NTNU
Innovation and Creativity

# Open Source Software: critical review of scientific literature and other sources

**Marc Querol del Amo**

Master of Science in Computer Science
Submission date: June 2007
Supervisor: Maria Letizia Jaccheri, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Open source software has received enormous attention by practitioner, research, and education communities. Different actors have different goals and even different understanding of what the "open source phenomen" means. In this work the candidate will have to provide a critical roadmap of the literature, printed and online, about open source software.

Assignment given: 01. February 2007
Supervisor: Maria Letizia Jaccheri, IDI

# Abstract

This thesis presents the results of a survey of Open Source Licensing literature. It aims to assist the reader in choosing the best license for his/her business. For this reason, the content of this thesis can be divided into: (i) an open source licensing overview, (ii) the explication of the main features of the most popular open source licenses, (iii) the consequences of using one or another and (iv) the critical or controversial issues related to Open Source Licensing.

Furthermore, at the end of the thesis, the reader can find the method we followed to collect, classify and analyze the relevant information for the purpose of the survey.

**Key words:** Open Source Licensing, Open Source Licenses, Licenses consequences, critical review of Open Source literature, systematic review of Open Source literature, survey of Open Source literature, why to use an Open Source License, reason to use an Open Source License, advantages of Open Source Licenses, advantages of Open Source Licensing, consequences of Open Source Licenses, consequences of Open Source Licensing, how to choose an Open Source License, choosing an Open Source License.

# Preface

This master thesis is the result of one semester of work, from February to June 2007, at the Norwegian University for Science and Technology (NTNU). In addition, this master thesis has to let me finish my studies of Computer Engineering at the Technical University of Catalonia (UPC).

I would sincerely like to thank my supervisor, Professor Letizia Jaccheri, for her guidance throughout this thesis. I would also like to thank the vice-dean for International Affairs of the Faculty of Informatics of Barcelona, Professor Núria Castell, for giving me the opportunity of studying abroad.

I also have to mention the name of some friends, who have helped me with LaTeX: Eric Gutiérrez, Bernardino Casas and Marc Musquera. A special mention for Anna Torrents, Albert Torras and Xavier Orduña, who supported me through difficult times.

Finally, I also have to sincerely thank Brian Andersen and Chris Mueller, two English native speakers, for their patience answering my grammatical and orthographical doubts.

Trondheim, June 2007

Marc Querol

*The world is a book, and those*
*who do not travel read but a page.*
Saint Augustine

# Contents

# List of Figures

# List of Tables

# Part I

# Introduction

# Chapter 1

# Introduction

Open Source Software has increased in popularity in the latest years, and as a result, new software businesses have appeared. Open Source is a new way to release software that considerably differs from the traditional way of selling software applications. It has many advantages[1] but it is important to take into consideration several aspects[2] before starting to use, modify or distribute such products.

Licensing is an important aspect of Open Source Software. Although all Open Source Software has common characteristics[3] (defined by the Open Source Initiative), there are several kinds of licenses. These licenses differ in the rights they grant to the users and developers who are willing to use, modify or redistribute the licensed code. Therefore, businesses that misunderstand the license clauses of an Open Source product can face serious consequences. A business can also be damaged due to, for example, the rights allowed to competitors.

Licensing is therefore a crucial issue in Open Source Software. This work will thus attempt to resolve the main question relating to the different forms of Open Source licensing available: exposing the advantages and disadvantages of the main Open Source Licenses, we will try to assist the reader in choosing the proper Open Source License for his/her business.

[1]Some advantages of Open Source Software: less dependance on vendors, easier to customize, lower cost, reusability and continuous improvement.

[2]Aspects of the Open Source Software: Organization issues, Software Development process, business models and licensing.

[3]Free redistribution and access to the source code are the main common characteristics. See the others in subsection 2.3.1, page 10.

## 1.1 The purpose of this thesis

The purpose of this thesis is to provide a critical roadmap of the literature, printed and online, about Open Source Software. Since Open Source is not a single concept and there is a lot of literature written, we have focused on a specific part of the Open Source Software: licensing. To do that, we have proposed the following research question:

> **Research Question:**
>
> What are the different Open Source Licenses and what are their consequences?

The books, papers and articles found have been classified and analyzed. In addition to this, the extensive answer to the research question has been reported.

We will discuss about:

- what the most representative Open Source licenses are
- the consequences of using them (advantages and disadvantages)
- what considerations should be taken in order to choose an Open Source license
- what controversial issues of open source licensing exist

At the end, an overview of the research process is also included.

## 1.2 Content and Structure

This thesis consists in 4 parts distributed in a total of 9 chapters:

1. *Introduction*: contains one chapter and presents this thesis.
2. *Open Source Licensing*: contains the results of our critical review. It is formed by 5 chapters:
    - *What is Open Source Software?* where the principles of Open Source are described.
    - *Open Source Licenses* where the main features of the most popular licenses are explained.

   - *Advantages, disadvantages and consequences* where the reader can learn the reasons to use one or another license.
   - *Controversial issues and trends* where critical issues and possible trends are explained.
   - *Conclusions* where the main ideas of Open Source Licensing are underlined.

3. *Thesis Development*: contains the development of the thesis. Its 2 chapters are:

   - *Overview of the Thesis Development*: contains an introduction to critical reviews and to the research process.
   - *The Review Process*: where the specific review done for this thesis is explained.

4. *Conclusions*: contains one chapter of the same name.

   - *Conclusions:* where the reader can see the conclusions of this critical review.

# Part II

# Open Source Licensing

# Chapter 2

# What is Open Source Software?

## 2.1 Free Software: The origins of Open Source Software

Free Software is a term coined by Richard Stallman and the Free Software Foundation (FSF) to refer to software that can be used, studied, and modified without restriction, and which can be copied and redistributed in modified or unmodified form either without restriction, or with certain requirements to ensure that further recipients also have these freedoms. [Wikb]

According to Richard Stallman and the FSF, software is Free Software if people who receive a copy of the software have the following four freedoms:

- 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 neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this. [PTFSF]

## 2.2 The Open Source Initiative

Since the word "free" can have the meaning of gratis, some users of the Free Software community created the term Open Source to avoid ambiguities. Free in Free Software referred to free as "liberty" not as "gratis". The Open Source Initiative (OSI), created in 1998 by Eric S. Raymond and Bruce Perens [Wikc], presents The Open Source Definition to determine whether a software license can be considered Open Source or not.

It is important to note that Open Source and Free Software are not synonymous, although they have common characteristics. The FSF explains it in the following way:

> "For the Open Source movement, the issue of whether software should be open source is a practical question, not an ethical one. As one person put it, "Open source is a development methodology; free software is a social movement." For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and Free Software is the solution." [FSF]

As a result, there are Open Source licenses that have been approved by the Open Source Initiative and by the Free Software Foundation and other licenses that have only been approved by one of these organizations.

## 2.3 What makes a license Open Source?

### 2.3.1 The Open Source Definition

The Open Source Definition says that the distribution terms of Open Source Software must comply with the following criteria: [SI]

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."

### 2.3.2 What is not required in Open Source Licenses?

The Open Source Definition is not as extensive as typical software licenses. There are no requirements on license compatibility, warranties or formalities either: [Väl05]

**Moral Rights:** Open Source Definition does not say anything about attribution and reputation.

**Patent and other intellectual property licenses:** Open Source Definition places essentially requirements on copyright but not on other intellectual property rights. Apparently, the requirements of royalty-free use, distribution and modification imply that a patent or a trademark that requires royalty payments cannot exist. However, it seems to allow that a licensor licenses his patent without royalty to the open source community, whereas he collects royalties from the users of another proprietary software that implements the same patented invention.

**Controlled rights ownership:** Open Source licenses do not require controlled rights ownership.

**Warranties:** In the Open Source Definition, no warranties are required.

**Compatibility:** The Open Source Definition only defines a criteria to certify software licenses. As it is not a standards specification, it is possible that some open source code cannot be combined with another open source code, since the respective licenses of the open source codes are not compatible with each other.

**Formalities:** The Definition does not require that licenses should be legally binding or enforceable. The validity of any provision depends on its legal enforceability.

## 2.4 What is not Open Source?

### 2.4.1 Public Domain, Shareware and Freeware

Some software is released freely into the public domain and the author has no copyright rights, whereas Open Source not. Open Source Software is governed by a license and the owners of the copyright in the software continue to own the copyright and assert their rights thereto. In Open Source Software, people receive a software license that gives them more rights than they have become accustomed to expect under commercial software licenses, such as the possibility of modifying the code and even redistributing it.

Freeware or Shareware often has the connotation of being “free”. They are free because of the price, not because of the availability of their source code. Actually, their licenses usually have many of the restrictions found in commercial licenses. [Ken01]

# Chapter 3

# Open Source Licenses

## 3.1 Licenses classification

Even though there are many different Open Source Licenses, it is possible to group them according to how derivative works[1] are treated (functional differences) or according their historical origins. [Väl05]

### 3.1.1 Classification based on functional differences

The licenses classification depending on functional differences is as follows:

- **Highly restrictive licenses**: licenses with strong reciprocity obligations
- **Restrictive licenses**: licenses with standard reciprocity obligations
- **Permissive licenses**

**The Standard Reciprocity Obligation** means that the distribution terms of the source code must be maintained. Such licenses are commonly called as *copyleft*. If the source code is modified, the licenses terms cannot be changed. However, if the source code is combined with other source code to create a new work, the Standard Reciprocity Obligation does not apply to the combined work.

**The Strong Reciprocity Obligation** extends the standard reciprocity obligation: even adaptations and derivative works must keep the license terms intact. These licenses are considered to have a "viral" or "contagious" effect.

[1]A derivative work is a work based upon one or more preexisting works.

**The Permissive licenses** allow free distribution, copying and modifying. Licenses terms of the derivative works can be different and there are no reciprocal requirements.

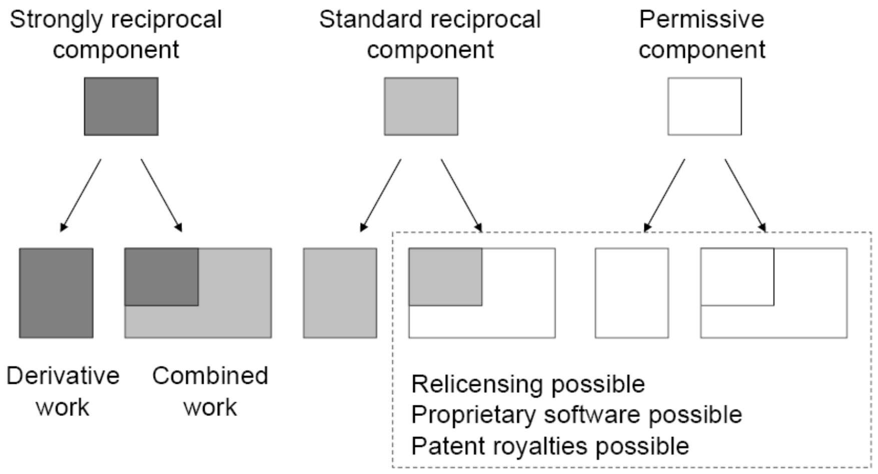

Strongly reciprocal component
Standard reciprocal component
Permissive component
Derivative work
Combined work
Relicensing possible
Proprietary software possible
Patent royalties possible

**Figure 3.1:** Functional differences

### 3.1.2 Classification based on historical origins

The classification based on the historical origins of the licenses is this one:

- **GNU licenses**
- **Academic licenses**
- **Community licenses**
- **Corporate licenses**

The **GNU licenses** were introduced by Richard Stallman and the Free Software Foundation in the 1980s. They are licenses with a strong ideological message. These licenses are familiar to developers, but attorneys have become hesitant about GNU licenses for their vague language and uncertain implications. In addition to this, GNU licenses are not compatible with many other open source licenses.

**Academic licenses** have their origins in the University of Berkeley. They are short and rather clear in language. Academic licenses are permissive and mostly compatible with other Open Source licenses.

**Community licenses** typically originate from some major Free Software project. The two most popular licenses are the Artistic License, distributed with the Perl programming language, and the Apache License. The Perl programming license is ambiguous. Legally, the Apache License is considerably more rigid than the Artistic License taking into account patents and trademarks.

The first major **Corporate license** was introduced by Netscape in 1998 when the open source code of its browser was released. The Corporate licenses are typically very detailed addressing issues such as patent and trademark licensing, copyright to code contributions and many formal issues not included in other Open Source licenses.

## 3.2 Popularity of the Open Source licenses

The following table shows the popularity of the Open Source Licenses at Sourceforge.net in July 2005. By that time, the number of projects registered in Sourceforge.net were higher than 110.000[2], but only 65.362 were approved by the Open Source Initiative. The rest of the projects were either proprietary licensed software not approved by the Open Source Initiative or projects under the public domain. [Cha05]

| **License name** | **Quantity** | **Percentage** |
|---|---|---|
| GNU General Public License | 45101 | 69% |
| GNU Library or Lesser General Public License | 7388 | 11% |
| BSD License | 4724 | 7% |
| Artistic License | 1230 | 2% |
| MIT License | 1195 | 2% |
| Apache Software License v1.1 | 968 | 1% |
| Mozilla Public License 1.1 | 827 | 1% |
| Common Public License | 503 | 1% |
| Apache License V2.0 | 452 | 1% |
| **Total** | 62388 | 95% |

**Table 3.1:** Open Source license popularity in Sourceforge.net

Throughout an empirical analysis of Sourceforge, Lerner and Tirole [LT05] found that projects geared toward end-users tend to have restrictive licenses, while those oriented toward developers tend to have more permissive licenses. In addition to this, Freshman and Gandal [FG05] observed that the output per contributor in Open Source programs is much higher when licenses are less restrictive.

