executables, or modules, and provided that you do at least ONE of the following:</p> <p>(a) make the Modified Version available to the Copyright Holder of the Standard Version, under the Original License, so that the Copyright Holder may include your modifications in the Standard Version. (b) ensure that installation of your Modified Version does not prevent the user installing or running the Standard Version. In addition, the Modified Version must bear a name that is different from the name of the Standard Version. (c) allow anyone who receives a copy of the Modified Version to make the Source form of the Modified Version available to others under (i) the Original License or (ii) a license that permits the licensee to freely copy, modify and redistribute the Modified Version using the same licensing terms that apply to the copy that the licensee received, and requires that the Source form of the Modified Version, and of any works derived from it, be made freely available in that license fees are prohibited but Distributor Fees are allowed. Distribution of Compiled Forms of the Standard Version or Modified Versions without the Source</p>
Copyleft	FSF holds that it is too ambiguous to be a proper copyleft license.
GPLv2-Compat	Compatible with the GPL thanks to the relicensing option in section 4(c)
Linking from code with different license	?
Notes	You are always permitted to make arrangements wholly outside of this license directly with the Copyright Holder of a given Package
Estimated number of projects using the license	Medium

Main projects using the license	Most of the Perl implementation
--	---------------------------------

6.1.6 BSD License

License Name	BSD License
Copy	You can freely copy the software
Modifications	It is possible to modify the software freely.
Redistribution	Redistributions of source code must retain the copyright notice, the list of conditions and the disclaimer. Redistributions in binary form must reproduce the copyright notice, the list of conditions and the disclaimer in the documentation and/or other materials provided with the distribution.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	It is considered obsolete but it is really popular.
Estimated number of projects using the license	High.
Main projects using the license	FreeBSD Operating System, Xiph.org projects (in both cases with modifications)

6.1.7 Common Public License

License Name	Common Public License
Copy	It allows the user to copy the product
Modifications	It is possible to modify the code, but parts of code infringing own patents cannot be added. In this latter case, the developer must grant to recipients that the code is royalty-free. Modified and added source code must be made available to others.

Redistribution	It is possible to redistribute the software in source and binary forms. In the latter case you can avoid releasing the source, but you must attach a notice explaining where and how to get the source code from you.
Copyleft	Yes
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	CPL is considered by IBM "essentially the next version of the IBM Public License". The Eclipse Public License is based on the CPL.
Estimated number of projects using the license	Medium
Main projects using the license	

6.1.8 Eclipse Public License

License Name	Eclipse Public License
Copy	You can freely copy the software
Modifications	It is possible to modify the software, derivative works are allowed. Developers making changes or adding code can release their own portions of code under any license they like, but EPL-licensed code must stay EPL-ed. This means it is possible to ship proprietary software including EPL-licensed code.
Redistribution	You can redistribute the work, but EPL requires that "anyone distributing the work grant every recipient a license to any patents that they might hold that cover the modifications they have made".
Copyleft	No
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	This license is based on the CPL license v. 1.0. Nokia has publicly stated that will

	use this license to release the code of the recently acquired Symbian Os.
Estimated number of projects using the license	Low
Main projects using the license	?

6.1.9 GNU GPLv2

License Name	GNU General Public License
Copy	You can freely copy the software
Modifications	It is possible to modify the software freely. All changes to the source code must be released under the GPL license. It is not allowed to relicense code under a more restrictive license.
Redistribution	It is possible to redistribute the software in binary and source forms. Source code must always be available. Redistribution can happen either for free or behind a charge for the recipients. Dual licensing and mixed licensing is allowed as far as the chosen license is GPL-compatible.
Copyleft	Yes
GPLv2-Compat	Yes
Linking from code with different license	No
Notes	The General Public License is by far the most used license for free software. It is considered a "strong copyleft" license. The GNU GPL is recommended as a best practice by the OSI.
Estimated number of projects using the license	High
Main projects using the license	Linux Kernel, Gnome project, Kde Project, MySQL..

6.1.10 GNU LGPLv2

License Name	GNU Lesser General Public License
Copy	See The GNU General Public License (GPLv2)
Modifications	See The GNU General Public License (GPLv2)
Redistribution	See The GNU General Public License (GPLv2)
Copyleft	Yes
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	This license is a GPL License allowing linking of the software from proprietary software or more generally from software released under a non (L)GPL license. Thus LGPL is less restrictive and is mainly used by libraries. The Free Software Foundation discourage the use of this license for new licenses and encourage the relicensing of LGPL libraries under the GPL.
Estimated number of projects using the license	High
Main projects using the license	Gstreamer framework.

6.1.11 GNU GPLv3

License Name	GNU General Public License Version 3
Copy	See The GNU General Public License (GPLv2)
Modifications	See The GNU General Public License (GPLv2)
Redistribution	See The GNU General Public License (GPLv2)
Copyleft	Yes
GPLv2-Compat	No

Linking from code with different license	No
Notes	This is the third revision of the GPL License. The Free Software Foundation encourages developers releasing new software under this license. There is big debating around GPLv3 as it introduces some restrictions to the use of DRM (Digital Rights Management) in the software. Nonetheless some other restrictions about software patenting were introduced (you can have a look here to understand the new revision of the license). FSF explains these new restrictions as measures taken to preserve freedom of free software. Part of the vast Open Source community (including Linux Kernel Developers) commented against the adoption of GPLv3.
Estimated number of projects using the license	Medium
Main projects using the license	

6.1.12 GNU LGPLv3

License Name	GNU Lesser General Public License Version 3
Copy	See The GNU General Public License (GPLv3)
Modifications	See The GNU General Public License (GPLv3)
Redistribution	See The GNU General Public License (GPLv3)
Copyleft	Yes
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	The LGPLv3 differentiate from GPLv3 in the way it allows LGPL-licensed code to be linked from proprietary software or not (L)GPL-ed released software.
Estimated number of projects using the license	Medium

Main projects using the license	?
--	---

6.1.13 Microsoft Public License

License Name	Microsoft Public License
Copy	It is possible to modify the software freely.
Modifications	It is possible to modify the software freely.
Redistribution	It is possible to redistribute the software. A copy of the license must be included with the software.
Copyleft	Weak copyleft
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	None.
Estimated number of projects using the license	Medium
Main projects using the license	?

6.1.14 Microsoft Reciprocal License

License Name	Microsoft Reciprocal License
Copy	It is possible to copy the software.
Modifications	It is possible to modify the software. Modified sources must be included and retain the MS Reciprocal license.
Redistribution	It is possible to redistribute the software. Derivatives works allowed. Binary distribution allowed under own terms for completely newly created works.

Copyleft	Weak copyleft
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	It is also referred to as "shared source" as the license was written by Microsoft to share relevant parts of code with governments, enterprises and academic institutions.
Estimated number of projects using the license	Medium.
Main projects using the license	Parts of the Windows Operating System, parts of Windows Studio 2005.

6.1.15 MIT License

License Name	MIT License
Copy	It is possible to copy the software.
Modifications	It is possible to modify the software freely, the copyright notice must be retained.
Redistribution	It is possible to redistribute the software. As above, the copyright notice must be retained.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	This license is marked as obsoleted but is still very used by many projects. It is also known as the X11 License (as it was first written for the X Window System).
Estimated number of projects using the license	High

Main projects using the license	X11, Ruby on Rails, Ncurses libraries, Expat, Fluxbox, Haiku OS
--	---

6.1.16 Mozilla Public License

License Name	Mozilla Public License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely. Changes to the source code must be released under the MPL.
Redistribution	It is possible to redistribute the software in binary and source forms. Code under the MPL can be combined with proprietary code but not in the scope of the same file.
Copyleft	Weak Copyleft
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	The license is recommended as a best practices by the OSI.
Estimated number of projects using the license	Low.
Main projects using the license	Mozilla Browser, Firefox Browser, Thunderbird, Sun Solaris, Adobe Flex.

6.1.17 Nokia Open Source License

License Name	Nokia Open Source License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely, but changes must be made available for a defined range of time.
Redistribution	It is possible to redistribute the software. People cannot be charged for

	distribution.
Copyleft	Semi-Copyleft
GPLv2-Compat	No
Linking from code with different license	?
Notes	Symbian OS will NOT be released under this license. The Eclipse Open Source license will be used instead.
Estimated number of projects using the license	Low.
Main projects using the license	?