[2]In June 2007, the number of registered projects in Sourceforge.net is higher than 149.000. [Sou]

## 3.3 Most popular Open Source Licenses

### 3.3.1 GNU General Public License (GPL)

The GNU General Public License explicitly requires that derivative works be distributed under the terms of the GPL License and also that derivative works may only be permitted to be distributed under the terms of the license. [St.04]

One of the the most important requirements of GPL is that the Open Source code always be made available for the recipient of the software.

The GPL license does not mean to give away anything for free. It is not forbidden to sell a work licensed under GPL. Furthermore, if a company is the only one who has written code for a certain GPL application, this company owns the full copyright of the code. As a a result, this company can distribute the code under whatever licenses it wishes. This is called *Dual Licensing* and consists in releasing software under different licenses.

For large collective projects, where each contributor owns the copyright to their pieces of source code, it would be unreasonable to consider Dual Licensing. Therefore, these large projects will be GPL forever. [Ing]

### 3.3.2 GNU Lesser General Public License (LGPL)

The GNU Lesser General Public License was created with the purpose of permitting a certain class of programs, generally subroutine libraries, to be licensed under an FSF license but be permitted to link with non-GPL sofware. [St.04]

As the GPL, the LGPL ensures the distribution of the licensed code will be under the same license model, but it lets users extend the LGPL-licensed source code with proprietary modules. [WL01]

Therefore, the GNU Lesser General Public License is a software license option available to commercial software developers without the obligation to release all of their software source code in derivative works. [Fle04]

### 3.3.3 The BSD style Licenses

The BSD (Berkeley Software Distribution) style licenses are the least restrictive of the Open Source licenses.

Under the BSD Licenses, distribution of source code is permitted but not mandated for derivative works. Therefore, programs under the BSD Licenses can be combined with proprietary software.

The BSD Licenses allow redistribution and use of source code and object code with or without modification so long as the redistribution of source code retain required copyright and other notices and the disclaimer of warranties and limitation of liability clauses.

The original BSD License (prior to 1999) had certain attribution requirements, including mandatory attribution of naming of contributors in advertising of software using the code. [Ken01] After removing this clause, the only substantial difference with the **MIT License** is the nonattribution provision. This provision requires prior permission for the use of the name of the creator and it protects the reputation of the creator from being explicitly associated with derivative versions of the program. Such restrictions permit creators to protect themselves from the injury to their reputations that can result from association with a defective or poorly written program, while still allowing others to use or modify the work. [St.04]

The BSD Licenses are considered by many to be more "free" than the GPL because they permit developers to release derivative works under whatever license they prefer. In other words, the BSD Licenses do not contain copyleft terms. This is the reason why the BSD type of licenses is attractive to commercial developers. [Ken01]

### 3.3.4 The Apache License

The older Apache License version 1.1 is very similar to the BSD-License, but includes a requirement for the acknowledgement of the creator's contributors of the software. The Apache License version 2.0 is more complex. It includes provisions for patent rights granted by the license and the use of other licenses for derivative software. The Apache License version 2.0 also explicitly defines "Contributions" that are special modification of the software provided to the licensor of the software for its inclusion into the original one. If accepted, the modifications will become part of the original software and will fall under the same license. [Cha05]

### 3.3.5 The Artistic License

The Artistic License was designed to maintain control over the Perl project while encouraging participation in the project and innovation outside the project. The Artistic License is ambiguous, self-contradictory and virtually impossible to interpret. [Cha05] One problem with the Artistic License is that although it prohibits sale of the software, it also allows an aggregate distribution of the Artistic Licensed software with another piece of software. Interpreted literally, someone can defeat the license by merely including a trivial piece of software together with the licensed software. [Cha05]

Under the Artistic License, it is possible to modify the source code and make it private if the standard version is also supplied with the modified files; or, if the modified files have been renamed and the standard version still works. [Ken01]

The Artistic License does not require distributing derivative works under the same terms when a company uses them internally. [WL01]

### 3.3.6 The Mozilla Public License

The Mozilla Public License can be regarded as a hybrid of ideas between the GPL and the MIT/BSD Licenses. MPL-licensed code can be combined with code under another license.

The MPL divides a software work into an Open Source part (called "Covered Code") and anything a contributor adds. The arrangement allows any developer to add his own files and distribute them with the covered code, provided he does not modify the covered code. However, if he does modify the covered code, he must distribute the modified code under MPL.

The MPL is considered one of the better drafted Open Source licenses and is used in many open source projects including the popular Firefox browser. [Cha05]

### 3.3.7 The Common Public License (CPL)

Introduced by IBM, the Common Public License has some terms that are similiar to the GNU General Public License. However, there are significant differences. The first one is that one may compile a program under the CPL without modification and license it under a proprietary license. The second one is that the CPL requires the contributor to grant a royalty-free license to all recipients.

As in the GPL, the CPL imposes to make the source code to the modified programs available. [Wika] [webb]

The **Eclipse Public License** (EPL) is a license derived from the Common Public License. The difference between the CPL and the EPL is that the EPL does not terminate any patent licenses granted by a Contributor to a recipient, if this recipient has instituted a patent litigation against the mentioned contributor. [weba]

### 3.3.8 The Common Development and Distribution License (CDDL)

The Common Development and Distribution License is based on the Mozilla Public License and makes it reusable without modification. The CDDL is a copyleft license that provides open source protections and freedom and also enable creation of larger works for commercial purposes.

The CDDL allows files released under CDDL to be linked with files released under other licenses. The MPL license allows the same practice, but unfortunately, the MPL is not a "template" license allowing reuse by others.

Therefore, Solaris found it necessary to create the CDDL to license its software and to have a template for reducing the proliferation of other licenses with similar clauses to MPL. [webc]

### 3.3.9 The Q Public License (QPL)

The Q Public License was designed by the Norwegian firm Trolltech to govern the distribution of its software, the Qt toolkit. The most significant feature is that QPL requires that the modified Open Source distributions be distributed as patches. [WL01]

### 3.3.10 Creative Commons Licenses: Extending the Open Source philosophy

The Creative Common Licenses were not written for use in connection with software. However, it is interesting to mention them since they are designed to encourage creators of works to make their work available for public use. The Creative Common Licenses provide a solid basis for licensing the "Open Source" use of texts, music, web sites and films. [St.04]

## 3.4 Mixing code from different licenses

It is possible to mix code from different licenses provided they do not contain incompatible clauses. As mentioned in section 2.3.2 page 12, Open Source licenses may be incompatible since the Open Source Definition is not a standards definition.

Given that the GPL is the most popular Open Source license, it is interesting to show what licenses are compatible with the GPL. See table 3.2.

| Open Source License | GPL-compatible |
|---|---|
| GNU Lesser General Public License | ✓ |
| MIT License | ✓ |
| BSD License (current version) | ✓ |
| Apache 2.0 | × |
| Artistic License (original version) | × |
| Mozilla Public License | × |
| Common Public License | × |
| Eclipse Public License 1.0 | × |
| Common Development Distribution License | × |
| Q Public License | × |

**Table 3.2:** Licenses compatibility with GPL

## 3.5 Summary

In this chapter, the most representative Open Source Licenses have been explained, taking into consideration the main features of each license. The table 3.3 obtained from Sau Sheong Chang [Cha05] is a good comparison between the rights that some of the explained licenses grant.

| **Freedoms or Restrictions** | Public Domain | MIT/ BSD | Apache 1.1 | Apache 2.0 | Artistic | MPL 1.1 | GPL | LGPL | Closed Source |
|---|---|---|---|---|---|---|---|---|---|
| Has copyright owner | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Copyright acknowledgement | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Freely copy and use as-is | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Distribute modified versions with the same license | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Distribute modified versions under the same license | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | × | × |
| Link with code under different license | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × |
| Must include source code in the distribution | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Grants licensee patent rights | × | × | × | ✓ | × | × | ✓ | ✓ | × |
| Disclaimer of warranty/limitation of liability | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Non-endorsement provision | × | × | ✓ | ✓ | × | ✓ | × | × | NA |
| Reciprocity obligations for derivative works (copyleft) | × | × | × | × | × | ✓ | ✓ | ✓ | × |
| Number of words in license document (complexity of the license) | NA | 167 / 222 | 294 | 1581 | 771 | 3666 | 2956 | 4020 | Varied |

**Table 3.3:** Free and open source license rights matrix

# Chapter 4

# Advantages, Disadvantages and Consequences of Open Source Licensing

## 4.1 Why use an open Source License?

The unlimited improvement of software is a major key of the Open Source products. However, why would a software company be interested in releasing its products under Open Source Licenses? Even though this company cannot get revenues using traditional software licenses and fees, it can find other ways of generating revenues and profits based on the value it provides to its customers: [Hec99]

- Support Sellers: Revenue comes from media distribution, branding, training, consulting, custom development, and post-sales support.
- Loss Leader: A no-charge Open Source product is used as a loss leader for traditional commercial software.
- Accessorizing: A company distributes books, computer hardware, and other physical items associated with and supportive of Open Source Software.
- Service Enabler: Open Source Software is created and distributed primarily to support access to revenue-generating online services.
- Brand Licensing: One company charges other companies for the right to use its brand names and trademarks in creating derivative products.

- Sell It, Free It: A company's software products start out their product life cycle as traditional commercial products and then are continually converted to Open Source products when appropriate.

- Software Franchising: This combines several of the preceding models (in particular Brand Licensing and Support Sellers). A company authorizes others to use its brand names and trademarks in creating associated organizations doing custom software development in particular geographic areas or vertical markets. The company might also supply franchises with training and related services in exchange for franchise fees of some sort.

## 4.2 Which is the best license for a specific project?

There are several kinds of licenses, but how can we know which one we should use to release our software? Lawrence Rosen, an attorney and computer specialist who was executive director and general counsel for the Open Source Initiative, posed several questions that a company should ask itself in order to decide what kind of license is better for its business. [Ros01]

- Do you intend to make money from licensing the software or from providing ancillary services like installation and training?

  Answer: proprietary software.

- What degree of freedom are you willing to grant to your licensees to modify your software?

  Answer:

  - BSD-type imposes virtually no restrictions on licenses; they can modify the licensed software and create proprietary versions without restriction.
  - GPL-type rquires the licensee's modifications to be licensed back under the same license; this is good for community contribution.
  - MPL-type imposes an intermediate level of freedom; modifications to individual files containing licensed code must be licensed back; but new files that simply work with the licensed code need not to be.

- Are you willing to grant warranties that the software will be "merchantable" or "fit for a particular purpose"?

Answer: If the software is royalty-free, you probably can't afford a warranty. However, you can charge your Open Source Software to provide warranties and other services.

- Is your software so well known that the main asset you need to protect is your trademark rather than your code?

  Answer: Apache License is an excellent example. You are allowed to do almost everything with the Apache code, but you will have to change the name. If you have a trademark to protect, make sure your license contains appropriate terms relating to that.

- Have you considered the possibility of Dual Licensing?

  Answer: you may want to license your software under GPL and simultaneously provide a proprietary version for those of your customers who are afraid of the GPL's inheritance features; this can be treated as a revenue opportunity.

- Have you considered using different licenses for different parts of your software?

  Answer: client software might be distributed under an MPL-like license, but server software might be distributed under a proprietary license. That way, you could make money from the bigger customers that will pay to license your server software, and simultaneously build a large customer base with free clients.

- What is it about your software that you are really trying to protect? Is it the code itself, or the standards that are implemented using that software?

  Answer: A license like SISSL (Sun Industry Standards Source License) allows anyone to develop modifications of licensed software as long as the licensee complies with all requirements set out by a standards body; a licensee who elect not to comply with the specification must publish a royalty-free reference implementation of the modifications so that the standard cannot be abducted by another company.

- Are there any patents that relate to your software?

  Answer: if so, you will have to consider licensing your patents along the code.

All these questions are not enough to correctly choose the right Open Source License for some specific software. It is true that they give a good idea about how to face the problem of the license choice. It is also true that more details about the consequences of choosing each licenses will be exposed. However, if the reader wants to release a product under an Open Source License, we suggest that he ask for advice from specialists in Open Source Licenses.

## 4.3 Consequences of Restrictive and Permissive Licenses: A first approach

Before starting to explain the main reasons to use one or another license, it is interesting to look at the conclusions found by Lerner and Tirole [LT05] and by Fershtman and Gandal [FG05]:

- Projects geared toward end-users tend to have restrictive licenses, while those oriented toward developers are less likely to do so.
- Projects that are designed to run on commercial operating systems and whose primary language is English are less likely to have restrictive licenses.
- Projects that are likely to be attractive to consumers -such as games- and software developed in a corporate setting are more likely to have restrictive licenses.
- Projects with permissive licenses attract more contributors.
- The output per contributor is much higher when licenses are less restrictive.
- On average, firms that employ software with restrictive licenses supply fewer proprietary products than firms that employ software with less restrictive licenses.

From these conclusions, we can extract that a company should use **permissive licenses** if it wants a large number of developers to help improve its software. Furthermore, the work of each developer will be also higher since the output per contributor is higher with permissive licenses.

However, if a company wants to release software geared toward end-users, it would be reasonable that it uses **restrictive licenses**, since the projects derived from this software are also under restrictive licenses (containing copyleft clauses).

In addition to this, it is important to note that permissive licenses allow source code to become proprietary and their derivative works do not need to be under the same license. Therefore, a company can be damaged if it releases source code under permissive licenses and a competitor takes advantages of it.

## 4.4 Specific consequences of each license and the reasons to use it

### 4.4.1 Reasons to use the GPL

The main reasons to use the GPL: [Ing]

- The code developed in a GPL project will stay GPL forever and nobody will be able to use your source code in his/her own proprietary software.
- *Dual Licensing* is possible, providing (in exchange for money) an alternative license for those who not accept the GPL requirements.