6.1.18 Open Software License

License Name	Open Software License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely, derivative works are allowed.
Redistribution	It is possible to redistribute the software.
Copyleft	Yes
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	The license is recommended as a best practice by the OSI
Estimated number of projects using the license	Medium
Main projects	ImageMagick, Qemu, Globus Toolkit

using the license	
--------------------------	--

6.1.19 Python Source Foundation License

License Name	Python Source Foundation License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely and derivative works are allowed. You are not forced to make the source code available.
Redistribution	It is possible to redistribute the software.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	None.
Estimated number of projects using the license	Low
Main projects using the license	Python, Scintilla, Roundup Tracker

6.1.20 QT Public License

License Name	QT Public License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely. You must not remove original copyright notices.
Redistribution	It is possible to redistribute the software in binary and source forms.
Copyleft	No

GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	None.
Estimated number of projects using the license	Low
Main projects using the license	QT framework

6.1.21 SleepyCat License

License Name	SleepyCat License
Copy	You can freely copy the software
Modifications	It is possible to modify the software freely.
Redistribution	It is possible to redistribute the software. The full source code of the distribution must be available at a cost which not greater than the distribution itself plus a nominal fee.
Copyleft	Yes
GPLv2-Compat	No
Linking from code with different license	Yes
Notes	None.
Estimated number of projects using the license	Low
Main projects using the license	BerkeleyDB, BerkeleyDB Java Edition

6.1.22 Sun Public License

License Name	Sun Public License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely, but changes must be made available under the terms of the license.
Redistribution	Redistribution permitted under license.
Copyleft	No
GPLv2-Compat	No
Linking from code with different license	?
Notes	None.
Estimated number of projects using the license	Medium
Main projects using the license	Sun Solaris.

6.1.23 Sybase Open Watcom Public License

License Name	Sybase Open Watcom License
Copy	Permitted solely for internal research or personal use. You must retain the copyright and the license notice.
Modifications	It is possible to modify the software. The source code of each modification must be publicly released under the terms of this license.
Redistribution	It is permitted to redistribute in source and binary form. Modified binary's source must be released under above terms.
Copyleft	Yes

GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	None.
Estimated number of projects using the license	Low
Main projects using the license	OpenWatcom C/C++ compiler.

6.1.24 Vovida Software license

License Name	Vovida Software License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software. You cannot use the name “VOCAL” in derived works.
Redistribution	It is possible to redistribute the software in source and binary forms. You must retain the copyright notice, the list of conditions and the disclaimer.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	This license has been marked as obsoleted.
Estimated number of projects using the license	Low
Main projects using the license	Vovida Open Communication Application Library

6.1.25 W3C License

License Name	W3C License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely, changes must be plainly marked and you must include the full text of the license notice.
Redistribution	It is possible to redistribute the software. You must include the full text of the license in every file.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	None
Estimated number of projects using the license	Low
Main projects using the license	Amaya

6.1.26 WxWindows License

License Name	WxWindows License
Copy	You can freely copy the software.
Modifications	Modified files must carry a prominent notice stating you changed it. You must include the date of any change.
Redistribution	It is possible to redistribute the software in source and binary forms. Binary redistribution is permitted under own terms.
Copyleft	No
GPLv2-Compat	Yes

Linking from code with different license	Yes
Notes	It is basically a variant of the LGPL license.
Estimated number of projects using the license	Low
Main projects using the license	wxWidgets, wxGTK, wxWindows.

6.1.27 X.net License

License Name	X.net License
Copy	You can freely copy the software.
Modifications	You can freely modify the software
Redistribution	It is possible to redistribute in binary and source forms but the license notice must be retained.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	Yes
Notes	The license has been deprecated by the author
Estimated number of projects using the license	Low
Main projects using the license	?

6.1.28 Zlib License

License Name	Zlib/png License
Copy	You can freely copy the software.
Modifications	It is possible to modify the software freely, but changes must be plainly marked.
Redistribution	It is possible to redistribute the software. You cannot claim you wrote the original software. In a source redistribution you must not remove or alter the license notice.
Copyleft	No
GPLv2-Compat	Yes
Linking from code with different license	?
Notes	None
Estimated number of projects using the license	Low
Main projects using the license	Zlib, libpng

6.2 RESULTS

Full results of the benchmarking process were made available through the O4S instance that will be available at <http://www.share-project.eu>. They will be in the form of a comparative matrix and/or a graphical radar diagram. In this section there is just a brief summary and licensing best practices are suggested.