See the subsection 4.4.6 page 31 to see how the GPL-source code cannot remain available in Web Services. However, this problem might be resolved in the new version of the GPL license (GPLv3) that should appear within this year. [Ing]

### 4.4.2 Reasons to use the LGPL

The main reason to use the LGPL license is to allow the “normal use” of the Open Source Software by everyone. Even though it lets users extend the LGPL source with proprietary modules, it is very convenient for a library that will be used for many different applications. [Ing] [WL01]

Richard Stallman argues that using the GPL license makes the library available only for free (and Open Source) programs. Therefore, the use of the GPL for a library gives Free Software developers an advantages over proprietary developers. Nevertheless, if the library’s features are available for proprietary software through other alternative libraries, it is not useful to use the ordinary GPL license since it does not give Free Software any particular advantage. [Sta99]

### 4.4.3 Reasons to use the BSD style license

The main reason to use this kind of licenses is its permissiveness. A company can do whatever it wants with the code. Relicensing with proprietary licenses is allowed since it is not mandatory to redistribute the source code. Nevertheless, it is important to verify if the license contains any advertising clause that requires to give the appropriate credit to the copyright holders of the copied code. [Ing]

### 4.4.4 Reasons to use the CPL

The main reasons to use the CPL are that: [webb]

- It is possible to make proprietary versions of the CPL code unless it has been modified.
- It is also allowed to link the CPL code with proprietary modules.
- It is only mandatory to redistribute the CPL source code if it has been modified.
- The improvements of the source code will be always available.

It is important to note that an enterprise that modifies the CPL code cannot keep it proprietary since it is mandatory to redistribute the *modified* CPL-source code.

### 4.4.5 Reasons to use the CDDL

The main reasons to use the CDDL are that: [webc]

- It is a copyleft license that protects the files licensed under CDDL from becoming proprietary.
- It is possible to create proprietary programs linking proprietary modules with CDDL code since the non CDDL files do not need to be licensed under the CDDL.

### 4.4.6 Reasons to use the other licenses

Some of the licenses explained in the section 3.3 have not been exposed in this section. According to the Open Source Initiative, some of them have been superseded and others are redundant with more popular licenses[1]. There are also licenses that are not reusable because they are related to a specific software product or company[2].

In addition, it is important to note that the use of some licenses can be assessed based on for their vast use in similar projects. A well-known license in the

[1]The Mozilla Public License and the Apache Software licenses are examples of superseded licenses.

[2]The Nokia Open Source License is an example of a non-reusable license.

developer community will be more attractive for developers than another license that is not.

### Web Server Licenses

Even though Web Server Licenses are not among the most popular, it is interesting to see when they can be useful.

The **Affero General Public License**[3] is a Web Server License. The Affero General Public License is identical to the GPL, but with the added requirement that publishing the software as a web service also requires providing an opportunity for users to download the source code. Under a GPL license, if the code is published as a webserver for users to enjoy, it is not necessary to distribute the source code, since there is no redistribution of software to the users. [Ing]

[3]The Affero General Public License is a Free Software license, but not an Open Source License.

# Chapter 5

# Controversial issues and trends

Open Source Licenses have been created to extend and preserve users' rights. However, some problems may arise due to the way that Open Source Licenses have been redacted and the fact that different countries have different laws. There are also critical points related to the development model and the trustworthiness of Open Source Software for consumers. This chapter will thus present and explain important controversial issues and interesting trends.

## 5.1 Open Source Licenses - copyright license or contract?

If Open Source licenses are copyright licenses or legal contracts is yet to be determined. Although both legal contracts and copyright licenses have different enforcements (one which is covered through the contract itself, and the other which is through copyright legislation), both enforce the same terms and conditions of the licenses. However, one noticeable difference is that without a legal contract, licensors can revoke their licenses at any point in time, subject to equitable rules. This has some serious repercussions if the software is already well known in the market as the licensor is not obliged to continually provide the software under the same license.

In addition to this, it is not clear if all Open Source licenses can be considered valid contracts, althought similar to closed source licenses have been accepted as valid contracts, albeit controversially. Interestingly, if a contract does not exist for Open Source licenses, sometimes copyright laws of certain countries impose a harsher criminal offence on copyright infringers; therefore it would seem that it is to the benefit of the licensee not to use this as a defense against enforceability of Open Source Software licenses. [Cha05]

## 5.2 Cross-jurisdictional issues in Open Source licenses

A major issue with the copyright is the issue with the applicability of the law in different countries. Copyright laws are territorial and generally do not cross borders: [Cha05]

1. Most Open Source licenses are written in English and in fact assume certain facts that are only applicable in US laws. However, in many countries, there are laws that mandate the use of the national language for legal documents including licenses and contracts.

2. Legal background which copyright law is derived from. In countries that derive laws from English legal system, copyright arose from the economic rights of copywriters and publishers while most continental European countries derive copyright from the concept of *droit d'auteur*, which focuses on the moral rights of the original author.

3. Warranties and disclaimers. In certain countries, especially European countries, general disclaimers are not valid in a contract due to provisions for unfair terms in contracts.

4. The possible different interpretation of derivative works across different countries.

## 5.3 Other critical issues and trends

1. The exclusion of warranties for software defects in most Open Source licenses should cause organizations considering the adoption of Open Source Software to carefully consider how quality and reliability can be assured.

2. Since the enhancement of reputation is an important motivating factor in Open Source Software development, software authors might benefit from more uniform international recognition of their right to assert authorship and their right to avoid derogatory treatment as author of a work.

3. Quality and reliability characteristics of Open Source Software raise concerns for organizations in areas where certification is needed such as in

mission-critical activities or medicine. The lack of formal tools for testing should lend a note of caution to those considering the use of Open Source Software.

4. An organization that has been granted a software patent for some algorithm or implementation is granted the rights to charge royalties for use, or may force others to cease distribution of software that employs the scheme covered by the patent. Open Source Software is vulnerable to this form of restriction since all source code is publicly-available.

5. Peer-review of public software is an advantage, but successful outcomes still depend on the motivation of properly-skilled individuals to methodically study, probe and fix Open Source Software problems. [Fle04]

6. Open Source Licenses do not provide that the licenses are perpetual. This omission may cause problems at a later point when a court determines that an Open Source License respect to a program has expired because the “reasonable” duration is over. [Ken01]

# Chapter 6

# Conclusions

Licensing is an important aspect of Open Source Software. Although all Open Source Software has common characteristics (defined by the Open Source Initiative), there are several kinds of licenses. We can classify them depending on its redistribution requirements:

- Permissive licenses (such as the BSD) impose virtually no restrictions on licenses and it is possible to modify the licensed software and create proprietary versions without restriction.
- Restrictive licenses (such as the MPL) impose an intermediate level of freedom since modifications to individual files containing licensed code must be relicensed under the same terms, but new files that simply link to the licensed code do not need to be.
- Highly restrictive licenses (such as the GPL) require that any derivative work or any work that links to the licensed code must be relicensed under the same terms. It is good for community contribution.

Depending on the business, it should be better to use one or another type of licenses. When a company owns all the rights to specific software, *Dual Licensing* is possible, and consists in releasing software under different licenses. Companies can take different kind of benefits thanks to the different licenses terms.

As seen in the chapter 4, a company should use permissive licenses if it wants a large number of developers to help improve its software. Furthermore, the work of each developer will be also higher since the output per contributor is higher with permissive licenses. However, a company can be damaged if it releases source code under permissive licenses and a competitor takes advantages of it.

If a company wants to release software geared toward end-users, it would be reasonable that it uses restrictive licenses, since the projects derived from this software are also under restrictive licenses.

As explained in the chapter 5 some problems may arise due to the way that Open Source Licenses have been redacted and the fact that different countries have different laws. Disclaimer clauses are not allowed in many countries and they also have a negative impact on the confidence that users give to Open Source Software.

# Part III

# Thesis Development

# Chapter 7

# Overview of the Thesis Development

This part of the thesis contains an overview of the development of the thesis. To document it, we find interesting to start explaining what a critical review is, its steps and what kind of data we can extract with it. Afterwards, we will explain in detail all the steps followed to write this thesis.

It is important to mention that the thesis development has been based on the paper: "Procedures for Performing Systematic Reviews" by Barbara Kitchenham [Kit04]. Actually, the following sections of this chapter summarize the most important points of a critical review exposed by Kitchenham.

## 7.1 What is a Critical Review?

A critical review is a means of evaluating and interpreting all available research relevant to a particular research question, topic, area or phenomenon of interests. It contains three phases:

- planning the review
- conducting the review
- reporting the review

### 7.1.1 Importance of a Critical Review

A critical review synthesizes existing work in manner that is fair and seen to be fair.

### 7.1.2 Feature of Critical Reviews

- Critical reviews start by defining a review protocol that specifies the research question being adressed and the methodes that will be used to perform a review.
- Critical reviews are based on defined search strategy that aims to detect as much of the relevant literature as possible.
- Critical reviews document their search strategy.
- Critical reviews require explicit inclusion and exclusion criteria to asses each potential primary study.
- Critical reviews specify the information to be obtained from each primary study including quality criteria by which to evaluate each primary study.

## 7.2 The Review Process

The stages associated with planning the review are:

1. Identification of the need for a review
2. Development of a review protocol

The stages associated with conducting the review are:

1. Identification of the research
2. Selection of primary studies
3. Study quality assessment
4. Data extraction and moinioring
5. Data synthesis

Reporting a review is a single stage phase.

## 7.3 Planning

The need for a critical review arises from the requirement of researchers to summarize all existing information about some phenomenon in a thorough and unbiased manner. This may be in order to draw more general conclusion about some phenomenon than is possible from individual studies, or as a prelude to further research activities. Prior to undertaken a critical review, researches should ensure that a critical review is necessary. See the following checklist:

- What are the review's objectives?
- What sources were searched to identify primary studies? Were they any restrictions?
- What were the inclusion/exclusion criteria and how were they applied?
- What criteria were used to assess the quality of primary studies and how were they applied?
- How were the data extracted from the primary studies?
- How were the data synthesized? How were differences between studies investigated? How were data combined? Was it reasonable to combine the studies? Do the conclusions flow from the evidence?

### 7.3.1 Development of a review Protocol

A review protocol specifies the methods that will be used to undertake a specific critical review. A pre-defined protocol is necessary to reduce the possibility researcher bias.

The components of a protocol:

- Background. The rationale for the survey.
- The research question that the review is intended to answer.
- The strategy that will be used to search for primary studies including search terms and resources to be searched, resources include databases, specific journals, and conference proceedings. An initial scoping study can help determine an appropriate strategy.

- Study selection criteria and procedures. Study selection criteria determine criteria for including in, or excluding a study from, the critical review. It is usually helpful to pilot the selection criteria on a subset of primary studies. The protocol should describe how the criteria will be applied.
- Study quality assessment checklists and procedures. The researchers should develop quality checklists to assess the individual studies. The purpose of the quality assessment will guide the development of checklists.
- Data extraction strategy. This should define how the information required from each primary study would be obtained. If the data require manipulation or assumptions and inferences to be made, the protocol should specify an appropriate validation process.
- Synthesis of the extracted data. This should define the synthesis strategy. This should clarify whether or not a formal meta-analysis is intended and if so what techniques will be used.
- Project timetable. This should define the review plan.

### 7.3.2 The Research Question

A critical issue in any critical review is to ask the right question.

### 7.3.3 The Review Protocol

PhD (or master) students should present their protocol to their supervisors for review and criticism.

## 7.4 Conducting the review

### 7.4.1 Identification of the Research

Generating a search strategy:

- Reference list from relevant primary studies and review articles.
- Journals, grey literature and conference proceedings.
- Research registers.

- The Internet. It is also important to identify specific researchers to approach directly for advice on appropriate source material.

Documenting the Search:

- The review must be documented in sufficient detail for readers to be able to assess the thoroughness of the search.
- The search should be documented as it occurs and changes noted and justified.
- The unfiltered search results should be saved and retained for possible reanalysis.
- Study selection criteria are intended to identify those primary studies that provide direct evidence about the research question.
- Contents of Data Collection Forms
  - Name of the review
  - Date of Data extraction
  - Title, authors, journal, publication details
  - Space for additional notes

## 7.5 Final remark

In the case of a single research (such as PhD student), the most important steps suggested to undertake are:

- Developing a protocol
- Defining a research question
- Specifying what will be done to address the problem of a single researcher applying inclusion/exclusion criteria undertaking all the data extraction.
- Defining the research strategy.
- Defining the data to be extracted from each primary study including quality data.
- Maintaining lists of included and excluded studies.
- Using the data synthesis guidelines.
- Using the reporting guidelines

# Chapter 8

# The Review Process

After having explained the different steps to perform a critical review, we can proceed to explain and document the specific review related to this thesis.

## 8.1 The Review Protocol

### 8.1.1 Background

As explained in the introduction, Open Source Licensing is a crucial issue of the Open Source Movement. There are many licenses and the consequences of using one or another are very different.

Many papers discuss Open Source Software. However, very few discuss the specific topic of the Open Source Licensing. Therefore, we considered that it was very interesting to perform a critical review of the existing literature linked to the Open Source Licensing in order to better understand the topic.

### 8.1.2 The Research Question

The Research Question that we wanted to answer was:

> **Research Question:** What are the different Open Source Licenses and what are their consequences?

### 8.1.3 Search Strategy

#### Reliable Sources

To perform the search, it was important to determine the resources that would be consulted. A list of reliable sources of Software Engineering topics is showed bellow:

**IEEE Xplore** (`http://ieeexplore.ieee.org/`)

- IEEE Computer
- IEEE Software
- Transactions on Software Engineering

**ACM Digital Library** (`http://www.acm.org/`)

- International Symposium on Empirical Software Engineering
- International Conference on Software Engineering
- Transactions on Software Engineering and Methodology

**Springerlink** (`http://www.springerlink.com`)

- Empirical Software Engineering

**ScienceDirect** (`http://www.sciencedirect.com/`)

- Information and Software Technology

**BRINT database** (`http://portal.brint.com/`)

**Google scholar** (`http://scholar.google.com/`)

**CiteSeer** (`http://citeseer.ist.psu.edu/`)

#### Key Words

We also defined the *key words* to find the information in the above mentioned databases. The key words were the following:
open source, oss, license, licensing, consequences, implications, survey, review.

### Inclusion/Exclusion criteria

Finally, we defined the inclusion/exclusion articles criteria:

How to include or exclude an article?

- Reading the title article and the abstract
  - Does the abstract mention issues linked to OSS[1] licensing?
    - YES - First look at the full text looking for the parts that talk about OSS licensing.
      - Does it give useful information for our Research question?
        YES - Article Included
        NO - Article Excluded
    - NO - Article Excluded
- Reading the references of already selected articles, we could also find more articles

### Analysis and classification

The important data to be recorded for each selected article / paper / book was:

- Title
- Author
- Publisher / Journal / Conference
- ISSN number
- Year
- Month
- Pages
- Topic
- Date of Data extraction
- Why I included this
- Time needed for analysis

[1]Open Source Software.

- Summary - synthesis of the main points of the text making emphasis in the OSS licensing part.

## 8.2 Documenting the search

After having defined the review protocol, we could start the research process. We started consulting the mentioned databases, and the number of papers found was very disappointing.[2] See tables 8.1 and 8.2.

| Journals / Conferences / Transactions | Documents found | Total | % |
|---|---|---|---|
| IEEE Computer | 3 | 476712* | 0,0 |
| IEEE Software | 13 | 476712* | 0,0 |
| Transactions on Software Engineering | 0 | 476712* | 0,0 |

* Documents of different publications mixed.

**Table 8.1:** Documents found in the IEEE Explorer database using the key words: *open source licenses*

| Journals / Conferences / Transactions | Documents found | Total | % |
|---|---|---|---|
| IEEE Computer | 0 | 476712* | 0,0 |
| IEEE Software | 0 | 476712* | 0,0 |
| Transactions on Software Engineering | 0 | 476712* | 0,0 |

* Documents of different publications mixed.

**Table 8.2:** Documents found in the IEEE Explorer database using the key words: *survey licenses*

The results obtained in the first consulted database were very poor. We had only found 16 documents. However, it did not mean all of them were useful for our research. We had to include or exclude them to our review following the criteria defined in the Review Protocol.

Using the other databases and other key words, we tried to find more articles. The results were not much better. However, it is interesting to note the results found on *Transactions on Software Engineering and Methodology* and *International Conference on Software Engineering.* See tables 8.3 and 8.4.

[2]The Search was performed at the end of February 2007.

| Journals / Conferences / Transactions | Documents found | Total | % |
|---|---|---|---|
| Trans. on Software Engineering and Methodology | 102 | 258 | 39,5 |
| International Conference on Software Engineering | 860 | 2950 | 29,1 |

**Table 8.3:** Documents found in the ACM Digital Library database using the key words: *open source licenses*

| Journals / Conferences / Transactions | Documents found | Total | % |
|---|---|---|---|
| Trans. on Software Engineering and Methodology | 35 | 258 | 13,5 |
| International Conference on Software Engineering | 337 | 2950 | 11,4 |

**Table 8.4:** Documents found in the ACM Digital Library database using the key words: *survey licenses*

In the first moment, we thought that the results obtained from the ACM database were much better. However, after performing the inclusion and exclusion criteria, few documents were kept for a posterior analysis.

The other key words mentioned in the Review Protocol and combinations of them were used in order to find more documents. We did not record all these researches, but the obtained results were not satisfying either.[3]

After having summarized some of the articles, we decided to proceed with the data collection throughout the articles references.

### 8.2.1 Classification of the articles

When an article, paper or book passed the criteria for being studied in this critical review, all the important data were recorded in a table.[4] See figures A.1, A.2, A.3 in the appendix (page 66).

After recording an article, it was analyzed. This analysis consisted in a carefully reading and a posterior summary of the article. However, since the extracted information had to be useful to answer the research question, there were long

[3]During the time of collecting data, I was more worried about trying to find more documents than recording searches. This is the reason why I do not show more searches in this report.

[4]All electronic documents were stored.

articles which contained very little information and their summaries were much more shorter than other shorter articles that contained much more interesting information for this research.

See appendix B page 69 to see all the summaries. Books were only underlined and not summarized, because of their length.

## 8.3 Reporting the results

After having done the research of articles and extracted their most important information related to the research question, we had to write the results. As seen in part II, the answer to our research question has been reported.

In order to write the results, we started by thinking what we wanted the reader to learn with this thesis. Afterwards, using the different summaries and extra data obtained from the Internet, the Open Source Licensing part was performed.

### 8.3.1 Problems with reporting the results

The Research Question was “What are the different Open Source Licenses and what are their consequences?”. However, were the articles found enough to fully answer this question?

We decided to explore the Open Source Initiative web site to see what were the different Open Source licenses accepted at the present time. We found that some of the most relevant licenses, such as the CPL and the CDDL, had not been explained in the articles found.

Therefore, searching on the Internet, interesting data related to those licenses were found and reported. We visited the official web sites of the licenses creators.

# Part IV

# Conclusions

# Chapter 9

# Conclusions

Through this thesis, the results of a critical review of Open Source Software literature have been presented. Since Open Source Software has different aspects and the thesis had to be performed in one semester, we focused the review on Open Source Licensing.

In order to find relevant information to answer the question "What are the different Open Source Licenses and what are their consequences?", we established a specific protocol to assure the validity of the data and to avoid bias. We decided to follow the steps explained in "Procedures for Performing Systematic Reviews" [Kit04].

We found some relevant articles related to Open Source Licensing that helped us to answer our research question. However, we think that a good work could not have been done if we had not visited web sites not defined in our protocol. Was it a mistake to not follow the protocol in some specific cases?

A critical review has to be performed throughout the analysis of primary studies. However, we did not have enough primary studies to write a consistent report answering our research question. We did not analyze licenses to extract conclusions, since we are not expert attorneys, but we tried to find the necessary data on the Internet. Principally, we obtained the information from official pages of the licenses creators.

In short, this thesis is a well-documented work aimed to explain the main Open Source Licenses and consequences of their use. If we did not make any mistake defining the protocol, the information reported in this thesis must be reliable. It is true that we allowed some exceptions to the protocol, but we were always taking into account the reliability of the sources with the main goal of improving the content of the thesis.

# Glossary

**Contract:** an official written agreement.

**Copyleft:** the distribution terms of the source code must be maintained.

**Copyright:** if a person or an organization holds the copyright on a piece of writing, music, etc., that person or organization is the only one who has the legal right to publish, broadcast, perform it etc., and others must obtain their permission to use it or any part of it.

**Critical review:** a means of evaluating and interpreting all available research relevant to a particular research question, topic, area or phenomenon of interests.

**Derivative work:** a work based upon one or more preexisting works.

**Dual Licensing:** to release the same software product under different license terms.

**Free Software:** software that can be used, studied, and modified without restriction, and which can be copied and redistributed in modified or unmodified form either without restriction, or with certain requirements to ensure that further recipients also have these freedoms.

**Freeware:** copyrighted computer software which is made available for use free of charge, for an unlimited time.

**License:** an official document that shows that permission has been given to do, own or use something.

**Open Source Software:** software whose source code is available under a license (or arrangement such as the public domain) that permits users to use, change, and improve the software, and to redistribute it in modified or unmodified form.

**Patent:** an official right to be the only person to make, use or sell a product or an invention.

**Public Domain:** comprises the body of knowledge and innovation (especially creative works such as writing, art, music, and inventions) in relation to which no person or other legal entity can establish or maintain proprietary interests within a particular legal jurisdiction.

**Shareware:** copyrighted computer software, accompanied with a request of payment, which is made available for use free of charge during a limited period.

**Trademark:** a name, symbol or design that a company uses for its products and that cannot be used by anyone else.

# Bibliography

[Agr02] P. Agrain. A framework for understanding the impact of GPL copylefting vs. non copylefting licenses. 2002.

[Ane99] J. Anez. The true meaning behind open source licenses. 1999. Online: `http://www.suigeneris.org/writings/1999-10-04.html`, accessed on March 24th, 2007.

[Cha05] S. S. Chang. Copyright and Open Source Software Licensing. 2005. Online: `http://law.bepress.com/expresso/eps/773`, accessed on March 24th 2007.

[Cue05] L.É. Cuellar. Open source license alternatives for software applications: is it a solution to stop software piracy? 2005. ACM Southeast Regional Conference. Proceedings of 43rd annual southeast regional conference.

[Dav01] R. Davis. The digital dilemma. 2001. Communications of the ACM.

[FG05] C. Fershtman and N. Gandal. Open Source Projects: Output per Contributor and Restrictive Licensing. 2005.

[Fit06] B. Fitzgerald. The transformation of Open Source Software. *MIS Quarterly*, 2006.

[Fle04] S. Flemming. Open Source Intellectual Property Rights. Issues and Trends. 2004. Online: `http://www.cs.otago.ac.nz/research/publications/oucs-2004-14ver2.pdf`, accessed on March 24th, 2007.

[FSF] GNU Project. The Free Software Foundation. Why "Free Software" is better than "Open Source". Online: `http://www.gnu.org/philosophy/free-software-for-freedom.html`, accessed on May 23rd 2007.

[Hec99] F. Hecker. Setting up the shop: The business of opensource software. *IEEE Software*, 1999.

[Ing] H. Ingo. Open Source Licensing.
Online: `Online:http://www.asptoday.com/Content.aspx?id=2397`, accessed on March 26th 2007.

[Ken01] D.M. Kennedy. A Primer on Open Source Licensing Legal Issues: Copyright, Copyleft and Copyfuture. 2001.
Online: `http://www.denniskennedy.com/opensourcedmk.pdf`, accessed on March 24th 2007.

[Kit04] B. Kitchenham. Procedures for Performing Systematic Reviews. Technical report, Software Engineering Group Department of Computer Science Keele. University Keele, 2004.

[LT05] J. Lerner and J. Tirole. The scope of Open Source Licensing. *Journal of Law, Economics and Organization*, 2005.

[OV02] V. Oksanen and M. Välimaki. Evaluation of Open Source licensing models for a company developing mass market software. 2002. Helsinki Institute for Information Technology.

[Oz98] E. Oz. Acceptable protection of software intellectual property. a survey of software developers and lawyers. *Information & Management*, 1998.

[Pau04] L.D. Paulson. Open Source databases move into the marketplace. *IEEE Computer*, 2004.

[PTFSF] GNU Project. The Free Software Foundation. The Free Software Definition.
Online: `http://www.gnu.org/philosophy/free-sw.html`, accessed on May 20th 2007.

[Ros98] D. Rosenberg. Evaluation of public software licenses. 1998.
Online: `http://www.stromian.com/Public_Licenses.html`, accessed on March 20th 2007.

[Ros01] L. Rosen. Which Open Source license should i use for my software?
Online:
`http://www.rosenlaw.com/html/GL5.pdf`, accessed on March 20th, 2007, 2001.

[SI] Open Source Initiative. The Open Source Definition.
Online: `http://www.opensource.org/osd.html`, accessed on May 20th 2007.

[Sou] Sourceforge.net.
Online: `http://sourceforge.net/`, accessed on June 4th 2007.

[St.04] A. M. St. Laurent. *Understanding Open Source & Free Software Licensing.* O´Reilly, 2004.

[Sta99] R. Stallman. Why you shouldn't use the library gpl for your next library. Online:
"`www.gnu.org/philosophy/why-not-lgpl.html`, accessed on March 24th, 2007, 1999.

[Ued06] M. Ueda. A Model of Open Source Software Style R&D on Business. 2006. International Conference on Software Engineering Advances.

[Väl05] M. Välimäki. *The Rise of Open Source Licensing. A Challenge to the Use of Intellectual Property in the Software Industry.* Turre Legal Consulting, 2005.

[weba] Eclipse website. Eclipse Public License (EPL) Frequently Asked Questions.
Online: `http://www.eclipse.org/legal/eplfaq.php#CPLEPL`, accessed on May 23th 2007.

[webb] IBM website. Common Public License (CPL) Frequently Asked Questions. Online:
`http://www.ibm.com/developerworks/library/os-cplfaq.html`, accessed on May 22th 2007.

[webc] OpenSolaris website. FAQ: Common Development and Distribution License (CDDL). Online:
`http://www.opensolaris.org/os/about/faq/licensing_faq/#whatis`, accessed on May 22th 2007.

[Wika] Wikipedia. Common Public License.
Online: `http://en.wikipedia.org/wiki/Common_Public_License`, accessed on May 22th 2007.

[Wikb] Wikipedia. Free Software.
Online: `http://en.wikipedia.org/wiki/Free_software`, accessed on May 20th 2007.

[Wikc] Wikipedia. Open Source Software.
Online: `http://en.wikipedia.org/wiki/Open_source_software`, accessed on May 20th 2007.