First of all, it must be considered that the whole discussion about licenses is controversial and probably each license would need an in-depth dedicated document. License analysis is made even more difficult by introducing factors like patents and DRM. Doing a solely technical review is to an extent not possible because ethical issues can arise. “Intellectual Property” is one of the more debated points, for example.

Use an Open Source or Free license. This whole document should suggest you that using a license that conforms to the Open Source definition or the four freedoms identified by the Free Software Foundation is a good practice.

Consult lawyers. How societies deal with licensing? When there is the need to have a legal knowledge base on licenses, there is no other way than hiring one or more lawyers. This is the first best practice for societies that can afford the choice. The alternative is to rely on licenses that have already “passed” a judge in court. All of the most popular licenses do so. Furthermore, help yourself reading:

- License text and annotated text when available
- Use cases
- Mailing lists: both OSI and FSF have lists dedicated to discussing licenses and terminology that can lead to ambiguity

Make yourself or FSF as the copyright holder. This depends on what license you plan to use. The Free Software Foundation let you choose among holding the copyright or giving it to FSF itself. The second choice is preferred, as this will let you use FSF lawyers in case of license infringement. The FSF will probably not accept this copyright unless you are choosing the GNU GPL license.

Do you need to protect from patents? If patents are a problem you should choose a patent that is patent-aware and that protects you from being sued for patent infringement. Using GPLv2 implicitly means that the licensor is giving up patents, in the way that it will not sue anyone using the ideas patented and will not ask for royalties over the patents. There are other licenses that offer the same grade of protection from software patents and just one license that does a little step ahead in protection from patents, being GPLv3. Citing Richard Stallman on why upgrading to GNU GPLv3:

“...Another threat that GPLv3 resists is that of patent deals like the Novell-Microsoft deal. Microsoft wants to use its thousands of patents to make GNU/Linux users pay Microsoft for the privilege, and made this deal to try to get that. The deal offers Novell's customers rather limited protection from Microsoft patents.

Microsoft made a few mistakes in the Novell-Microsoft deal, and GPLv3 is designed to turn them against Microsoft, extending that limited patent protection to the whole community. In order to take advantage of this, programs need to use GPLv3.

Microsoft's lawyers are not stupid, and next time they may manage to avoid those mistakes. GPLv3 therefore says they don't get a “next time”. Releasing a program under GPL version 3 protects it from Microsoft's future attempts to make redistributors collect Microsoft royalties from the program's users.

GPLv3 also provides for explicit patent protection of the users from the program's contributors and redistributors. With GPLv2, users rely on an implicit patent license to

make sure that the company which provided them a copy won't sue them, or the people they redistribute copies to, for patent infringement.

The explicit patent license in GPLv3 does not go as far as we might have liked. Ideally, we would make everyone who redistributes GPL-covered code surrender all software patents, along with everyone who does not redistribute GPL-covered code. Software patents are a vicious and absurd system that puts all software developers in danger of being sued by companies they have never heard of, as well as by all the megacorporations in the field. Large programs typically combine thousands of ideas, so it is no surprise if they implement ideas covered by hundreds of patents. Megacorporations collect thousands of patents, and use those patents to bully smaller developers. Patents already obstruct free software development.

The only way to make software development safe is to abolish software patents, and we aim to achieve this someday. But we cannot do this through a software license. Any program, free or not, can be killed by a software patent in the hands of an unrelated party, and the program's license cannot prevent that. Only court decisions or changes in patent law can make software development safe from patents. If we tried to do this with GPLv3, it would fail.

Therefore, GPLv3 seeks to limit and channel the danger. In particular, we have tried to save free software from a fate worse than death: to be made effectively proprietary, through patents. The explicit patent license of GPLv3 makes sure companies that use the GPL to give users the four freedoms cannot turn around and use their patents to tell some users "That doesn't include you." It also stops them from colluding with other patent holders to do this..." [RMS08].

Do you need to protect from DRM? GNU GPLv3 is the license of choice if you want to protect by DRM technologies too. If your software is going to be included in an embedded device and you don't want hardware vendors to limit modifications of your software, then your choice can only be GPLv3, as no other license protects you from this kind of threat.

Avoid license proliferation. If you are going to choose a license and hire lawyers to do a license analysis, it is important to ask them to choose an existing license and not to write a new license: license proliferation is a real problem and makes license scouting a nightmare. With every probability there is an already existing license that will fit your needs. Look first at widespread licenses and then, if you did not find the one that fit your needs, look at not so popular licenses.

OSI suggests some best practice licenses chosen for quality:

- OSL and GPLv2 if you want a copyleft license
- Academic Free License if you are considering a non-copyleft license
- Mozilla Public License if you are considering a weak copyleft license [RAY02]

The FSF suggests as a best practice:

- the GNU GPLv3 license

- upgrading from GPLv2 to GPLv3

Although the FSF is also publisher of the LGPL (version 2 and 3), use of this license is discouraged and deprecated [LGP07].

Analyse success stories. In your search of a license, studying success stories can be useful. Look at the business model and at the corresponding licensing schemes available to follow a given business model.

Bibliography

[WIP08]: WIPO, *Copyright and related rights - Frequently Asked Questions*, 2008, <http://www.wipo.int/copyright/en/faq/>

[MVE08]: Microsoft Corporation, *Microsoft Windows Vista EULA*, Microsoft Corporation, http://download.microsoft.com/documents/useterms/Windows%20Vista_Ultimate_English_36d0fe99-75e4-4875-8153-889cf5105718.pdf

RMS98: Richard Stallman, *Open Sources: Voices from the Open Source Revolution*, 1998

[FSF08]: Free Software Foundation, *Free Software Philosophy*, 2008, <http://www.fsf.org/about/what-is-free-software>

[OSD08]: Open Source Initiative, *The Open Source Definition*, 2008, <http://www.opensource.org/docs/osd>

[GFD08]: Gnu Project, *Free Software Definition*, 2008, <http://www.gnu.org/philosophy/free-sw.html>

[GCL08]: Free Software Foundation, *Copyleft*, 2008, <http://www.gnu.org/copyleft/>

[MOG08]: Moglen E., *Why assign the copyright?*, 2008, <http://www.gnu.org/copyleft/why-assign.html>

EPO09: European Patent Office, *European Patent Convention*, 2009

[DBR04]: Deutsche Bank Research, *Current issues: more growth for Germany*, 2004

[NSP06]: Stop Software Patents Petition, *Examples of patents accepted by the EU*, 2006, <http://stopsoftwarepatents.eu/>

[ORI06]: O'Riordan Ciaran, *Tivoisation explained - implementation and harms*, 2006, <http://blogs.fsfe.org/ciaran/?p=69>

[FIS05]: Ken Fisher, *On Windows Vista, DRM, and new monitors*, 2005, <http://arstechnica.com/old/content/2005/08/hdcp-vista.ars>

[SMI08]: Brett Smith, *A Quick Guide to GPLv3*, 2008, <http://www.gnu.org/licenses/quick-guide-gplv3.pdf>

USC09: Office of the Law Revision Counsel, *United States Code*, 2009

[LGP09]: Free Software Foundation, *LGPL License*, Free Software Foundaton, <http://www.gnu.org/copyleft/lesser.html>

[GPL91]: The Free Software Foundation, *General Public License v2*, 1991, <http://www.opensource.org/licenses/gpl-2.0.php>

[ART09]: The Perl Foundation, *Artistic License*, The Perl Foundation, <http://www.opensource.org/licenses/artistic-license-2.0.php>

[BSD99]: Regents of The University of California, *Modified BSD License (BSD License version 3)*, 1999, <http://www.opensource.org/licenses/bsd-license.php>

[PER09]: Bruce Perens, *Combining GPL and Proprietary Software*, 2009, <http://itmanagement.earthweb.com/osrc/artcle.php/3801396/Bruce-Perens->

Combining-GPL-and-Proprietary-Software.htm

[MPL95]: Mozilla Foundation, *Mozilla Public License 1.1*, 1995, <http://www.mozilla.org/MPL/MPL-1.1-annotated.html>

[VAL03]: Mikko Välimäki, *Dual Licensing in Open Source Software Industry*, 2003

[SLP99]: Sleepycat Software, *Sleepycat License*, 1999

[RMS08]: Richard Stallman, *Why Upgrade to GPLv3*, 2008

[RAY02]: Eric Raymond, Catherine Olanich Raymond, *Licensing HOWTO*, 2002, <http://www.gnu.org/licenses/why-not-lgpl.html>

[LGP07]: Free Software Foundation, *Why you shouldn't use the Lesser GPL for your next library*, 2007, <http://www.gnu.org/licenses/why-not-lgpl.html>