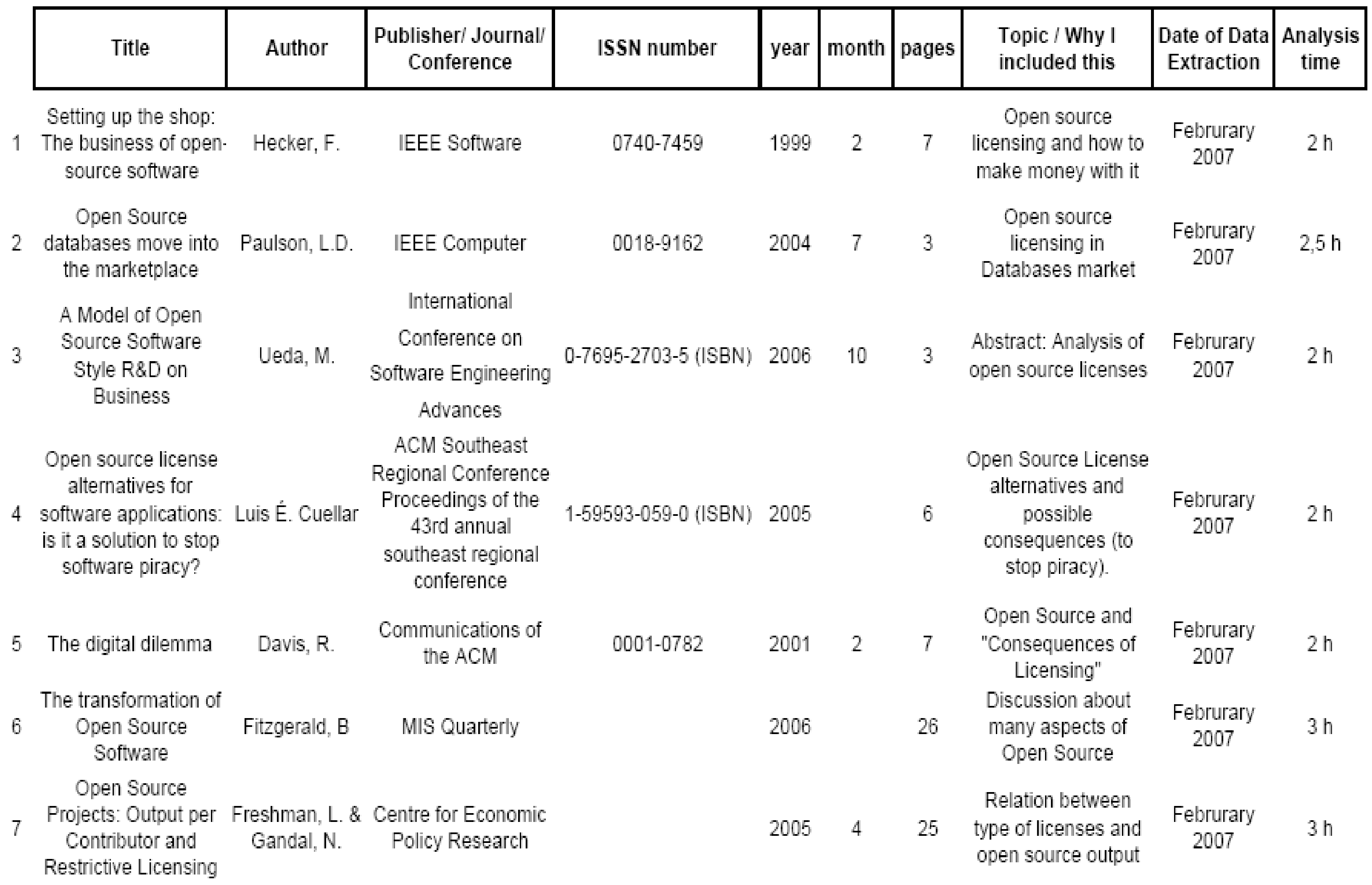
[WL01] Ming-Wei Wu and Ying-Dan Lin. Open Source Software Development: An overview. *IEEE Computer*, 2001.

# Index

# Appendix A

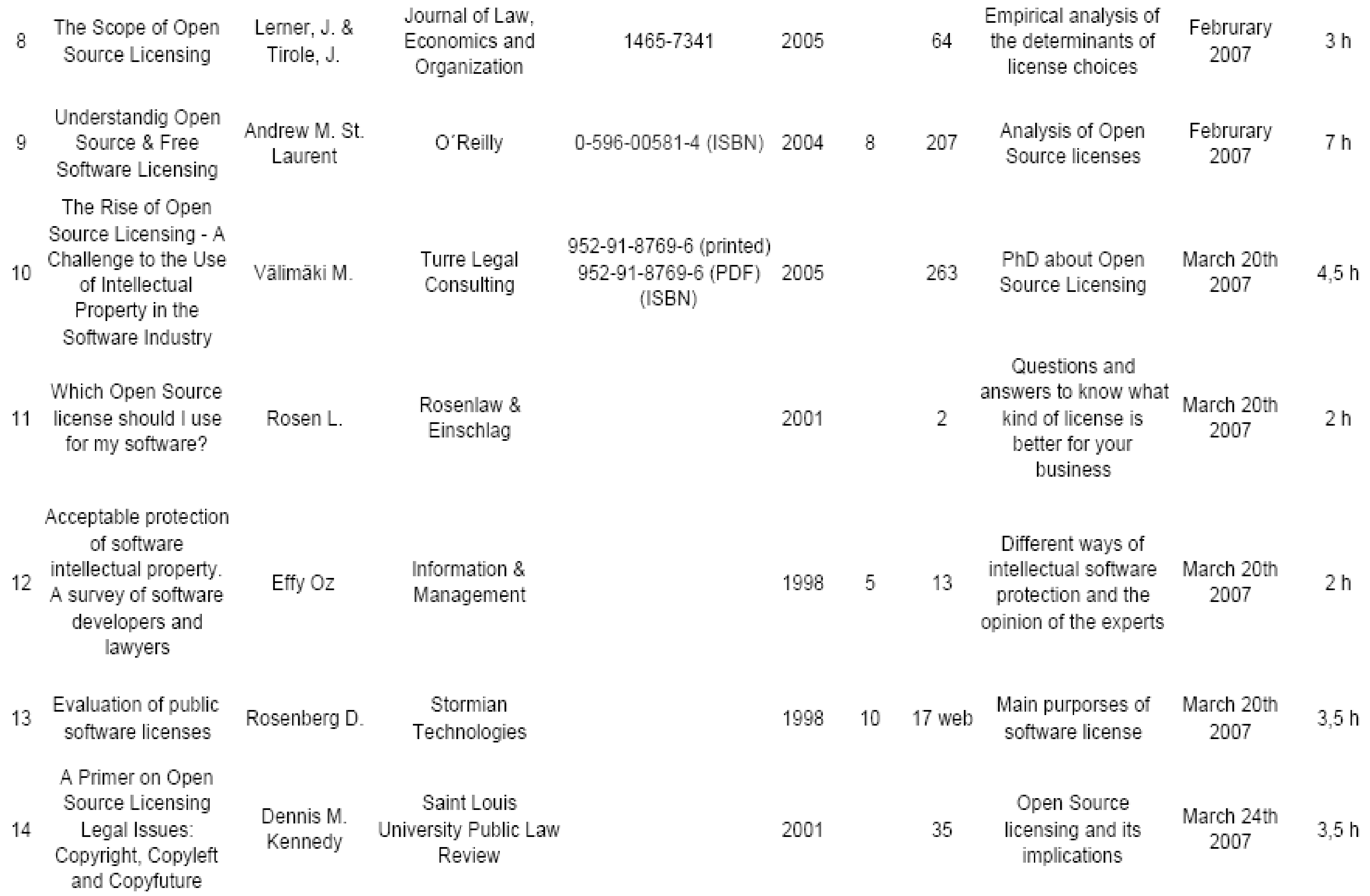
# Analysis and classification of the articles

The first column of each figure is an index to the summary of the article. For example, the first article “Setting up shop: the business of Open-Source Software” is summarized in the appendix B, section 1 (B.1).



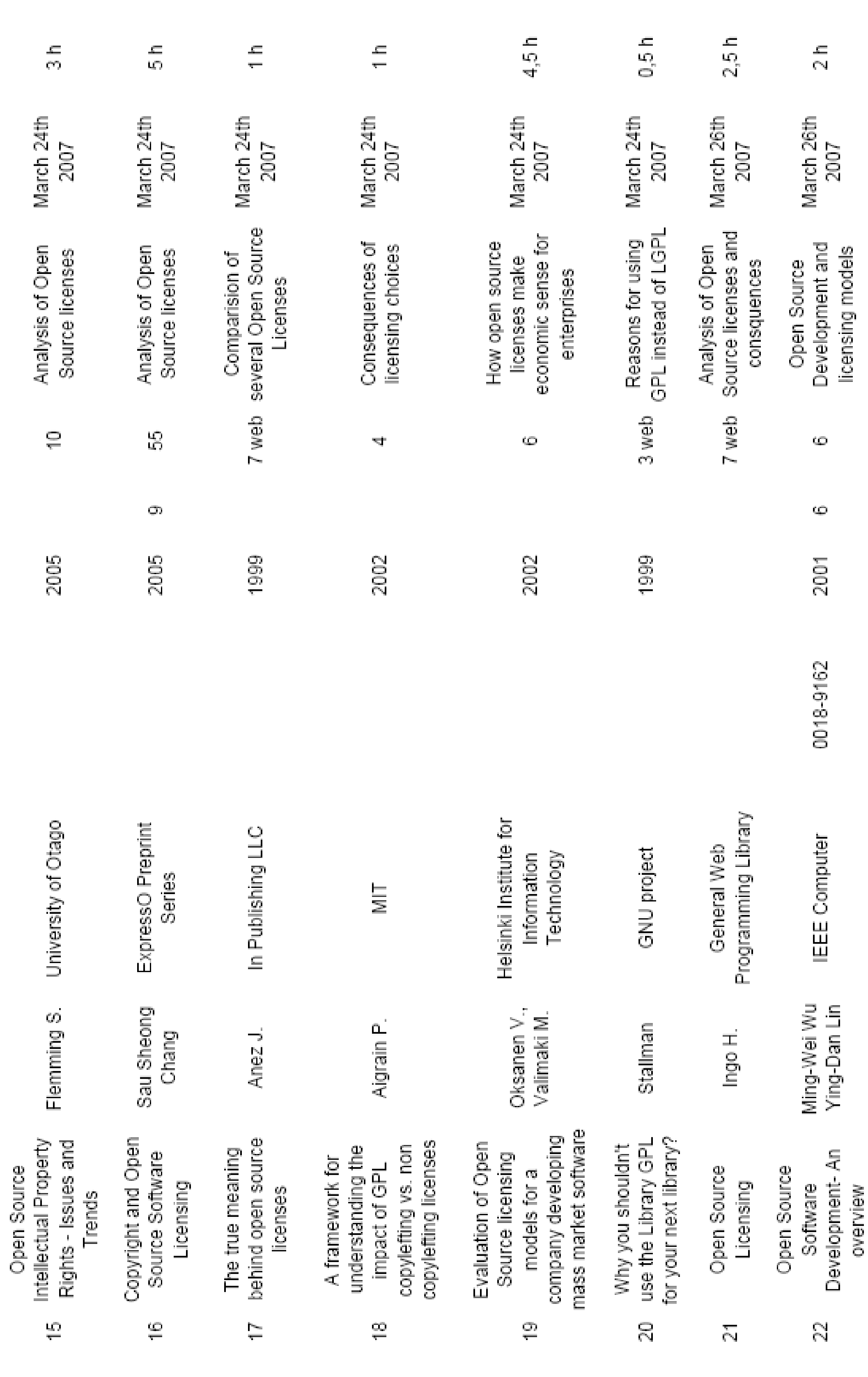
| | Title | Author | Publisher/ Journal/ Conference | ISSN number | year | month | pages | Topic / Why I included this | Date of Data Extraction | Analysis time |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Setting up the shop: The business of open-source software | Hecker, F. | IEEE Software | 0740-7459 | 1999 | 2 | 7 | Open source licensing and how to make money with it | Februrary 2007 | 2 h |
| 2 | Open Source databases move into the marketplace | Paulson, L.D. | IEEE Computer | 0018-9162 | 2004 | 7 | 3 | Open source licensing in Databases market | Februrary 2007 | 2,5 h |
| 3 | A Model of Open Source Software Style R&D on Business | Ueda, M. | International Conference on Software Engineering Advances | 0-7695-2703-5 (ISBN) | 2006 | 10 | 3 | Abstract: Analysis of open source licenses | Februrary 2007 | 2 h |
| 4 | Open source license alternatives for software applications: is it a solution to stop software piracy? | Luis É. Cuellar | ACM Southeast Regional Conference Proceedings of the 43rd annual southeast regional conference | 1-59593-059-0 (ISBN) | 2005 | | 6 | Open Source License alternatives and possible consequences (to stop piracy). | Februrary 2007 | 2 h |
| 5 | The digital dilemma | Davis, R. | Communications of the ACM | 0001-0782 | 2001 | 2 | 7 | Open Source and "Consequences of Licensing" | Februrary 2007 | 2 h |
| 6 | The transformation of Open Source Software | Fitzgerald, B | MIS Quarterly | | 2006 | | 26 | Discussion about many aspects of Open Source | Februrary 2007 | 3 h |
| 7 | Open Source Projects: Output per Contributor and Restrictive Licensing | Freshman, L. & Gandal, N. | Centre for Economic Policy Research | | 2005 | 4 | 25 | Relation between type of licenses and open source output | Februrary 2007 | 3 h |

**Figure A.1:** Classification of articles 1/3



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | The Scope of Open Source Licensing | Lerner, J. & Tirole, J. | Journal of Law, Economics and Organization | 1465-7341 | 2005 | | 64 | Empirical analysis of the determinants of license choices | Februrary 2007 | 3 h |
| 9 | Understandig Open Source & Free Software Licensing | Andrew M. St. Laurent | O´Reilly | 0-596-00581-4 (ISBN) | 2004 | 8 | 207 | Analysis of Open Source licenses | Februrary 2007 | 7 h |
| 10 | The Rise of Open Source Licensing - A Challenge to the Use of Intellectual Property in the Software Industry | Välimäki M. | Turre Legal Consulting | 952-91-8769-6 (printed) 952-91-8769-6 (PDF) (ISBN) | 2005 | | 263 | PhD about Open Source Licensing | March 20th 2007 | 4,5 h |
| 11 | Which Open Source license should I use for my software? | Rosen L. | Rosenlaw & Einschlag | | 2001 | | 2 | Questions and answers to know what kind of license is better for your business | March 20th 2007 | 2 h |
| 12 | Acceptable protection of software intellectual property. A survey of software developers and lawyers | Effy Oz | Information & Management | | 1998 | 5 | 13 | Different ways of intellectual software protection and the opinion of the experts | March 20th 2007 | 2 h |
| 13 | Evaluation of public software licenses | Rosenberg D. | Stormian Technologies | | 1998 | 10 | 17 web | Main purporses of software license | March 20th 2007 | 3,5 h |
| 14 | A Primer on Open Source Licensing Legal Issues: Copyright, Copyleft and Copyfuture | Dennis M. Kennedy | Saint Louis University Public Law Review | | 2001 | | 35 | Open Source licensing and its implications | March 24th 2007 | 3,5 h |

**Figure A.2:** Classification of articles 2/3



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 15 | Open Source Intellectual Property Rights - Issues and Trends | Flemming S. | University of Otago | | 2005 | | 10 | Analysis of Open Source licenses | March 24th 2007 | 3 h |
| 16 | Copyright and Open Source Software Licensing | Sau Sheong Chang | ExpressO Preprint Series | | 2005 | 9 | 55 | Analysis of Open Source licenses | March 24th 2007 | 5 h |
| 17 | The true meaning behind open source licenses | Anez J. | In Publishing LLC | | 1999 | | 7 web | Comparision of several Open Source Licenses | March 24th 2007 | 1 h |
| 18 | A framework for understanding the impact of GPL copylefting vs. non copylefting licenses | Aigrain P. | MIT | | 2002 | | 4 | Consequences of licensing choices | March 24th 2007 | 1 h |
| 19 | Evaluation of Open Source licensing models for a company developing mass market software | Oksanen V., Valimaki M. | Helsinki Institute for Information Technology | | 2002 | | 6 | How open source licenses make economic sense for enterprises | March 24th 2007 | 4,5 h |
| 20 | Why you shouldn't use the Library GPL for your next library? | Stallman | GNU project | | 1999 | | 3 web | Reasons for using GPL instead of LGPL | March 24th 2007 | 0,5 h |
| 21 | Open Source Licensing | Ingo H. | General Web Programming Library | | | | 7 web | Analysis of Open Source licenses and consquences | March 26th 2007 | 2,5 h |
| 22 | Open Source Software Development- An overview | Ming-Wei Wu Ying-Dan Lin | IEEE Computer | 0018-9162 | 2001 | 6 | 6 | Open Source Development and licensing models | March 26th 2007 | 2 h |

**Figure A.3:** Classification of articles 3/3

# Appendix B

# Summaries of the articles

## B.1 Setting up shop: the business of Open-Source Software

*Frank Hecker*
[Hec99]

The author of this article wants to show how companies can take advantage of giving free software to their customers instead of selling it in the traditional way.

Reading this article, we notice that OSS has many advantages and very few disadvantages. Using examples such as Netscape or Linux (and GNU-project), the writer encourages companies to follow similar strategies, since it is important to know how to motivate and coordinate developers without exerting coercion.

Although you cannot get revenues using traditional software licenses and fees, you can find other ways of generating revenues and profits based on the value you are providing to customers:

- Support Sellers: Revenue comes from media distribution, branding, training, consulting, custom development, and post-sales support.
- Loss Leader: A no-charge Open Source product is used as a loss leader for traditional commercial software.
- Accessorizing: A company distributes books, computer hardware, and other physical items associated with and supportive of Open Source Software.
- Service Enabler: Open Source Software is created and distributed primarily to support access to revenue-generating online services.

- Brand Licensing: One company charges other companies for the right to use its brand names and trademarks in creating derivative products.
- Sell It, Free It: A company's software products start out their product life cycle as traditional commercial products and then are continually converted to Open Source products when appropriate.
- Software Franchising: This combines several of the preceding models (in particular Brand Licensing and Support Sellers). A company authorizes others to use its brand names and trademarks in creating associated organizations doing custom software development in particular geographic areas or vertical markets. The company might also supply franchises with training and related services in exchange for franchise fees of some sort.

The text also provides you few lessons of how implement an OSS strategy talking about: code sharing, third-party technology, code sanitization, encrypt control and product development process.

From the point of view of F. Hecker, "Open Source Software is a "new" business tool that offers the potential to achieve results that are impossible with traditional software development practices alone".

## B.2 Open source databases move into the marketplace

*Linda Dailey Paulson*
[Pau04]

L.D. Paulson, the author of this article, shows several aspects of OSS Databases, including development, organizational models, cost and licensing advantages.

Furthermore, the reader can see in the article that the adoption of OSS Databases by enterprises is increasing in a significant way.

To end, we must say that Jeff Jones, director of Strategy for IBM Software Group's Information Management Solutions Unit, recognizes that "the Open Source Movement is getting companies to think about cost effectiveness".

## B.3 A model of Open Source Software Style R&D on business

*Masashi Ueda*
[Ued06]

Masashi Ueda analyses OSS licenses by cluster analysis and finds the importance of standards. In his analysis, we can see that over 70% of OSS projects registered on SorceForge.net adopt the GPL. Due to these results, the author deduces that perhaps it is important to have a standard. He demonstrates mathematically his hypothesis showing the reduction of cost achieved using an OSS style R&D instead of independent R&D.

## B.4 Open source license alternatives for Software Applications

Is it a solution to stop software piracy?

*Luis E. Cuéllar*
[Cue05]

Luis É. Cuellar explores licensing options available for OSS and analyzes the impact of theses licenses on commercial applications. He also suggests that the OSS is a legal alternative to software piracy and software license infringements since the motivation behind the illegal duplication of copyrighted software works could be removed completely.

## B.5 The Digital Dilemma

*Randall Davis*
[Dav01]

R. Davis, a professor in the computer science department at MIT, shows in his article the problems that intellectual property laws can produce.

The dilemma arises because of the easy mechanisms for reproducing digital information. The author proposes some suggestions that can help to reduce intellectual piracy, showing in the same time the strengths and weaknesses of each solution:

- Technological solution: Encryption, but strong encryption requires substantial costs.
- Business models: think in new business models:
  - Give away the product; make money from an auxiliary service.
  - Give away the product; sell updates.
  - Give away one piece that promotes another (i.e. Adobe Reader).
  - Offers extreme customization.
  - Offer a mass-market product at a low price and high volume, along with frequent improvements.
- Multiple perspectives: look the problem from several views (law, technology, market, economics, psychology and public policy).

However, any of these solutions are not the panacea. R. Davis says that "Authors and publishers will continue to do their work in the presence of some unauthorized reproduction" and emphasizes the difficulty of finding ways to motivate individual creation at the same time that we preserve benefits of creation for the common good.

## B.6 The transformation of Open Source Software

*Brian Fitzgerald*
[Fit06]

Throughout this paper, Brian Fitzgerald wants to show us the new movement of OSS that he defines as OSS 2.0. There are substantial differences between its free software antecedent (FOSS) and the already mentioned OSS 2.0.

The OSS 2.0 searches a balance between the Open Source values and the way of making money with value-added services related to Open Source projects. Actually, OSS 2.0 looks for a high-quality product that can be available for all type of end-users meanwhile in the FOSS projects, FOSS developers were also the users of the software developed.

In OSS 2.0, the development process becomes less bazaar-like since the strategic planning becomes very important. Analysis and design are much more planned and there are developers who are paid for their contribution on OSS.

More sophisticated business models have appeared. Customers will want a professional service and they will pay for it. In FOSS, support was obtained from forums or mailing lists and, sometimes, from a competent third-party provider.

Licensing is also an important matter in OSS 2.0. Actually, proprietary companies have realized of the importance of adopting an OSS strategy and they try to produce licenses in order to comply with the OSS definition.

## B.7 Open Source Projects: Output per Contributor and Restrictive Licensing

*C. Fershtman and N. Gandal*
[FG05]

The authors of this paper examine how the type of license and other factors affect output per contributor in Open Source projects. They make a division between restrictive licenses, “moderately restrictive” licenses and “nonrestrictive” licenses. They also argue that the different type of licenses may provide different incentives for developers to participate in Open Source projects and to exert effort. They demonstrate it concluding that “the output per contributor in Open Source programs is much higher when licenses are less restrictive. The results also suggest a status/signaling motivation for participation in Open Source projects with restrictive licenses”.

It is also important to say that this paper mentions Lerner and Tirole (2005) study to expose that “open projects that run on commercial operating systems and projects that are designed for developers tend to use less restrictive licenses, while projects that are targeted for end users tend to use more restrictive licenses”. Furthermore, “on average, firms that employ software with restrictive licenses supply fewer proprietary products than firms that employ software with less restrictive licenses”.

## B.8 The Scope of Open Source Licensing

*Josh Lerner and Jean Tirole*
[LT05]

This paper presents an empirical analysis of the determinants of license choices using the SourceForge database (a compilation of nearly 40.000 Open Source projects). Throughout this analysis, the authors prove that:

- Projects geared toward end-users tend to have restrictive licenses, while those oriented toward developers are less likely to do so.
- Projects that are designed to run on commercial operating systems and whose primary language is English are less likely to have restrictive licenses.
- Projects that are likely to be attractive to consumers -such as games- and software developed in a corporate setting are more likely to have restrictive licenses.
- Projects with unrestricted licenses attract more contributors.

## B.9 Open Source & Free Software Licensing

*Andrew M. St. Laurent*
[St.04]

Not summarized – book

## B.10 The Rise of Open Source Licensing - A Challenge to the Use of Intellectual Property in the Software Industy

*M. Välimäki*
[Väl05]

Not summarized – book

## B.11 Which Open Source license should I use for my software?

*Lawrence Rosen*
[Ros01]

Lawrence Rosen, an attorney for Open Source companies and projects, writes several questions that a company should ask itself in order to decide what kind of license is better for its business. The questions and answers are:

- Do you intend to make money from licensing the software or from providing ancillary services like installation and training?

  Answer: proprietary software.

- What degree of freedom are you willing to grant to your licensees to modify your software?

  Answer:

  - BSD-type imposes virtually no restrictions on licenses; they can modify the licensed software and create proprietary versions without restriction.
  - GPL-type rquires the licensee's modifications to be licensed back under the same license; it is good for the community contribution.
  - MPL-type imposes an intermediate level of freedom; modifications to individual files containing licensed code must be licensed back; but new files that simply work with the licensed code need not to be.

- Are you willing to grant warranties that the software will be "merchantable" or "fit for a particular purpose"?

  Answer: If the software is royalty-free, you probably can't afford a warranty. However, you can charge your Open Source Software to provide warranties and other services.

- Is your software so well known that the main asset you need to protect is your trademark rather than your code?

  Answer: Apache license is an excellent example. You are allowed to do almost everything with the Apache code, but you will have to change the name. If you have a trademark to protect, make sure your license contains appropriate terms relating to that.

- Have you considered the possibility of dual licensing?

  Answer: you may want to license your software under GPL and simultaneously provide a proprietary version for those of your customers who are afraid of the GPL's inheritance features; this can be treated as a revenue opportunity.

- Have you considered using different licenses for different parts of your software?

  Answer: client software might be distributed under an MPL-like license, but server software might be distributed under a proprietary license. That way, you could make money from the bigger customers that will pay to license your server software, and simultaneously build a large customer base with free clients.

- What is about your software that you are really trying to protect? Is it the code itself, or the standards that are implemented using that software?

  Answer: A license like SISSL allows anyone to develop modifications of licensed software as long as the licensee complies with all requirements set out by a standards body; a licensee who elect not to comply with the specification must be publish a royalty-free reference implementation of the modifications so that the standard cannot be abducted by another company.

- Are there any patents that relate to your software?

  Answer: if so, you will have to consider licensing your patents along the code.

## B.12 Acceptable protection of software intellectual property. A survey of software developers and lawyers

*Effy Oz*
[Oz98]

This article reports the results of a survey on optimal legal way to protect developer's rights to their intellectual property in the US. It exposes the differences between copyright and patents and it also shows the preferences of software developers and attorneys. Even the differences, the conclusion is that both groups would prefer a special law for software intellectual property, the scope of which should be broader than that of copyrights but narrower than that of patents.

## B.13 Evaluation of Public Software Licenses

*Donald K. Rosenberg*
[Ros98]

Donald K. Rosenberg introduces free software licenses. He does not go into the license in detail; he just wants to show the main purposes of each license and its results. The author also discusses about the kind of licenses that different software (applications, toolkits, libraries and Operating Systems) that interactive should have. Finally, he also discusses about forking and its causes.

The GNU General Public License (GPL) gives the users three rights:

- to copy the software and to give it away
- to change the software
- to have access to the source code

You can also charge for distribution costs of the program and source code. Proprietary products that simply link to GPL software are allowed to remain proprietary; only derivative works need to be licensed under GPL.

There is a loophole for commercial software: the LPGL:

- it protects GNU libraries from being incorporated into proprietary software
- it protects proprietary libraries from the viral effect of the GPL

The BSD license allows you to take the source code proprietary.

The Free Edition License from Qt is like the GPL, except that:

- no modifications are permitted to the Qt library
- no commercial distribution is permitted

Looking at the license details, under the Qt Free Edition License, the developer could:

- write his own free (or Open Source) license allowing free distribution and modifications, or
- use the GPL or the LGPL

However, there is an immediate problem: under the GPL, you cannot distribute the Qt library with your application because the Qt library is itself not under the GPL, so you cannot combine the code. If you want to use the BSD license, you have also problems, since BSD allows you to distribute the code only in binary form and it is not allowed by the Qt Free Edition License.

Commercial users must buy the Qt toolkit under the Professional Edition License. That way, they obtain the Qt toolkit and the right to distribute unlimited runtimes.

No modifications of the toolkit are permitted. Therefore, forking is prevented.

Moreover, we could call the Qt Free Edition License a “Change-of-State” License, since if your application changes its status from free to commercial, it needs a different license.

Aladdin’s Ghostscript splits itself into different free/commercial versions. There is the Aladdin Ghostscript commercial version. The free versions are the Aladdin Free Public License and GNU Ghostscript. The Aladdin Free Public License resembles the GPL and has additional restrictions: you cannot accept money for the free program except for cost of disks and copying.

Scriptics, which commercializes the Tcl/Tk toolkit, pays its in-house developers workers with the benefits of the commercial version. However, it wants to attract new users through the free version.

The Apache License says:

- no source code distribution requirement
- you can combine Apache code with other code

The Mozilla Public License (MPL) divides software into files, which are separated into:

- the Open Source part (called “Covered”) and
- anything the user adds

The arrangement allows the developer to add his own files and distribute them with the Covered files, provided he does not modify the Covered files. If he does modify the Covered files, then he must distribute those modified files under Open Source MPL rules.

Under the Artistic License, you can modify the source code and take it private if you also supply the standard version of Perl with yours; or, if you are careful to rename modified files so that the standard version still works.

## B.14 A Primer on Open Source Licensing Legal Issues: Copyright, Copyleft and Copyfuture

*Dennis M. Kennedy*
[Ken01]

Dennis M. Kennedy briefly explains the fundamentals of Open Source, how the Open Source licensing works and its implications. He also makes a historical introduction of the Open Source Movement and he poses some questions about possible problems or controversies of the Open Source way of licensing. The definition of Open Source and the analysis of the four main categories of licenses are also included.

### The Open Source Definition

Open Source is characterized by:

1. **Free Distribution.** "The license may 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 may not require royalty of 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 downloading the source code, without charge, via the Internet. The source code must be the preferred form in which a programmer would modify the program."

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."

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."
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."
9. **License Must Not Contaminate Other Software.** "The license must not place restrictions on other software that is distributed along with the licensed software."

#### Public Domain, Shareware and Freeware

Some software is released freely into the public domain and the author has no copyright rights, whereas Open Source does not. Open Source software is governed by a license and the owners of the copyright in the software continue to own the copyright and assert their rights thereto. In Open Source Software, people receive a software license that give them more rights than they have become accustomed to expect under commercial software licenses.

Freeware or shareware often has the connotation of being "free". They are free because of the price, not because of the availability of their source code. Actually, their licenses usually have many of the restrictions found in commercial licenses.

#### Taxonomy of Licenses

The Open Source licenses all have in common a requirement that source code be made available and that users of the software have the right to make derivative works. The licenses all disclaim warranties and many make an effort to limit liabilities. However, we can distinguish four main categories. The main difference among them is their permissiveness and whether derivative works can be proprietary.

#### The GPL Licenses

The important feature of the GPL is that it does not allow for GPL software to be mixed with non-GPL software without making the non GPL-software also subject to the terms of GPL. The GPL does not allow modifications to be taken private. Under the GPL, source code must be released and derivative works must be also released under the terms of the GPL. If a licensee distributes modifications of a program, source code must be made available and there is to be no fee other than copying and similar charges.

### The BSD Licenses

The least restrictive of the Open Source licenses. Under the BSD licenses, distribution of source code is permitted but not mandated for derivative works. Therefore, programs under the BSD Licenses can be combined with proprietary software. The BSD licenses allow redistribution and use of source code and object code with or without modification so long as the redistribution of source code retain required copyright and other notices and the disclaimer of warranties and limitation of liability clauses. The original BSD License had certain attribution requirements, including mandatory attribution of naming of contributors in advertising of software using the code.

The BSD Licenses are considered by many to be more “free” than the GPL because they permit developers to release derivative works under whatever license they prefer. In other words, the BSD Licenses do not contain copyleft terms. This is why the BSD type of licenses are attractive to commercial developers.

### The Mozilla Public License and Related Commercial Licenses

Under the Mozilla Public License, commercially licensing derivative works is permitted. Changes to covered program sources must be made freely available to anyone. The Mozilla Public License does not contain the copyleft provisions of the GPL. Additions of covered source code that form a “larger work” may be licensed differently and published or not even published at all.

### Other Open Source licenses

Variations of the above mentioned.

### Choosing an Open Source License

As a practical matter, there is an encouragement to use the GPL, the BSD Licenses, or the LGPL. In other specialized cases, such as the Apache software or Perl, certain licenses and approaches of licensing have been used for a long period of time and it is better to use the licenses that are commonly used in the area of software programming, in part due to the level of comfort with the licenses in the programmer community.

In general, anyone wanting to release software under an Open Source license must spend a significant period of time becoming very familiar with the licenses, learning the strengths and weaknesses for the particular situation, and determining and whether releasing under an existing Open Source license is acceptable or whether to start the process of developing a new license that fits the Open Source Definition and is also attractive to programmers. Obviously, Open Source developers will be more comfortable working on standard known licenses rather than trying to learn the nuances of a customized license agreement for a particular project. As a practical matter, only large software players as IBM and others will have the clout to create new type of licenses.

#### Multiple Licensing

It is permissible for a copyright owner to release source code under an Open Source license and to release a proprietary version of the software under the standard commercial licenses in order to make money.

#### Ownership

In Open Source development, the general principle is that the author of the code remains the copyright owner and he simply applies the Open Source license to his code.

#### Controversial Issues

GPL and other licenses do not provide that the licenses are perpetual. This omission may cause problems at a later point when a court determines that an Open Source license with respect to a program has expired because the “reasonable” duration is over.

The fact that the Open Source licenses are not signed and not negotiated may arise questions.

The potential applicability of consumer protection laws may also have an impact on warranties for Open Source warranties.

Developments in copyright law and, in particular, UCITA[1], can have important unintended consequences for the Open Source licenses.

## B.15 Open Source Intellectual Property Rights - Issues and Trends

*Stewart Fleming*
[Fle04]

This article examines Open Source licenses and their impact on copyright and intellectual property rights and the associated legal risks. In addition to this, there is an introduction about the Open Source phenomenon movement and the motivational factors governing Open Source participation.

### Motivational factors

- Individual
  - Intellectual challenge
  - Desire to improve the greater good
  - Expectation of improved employment prospects
  - Desire for fame and associated publicity
- Academic Institution
  - Costs accounted for
  - Expectation of greater prestige
  - Encouragement of staff development
  - Outlet for intellectual capital of the organization
- Commercial Enterprise
  - Excess of R&D expertise/capability
  - Reduction in operational cost by benefiting from collaborative activity
  - Outlet for intellectual capital of the organization

| License | OSI Compatible | GPL Compatible | Redistribution | Charging | Copyleft |
|---|---|---|---|---|---|
| GPL | ✓ | ✓ | Licensee must guarantee availability of entire code | Licensee may not charge for the code or royalties, but may charge for distribution cost | Derived work is subject to GPL |
| LGPL | ✓ | ✓ | As above | As above | × |
| BSD | ✓ | ✓ | No restriction | No restriction | × |
| MIT | ✓ | ✓ | No restriction | No restriction | × |
| MPL | ✓ | ✓ | Licensee must guarantee availability of source code for at least 12 months after first availability | Royalty-free | ✓ |

**Table B.1:** Classification of restrictions imposed by various licenses

### Licenses

BSD - Open Source license with few restrictions and no impact on derived works. It requires only that attribution of copyright be made in the distributions of the software. It specifically excludes any software warranties and disallows the use of the original organization in any advertising or promotion of derived works.

MIT - Open Source license with few restrictions and no impact on derived works. It requires only that a copyright notice be included with copies or substantial extracts of the software and excludes warranties.

The risk with unrestricted licenses such as BSD or MIT models is that a licensee can produce a derived work and not release improvements or enhancements, which

[1]The United States Uniform Computer Information Transactions Act (UCITA) is a proposed law to create a clear and uniform set of rules to govern such areas as software licensing, online access, and other transactions in computer information. In particular, UCITA attempts to clarify and/or codify rules regarding fair use, reverse engineering, consumer protection and warranties, shrinkwrap licenses, and their duration as well as the transferability of licenses. http://en.wikipedia.org/wiki/Uniform_Computer_Information_Transactions_Act

might be useful to the wider community.

Mozilla Public License - include clauses that are intended to deal with the software patent issue where source code that infringes on a software patent is deliberately or inadvertently introduced to a project.

GNU GPL - incorporate the copyleft provision. Derived works are licensed under the same license.

LPGL - no copyleft provision. This is a free software license option available to commercial software developers without the obligation to release all of their source code in derived works.

### Critical issues and trends

The article concludes exposing some critical issues and trends about Open Source:

1. The exclusion of warranties for software defects in most Open Source licenses should cause organizations considering the adoption of Open Source Software to carefully consider how quality and reliability can be assured.
2. Since the enhancement of reputation is an important motivating factor in OSS development, software authors might benefit from more uniform international recognition of their right to assert authorship and their right to avoid derogatory treatment as author of a work.
3. Quality and reliability characteristics of Open Source Software raise concerns for organizations in areas where certification is needed such as in mission-critical activities or medicine. The lack of formal tools for testing should lend a note of caution to those considering the use of Open Source Software.
4. An organization that has been granted a software patent for some algorithm or implementation is granted the rights to charge royalties for use, or may force others to cease distribution of software that employs the scheme covered by the patent. Open Source Software is vulnerable for this form of restriction since all source code is publicly-available.
5. Different interpretation of derivative laws for different courts
6. Peer-review of public software is an advantage, but successful outcomes still depend on the motivation of properly-skilled individuals to methodically study, probe and fix Open Source Software problems.

## B.16 Copyright and Open Source Software Licensing

*Sau Sheong Chang*
[Cha05]

This paper describes the past and present of legal software protection and traces the history of the Open Source Software Movement. The Open Source Licenses are compared and explained and some questions about the legal enforceability of them are also posed.

### GNU Licenses

The GPL license allows the software be distributed and modified without additional permission from the licensor. The license ensures that the licensees are aware that the software is distributed without warranty. The license frees the software from restrictive patents.

GPL prevents any software licensed under GPL to be combined with other piece of software not licensed under GPL or not compatible with GPL.

GPL also encourages separate written agreements between two parties to establish warranties or contracts for maintenance, as one of the business models in Open Source is the provision of warranties and software maintenance.

LGPL specifically allows LGPL-licensed software libraries to be linked with non GPL- licensed software, including closed sourced software. However, GNU Project encourages the use of GPL.

### Open Source Definition

The main concepts are:

- Free redistribution
- Access to source code
- Open modification of the source code
- No discrimination against persons or group of persons
- No discrimination against any fields of endeavor
- Be technology-neutral

### The MIT and BSD License

The MIT License basically grants all of the rights of a copyright holder including the exclusive right to commercially exploit and create derivative works. The only two conditions imposed are that the copyright and permission notices must be included in the copies of the software and a general disclaimer of warranty.

The BSD License is only slightly more restrictive: the name of the organization that created the software or it contributors cannot be used to endorse or promote the software without prior written permission.

The MIT and BSD licenses are OSI-Certified licenses and GPL-compatible licenses, although the original BSD license is not GPL-compatible.

### The Apache License

The older Apache License version 1.1 is very similar to the BSD-License, but includes a requirement for the acknowledgement of the creator's contributors of the software. The Apache License version 2.0 is more complex. It includes provisions for patent rights granted by the license and the use of other licenses for derivative software. The Apache License version 2.0 also explicitly defines 'Contributions' that are special modification of the software provided to the licensor of the software for its inclusion into the original one. If accepted, the modifications will become part of the original software and will fall under the same license.

The Apache licenses are OSI-certified but are not GPL-compatible.

### The Artistic License

The Artistic License was designed to maintain control over the Perl project while encouraging participation in the project and innovation outside the project. The Artistic License is ambiguous, self-contradictory and virtually impossible to interpret. One problem with the Artistic License is that although it prohibits sale of the software, it also allows an aggregate distribution of the Artistic Licensed software with another piece of software. Interpreted literally, someone can defeat the license by merely including a trivial piece of software together with the licensed software.

### The Mozilla Public License

The Mozilla Public License can be regarded as a hybrid of ideas between the GPL and the MIT/BSD licenses. MPL-licensed code can be combined with code under another license. MPL is not a GPL-compatible license but it is OSI certified.

The MPL divides a software work into an Open Source part (called "Covered Code") and anything a contributor adds. The arrangement allows any developers to add his own files and distribute them with the covered code, provided he does not modify the covered code. However, if he does modify the covered code, he must distribute the modified code under MPL.

The MPL is considered one of the better drafted Open Source licenses and is used in many Open Source projects including the popular Firefox browser.

| **Freedoms or Restrictions** | Public Domain | MIT/ BSD | Apache 1.1 | Apache 2.0 | Artistic | MPL 1.1 | GPL | LGPL | Closed Source |
|---|---|---|---|---|---|---|---|---|---|
| Has copyright owner | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Copyright acknowledgement | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Freely copy and use as-is | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Distribute modified versions with the same license | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Distribute modified versions under the same license | ✓ | ✓ | ✓ | ✓ | ✓ | × | × | × | × |
| Link with code under different license | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | × |
| Must include source code in the distribution | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| Grants licensee patent rights | × | × | × | ✓ | × | × | ✓ | ✓ | × |
| Disclaimer of warranty/limitation of liability | × | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Non-endorsement provision | × | × | ✓ | ✓ | × | ✓ | × | × | NA |
| Reciprocity obligations for derivative works (copyleft) | × | × | × | × | × | ✓ | ✓ | ✓ | × |
| Number of words in license document (complexity of the license) | NA | 167 / 222 | 294 | 1581 | 771 | 3666 | 2956 | 4020 | Varied |

**Table B.2:** Free and Open Source license rights matrix

| **License name** | **Quantity** | **Percentage** |
|---|---|---|
| GNU General Public License | 45101 | 69% |
| GNU Library or Lesser General Public License | 7388 | 11% |
| BSD License | 4724 | 7% |
| Artistic License | 1230 | 2% |
| MIT License | 1195 | 2% |
| Apache Software License v1.1 | 968 | 1% |
| Mozilla Public License 1.1 | 827 | 1% |
| Common Public License | 503 | 1% |
| Apache License V2.0 | 452 | 1% |

**Table B.3:** Free and Open Source license popularity in Sourceforge

### Open Source Licenses - copyright license or contract?

If Open Source licenses are copyright licenses or legal contracts is yet to be determined. Although both legal contracts and copyright licenses have different enforcements, one which is thoroughly covered by the contract itself, and the other is through copyright legislation, both enforces the same terms and conditions on the licenses. However, one noticeable difference is that without a legal contract, licensors can revoke their licenses at any point in time, subject to equitable rules. This has some serious repercussions if the software is already well known in the market as the licensor is not obliged to continually provide the software under the same license.

In addition to this, it is not clear if all Open Source licenses can be considered valid contracts but similar to closed source licenses have been accepted as valid contracts albeit controversially. Interestingly, if a contract does not exist for Open Source licenses, sometimes copyright laws of certain countries impose a harsher criminal offence on copyright infringers; therefore it would seem that it is to the benefit of the licensee not to use this as a defense against enforceability of Open Source Software licenses.

### Cross-jurisdictional issues in Open Source licenses

A major issue with the copyright is the issue with the applicability of the law in different countries. Copyright laws are territorial and generally do not cross borders.

Problems:

1. Most Open Source licenses are written in English and in fact assume certain facts that are only applicable in US laws. However, in many countries, there are laws that mandate the use of the national language for legal documents including licenses and contracts.

2. Legal background which copyright law is derived from. In countries that derive laws from English legal system, copyright arose from the economic rights of copywriters and publishers while most continental European countries derive copyright from the concept of droit d'auteur, which focuses on the moral rights of the original author.

3. Warranties and disclaimers. In certain countries, especially European countries, general disclaimers are not valid in a contract due to provisions for unfair terms in contracts.

4. The possible different interpretation of derivative works across different countries.

## B.17 The true meaning behind Open Source Licenses

*Juancarlo Añez*
[Ane99]

J. Añez analyzes and compares briefly several Open Source licenses. Before that, he mentions the public domain and explains that any work placed in the public domain has the same legal status as one for which the legal copyright term has elapsed: you can include public domain software in your own work, modify it, copy it and talk about it in publications and conferences.

The least restrictive licenses (BSD and MIT) reserve only the right to disposition, but they grant ample rights of use, creation of derivative works and redistribution. One of the most restrictive licenses (GPL) force derivative works to be licensed under GPL. Another restrictive license, the Java Community Source License, restricts the use that can be made of the software (free use to research); other uses require that compatibility tests be made and royalties be paid. The XML4J Evaluation License restricts the use to that which is "lawful and non-commercial". Surprisingly, there are also licenses that completely prohibit executing the software.

If you want to distribute your software commercially but do not want to distribute the source of the modifications you made, use a license such as the MIT or the

BSD one. GPL and LPGL are good choices if you don't mind distributing your source code. Licenses like the Java Community Source License are so complicated, and it's better to contact a specialized lawyer.

## B.18 A framework for understanding the impact of GPL copylefting vs. non copylefting licenses

*Philippe Aigrain*
[Agr02]

This paper discusses the consequences of licensing choices, especially, for publicly funded software. It argues that GPL copylefting serves the public good especially for software that can play a critical role in the activities of the information society. So, in publicly funded efforts, copylefting licenses should clearly be preferred. However, for private efforts, the choice is not clear: it will depend on whether companies want short term benefits in terms of take-up or the user trust effects of copylefting, as well as the long term sustainability of this approach. In practice, companies that choose non-copylefting licenses may take the risk of getting proprietary competitors. Therefore, dual licensing that clearly establish two types of dissemination can be the solution.

## B.19 Evaluation of Open Source licensing models for a company developing mass market software

*Mikko Välmimäki and Ville Oksanen*
[OV02]

The target of this article is to explain when and how Open Source licenses make economic sense for enterprises. To do that, the article contains a preliminary evaluation of several frequently used Open Source licenses and licensing models from the perspective of a company developing mass market software products for competitive markets.

The article starts explaining what the most popular licenses are and the relevant economic attributes describing mass market software.

| **License** | **GNU GPL** | **LGPL** | **BSD/MIT/PD** | **Commercial** |
|---|---|---|---|---|
| **Type** | Persistent and viral | Persistent | All permissive | All restrictive |
| **Popularity** | 67% among OS | 10% among OS | 12% among OS | NA |
| **Derivate works** | Only GPL | GPL or LGPL | No restrictions | Not allowed |
| **Bundling** | Only GPL | No restrictions | No restrictions | Restricted |
| **Patenting** | Free licensing required | Free licensing required | Not covered | Restricted |

**Table B.4:** Most popular mass market licenses

## Ownership of rights

Afterwards, there is a discussion about rights and ownership. It is important to know who the ownerships of rights is since it allows companies to price the software, change its licensing policy and distribute with different licenses. However, Open Source licenses could "dilute" the ownership and even eliminate the possibility of relicense the software.

Who is the author? The article makes a distinction between:

- Distributed incremental development with no coordination. In this case, every contributor has copyright to his contribution (bundled work and authorship)
- Focused and centrally controlled development. In this case, every contributor has copyright to the work as a whole (joint authorship)
- Complete rewriting of existing works. In this case the rewriter(s) have copyright to the new work overriding all previous copyrights (new authorship)

However, with distributed development, there is the employment relationships problem. According to many national laws, the employer owns automatically all copyright and therefore the employee cannot license his work without permission of his employer. Consequently, software under a persistent license may actually infringe some third party company's copyright without anyone's consent. (The employer cannot own the copyright of software having a persistent license).

Therefore, a company should obtain all rights to the product it wishes to license and make sure there are no hidden liabilities in code contribution from unknown third parties (Rights clearing). Alternatively, under all permissive licenses the copyright ownership does not restrict any successive third party. It is necessary only to make a little modification to the software in order to license it with new terms as a whole.

There are two ways to clear rights:

- Rewrite the software. It is legally the safest way.
- Obtain rights with a license term of a specific contract. There are legal risks possible.

*Licenses combinations:* A specific problem with GNU GPL licenses is that it is incompatible with many other licenses.

| **License** | **IBM public License** | **Mozilla Public License** | **Sun Industry Standards Source License** | **Nokia Open Source License** |
|---|---|---|---|---|
| **Type** | All permissive | Persistent and viral | Persitent and viral | Persistent and viral |
| **Typical use** | Open Sourced programs from IBM | Mozilla-project | OpenOffice | Research projects funded by Nokia |
| **GNU GPL-compatible** | No | No | No | No |

**Table B.5:** Company specific licenses and GNU GPL

## Development process

The articles also talks about the implications of the license choice to development.

Licenses should give incentives for developers and they should also give possibilities for controlling the project.

Permissive licenses allow the development results to be commercialized by any third party without any compensation paid to developers.

The main possibilities to control the development process are:

- Closed Source controlled by license
- Open Source controlled by leadership

### Product distribution

The market focus on software should be perhaps the most important factor affecting the license selection.

If a product is aimed at end-users, the benefits from choosing an Open Source license are little if none. Therefore, the license terms can be restrictive. The situation is slightly different if the target platform is Linux or some other free operating systems where most of the competing products are under Open Source licenses.

For developers and other third parties the license should be more permissive to maximize the incentives to get third party support for the product.

Many companies who release their software under persistent licenses also sell the same software under commercial license to those who do not want to be bind by the terms of the restrictive licenses (Dual Licensing). There may also be substantial switching costs when a company or project changes its licensing policy.

### When does Open Source licensing make sense?

The results of the paper, suggests some questions when making a license choice:

- *Market focus.* For end-users the license terms may be more restrictive but for developers the terms must be rather permissive.
- *Software patenting.* GNU GPL and LPGL licenses are incompatible with software patents.
- *Competition and leadership.* The risk with a permissive license is that if the project does not have strong leadership it may be hijacked by a competitor.
- *Third party developers.* GNU GPL is incompatible with most types of commercial add-on products.
- *Rights clearing.* Distributed and Open Source development processes require rights and clearing with costs and benefits.

## B.20 Why you shouldn't use the Library GPL for your next library

*Richard Stallman*
[Sta99]

R. Stallman provides reasons for not using the LGPL. The main difference between the GPL and LGPL is that LGPL permits the use of the library in proprietary programs; whereas a library under the GPL license makes the library available only for free programs. Therefore, using the GPL for a library gives free software developers an advantage over proprietary developers, since these cannot use the library. Nevertheless, if the library's features are available for proprietary software through other alternative libraries, it is not useful to use the ordinary GPL license since it doesn't give free software any particular advantage.

## B.21 Open Source Licensing

*Henrik Ingo*
[Ing]

Henrik Ingo introduces some of the common Open Source licenses, explaining what the main consequences of the licenses are.

### Licenses

In the GPL license, redistribution is allowed and will always be bound by the same GPL requirements, the most important of which is that Open Source code always be made available. The GPL doesn't require you to give away source code to everybody on the Internet. It requires that you give it to the recipient (for example, the customer) of your software. The GPL does not mean that you have to give away anything for free. Actually, if your company is the only one who has written code for a certain GPL application, you own the full copyright for it. Therefore, you can distribute it under whatever license you wish (Dual Licensing). For large collective projects, where each contributor owns the copyright to their pieces of source code, it would be unreasonable to consider Dual Licensing. Therefore, these large projects will be GPL forever.

With the LGPL, the library itself must be always distributed with some source code, including any bug fixes or changes you have made. But the applications using the library need not be: they can even be closed source. This is very convenient for a library that will be used by many different applications.

The Berkeley Software Distribution License (BSD) essentially says: “You can do whatever you want and there is no warranty”. Other very similar licenses are the MIT license and the X license.

The difference between a BSD-style license and GPL license is that a BSD license doesn’t require you to distribute source code.

Sometimes, the BSD license might contain “the advertising clause”. If so, you must give appropriate credit to the copyright holders of the code you have used for your applications.

The Affero General Public License is like the GPL, but with the added requirement that publishing the software as a web service also requires you to provide an opportunity for your users to download the source code. Under a GPL license, if you publish code in your own webserver for your users to enjoy, you don’t have to distribute the source code, since you’re not distributing copies of your software to anyone.

Another option is to mix code from different licenses. Many of the licenses are just incompatible, so it is important to carefully consider which projects you will use for your software.

### Choosing a license

The two main reasons to choose the GPL license are that:

- The code developed in a GPL project will stay GPL forever and nobody will be able to use your source code in proprietary software.
- You could do “Dual Licensing”, providing (in exchange for money) an alternative license for those who not accept the GPL requirements.

The main reason to use the LGPL license is to allow the “normal use” of your Open Source Software by everyone. If you used the GPL, developers who do not accept the GPL requirements could not use your Open Source Software.

The BSD-style licenses are simple and short. They are not incompatible with other licenses. However, you cannot feel bad if your software under BSD-style licenses is incorporated into proprietary software.

Apache-style licenses are similar to BSD licenses. They are used by people who take licensing more seriously and who consider the BSD-style licenses not enough.

Apache-style licenses are used by organizations such as the Apache Foundation or IBM.

## B.22 Open Source Software Development: An Overview

*Min-Wei Wu and Ying-Dar Lin*
[WL01]

This article briefly explains different aspects of the Open Source Software Development. In theses aspects, we can find the Licensing models, which fall into three general categories: free - the program can be freely modified and redistributed; copyleft - the owner gives up intellectual property and private licensing; and GPL-compatible - licenses are legally linked to the GPL licensing structure.

In addition to this, we can find some common Open Source license models:

- **GPL** - It ensures distribution of any derivative work under the same license model.
- **LGPL** - It lets users extend the source with proprietary modules.
- **BSD** - It offers free code distributions and allows covering derivative works under different terms as long as the necessary credit is given.
- **MPL** - It requires distributing derivative works under MPL, which means that derivative work loses patent rights but still can enjoy private licensing. However, a module that MPL covers cannot legally be linked together with a module that GPL covers.
- **Netscape Public License** - It is a MPL extension that permits Netscape to use your added code even in its proprietary versions of the program.
- **Qt Public License** - A noncopyleft free software license, QPL requires distributing any modified source distributions only as patches.
- **Artistic License** - Nearly identical to the GPL, AL doesn't require distributing derivative works under the same terms when a company uses them internally.

| **Licensing model** | **Free Software** | **Open Source** | **Copyleft** | **GPL-compatible** | **Examples** |
|---|---|---|---|---|---|
| GPL | ✓ | ✓ | ✓ | ✓ | CVS |
| LGPL | ✓ | ✓ | Partial | ✓ | GNU C library |
| X11 | ✓ | ✓ | × | ✓ | XFree86 |
| Python | ✓ | ✓ | × | ✓ | Python |
| BSD | ✓ | ✓ | × | × | Apache, Sendmail |
| MPL/NPL | ✓ | ✓ | × | × | Mozilla |
| QPL | ✓ | ✓ | × | × | Qt |
| Sun Industry Standard Source License (SISSL) | ✓ | ✓ | × | × | Commercial-version Star-Office |
| Artistic License(AL) | × | ✓ | × | × | Perl |
| Apple Public Source License (APSL) | × | ✓ | × | × | Darwin |

**Table B.6:** Open Source licensing models