JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 22, 1-17 (2006)

Open Source Licenses and the Creative Commons Framework: License Selection and Comparison

YI-HSUAN LIN, TUNG-MEI KO, TYNG-RUEY CHUANG AND KWEI-JAY LIN*
Institute of Information Science
Academia Sinica
Nangang, Taipei 115, Taiwan
**Department of Electrical Engineering and Computer Science*
University of California, Irvine
Irvine, CA 92697-2625, U.S.A.

Numerous licenses confuse new participants in the free/open source software (FOSS) world. In this paper, we provide a general introduction to eleven commonly-used FOSS licenses. By applying seven specific considerations that are proposed in our study, developers can identify which license suits their needs best. We further use the above considerations to rank these FOSS licenses, which are Open Source Initiative (OSI)-approved, in terms of their degree of *openness*. The recently established Creative Commons (CC), based on FOSS concepts, provides yet another model for licensing creative works. We attempt to use the CC licensing model to analyze FOSS licenses, so that new participants can better understand these licenses. By examining these FOSS licenses with CC's four differentiating elements (attribution; noncommercial; no derivative works; share alike), we construct a table with which new participants could understand FOSS and the above categories with the knowledge of the CC licensing model. We also rank these licenses based on CC's differentiating elements and offer a different approach to examine the openness of FOSS licenses.

Keywords: free software, open source, FOSS, program, license, copyleft, Creative Commons, copyright

1. INTRODUCTION

With the increasing number of free/open source software (FOSS) users, FOSS licenses are receiving more and more attention. These licenses have also inspired the creation of the Creative Commons (CC) licensing model for creative works (non-software works). However, most programmers and users are still unfamiliar with the FOSS and CC licensing models. To remedy this situation, books, web sites, and journal articles have been published to discuss the models and the differences among these licenses. Most of these works have tried to go deep, examining every detail of each license. Unfortunately, they have caused even more confusion, since most programmers do not have the training nor the time to digest these legal opinions. What they need is a simple way to guide them on what and why they need to consider.

It is the goal of this paper to provide simple guidelines to help programmers choose an appropriate license when they are ready to release their open source software. We aim

Received June 6, 2005; revised September 30, 2005; accepted October 10, 2005.
Communicated by Kwei-Jay Lin.

to provide programmers with a basic understanding of FOSS licenses and the CC licensing model, so that they can systematically decide which license they should adopt. In this paper, we first review FOSS-related concepts and introduce 11 commonly-used FOSS licenses' characteristics by dividing these licenses into three basic categories: GPL type, BSD type and Others. We then introduce the basic issues that programmers should consider when selecting a suitable license from among the numerous FOSS licenses. We also use these considerations to rank the licenses' degree of *openness*, referring to how open software may remain following distribution and redistribution. Then, we give a brief introduction to the development of CC and examine the above FOSS licenses based on their relationship with CC's four differentiating elements. We further use these elements to rank these FOSS licenses.

2. FREE/OPEN SOURCE SOFTWARE

2.1 The Concept of FOSS

FOSS is an abbreviation of "free/open source software," which is derived from two different terms: *free software* and *open source software*. The term *free software* was first proposed and adopted by Richard M. Stallman, the father of the *free software* movement. To accomplish his cherished goal of free software sharing, Stallman decided to devise an open operating system, the source code of which can be accessed, used, and modified freely by any one. Stallman called the result *free software* and named it *GNU Project* [1]!

Along with the *GNU Project*, Stallman also established the *Free Software Foundation* (*FSF*) [2] to further promote the concept of *free software* and announced four types of freedom [3]:

- the freedom to run the program, for any purpose;
- the freedom to study how the program works and adapt it to your needs;
- the freedom to redistribute copies so you can help your neighbor;
- the freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

Access to the source code is the basis for the above four types of freedom.

As mentioned above, *free software* does not refer to software that is distributed at no charge, but the term was found to be misleading and hindered the commercialization of *free software*. Therefore, an alternative term was sought. Finally the term *open source software* was chosen, its characteristics defined, and the *Open Source Initiative* (*OSI*) [4] formed for the management and timely updating of the *Open Source Definition* (*OSD*) [5]. *OSI* has even registered a certification mark [4]: the *OSI Certified* mark. This mark can be put on software prior to distribution, so that people can easily recognize that this license conforms to *OSD* and is *open source software*.

As stated above, the major difference between FOSS and proprietary software lies in their licensing models. While users of proprietary software are normally required to pay royalties for using, distributing, copying, or editing the software, in the case of FOSS, these rights are royalty-free. In general, FOSS has 3 legal characteristics:

1. no royalties,
2. no geographical restrictions on distribution, and
3. no specific licensees.

Since no royalties are required, FOSS is provided on an "AS IS" basis and without any warranty. However, the licensee can offer a warranty to the recipient[1] voluntarily.

2.2 Free/Open Source Software Licenses

Based on the characteristics of FOSS licenses listed above, there are numerous FOSS licenses, and their contents vary. Currently, there are some 60 licenses, which are categorized as *free software* licenses by *FSF* [6]. The licenses recognized by *OSI* also number close to 60. All these licenses can be divided into three types: GPL type, BSD type, and Others.

2.2.1 GPL-Type Licenses

To achieve the desired four kinds of freedom, Stallman created a licensing model, called *copyleft*. As with most *free software*, users can use, copy, distribute, or edit copylefted programs[2] in the form of source code, without paying any royalties. The major difference is that the program modified from copylefted programs also must be distributed under the same license of the original program. Program developers themselves have no right to decide under which licenses modified programs can be distributed.

Stallman put the idea of *copyleft* into action by establishing the GNU General Public License (GPL), which is now probably the most famous and widely-used *free software* license. The current version of GPL is 2.0 [7].

Since modified GPL-licensed programs also must be licensed under GPL, GPL is a so-called *viral license*. For example, a software programmer who adopts a portion of the GPL-licensed source code must open all of the source code, including the modification and the original GPL-licensed source code.

A license can be classified under GPL type if it requires derivatives to adopt the same license of the original program and to open its source code. This characteristic increases the openness of source code and the circulation of programs.

The GNU Lesser General Public License v.2.1 (LGPL) [7], which is especially suitable for libraries, is also a GPL-type license.

2.2.2 BSD-Type Licenses

Under the *copyleft* mechanism, GPL-type licenses prohibit program developers from choosing a license according to their wishes. At the other extreme are BSD-type licenses [7], which do not impose any restrictions on users in this respect. With BSD-

[1] Throughout this paper, if the licensee redistributes the program to others, the person who receives the redistributed program from the licensee is called the recipient.

[2] In subsections 2.2 and 2.3, we use the term "program," not "software," because generally a FOSS license is applied to a program, but not the whole software.

type licenses, developers/licensees can decide at their own discretion whether to open the modified source code and collect royalties when redistributing it. Therefore for developers/licensors who do not tend to keep their source code always open, but only hope that their programs can be redistributed as widely as possible, BSD-type licenses may be good choices.

The MIT license (MIT), Apache Software License (Apache v1.1), Apache license v.2.0 (Apache v.2.0), and zlib/libpgn license (zlib) are also BSD-type [7] licenses.

2.2.3 Others

GPL and BSD are at the two extremes of FOSS licenses, and only a few licenses can meet their conditions. Most FOSS licenses fall between these two extremes. Two examples are the Netscape Public License (NPL) [8] and Mozilla Public License v.1.1 (MPL) [7].

In 1998, Netscape announced its support for the open source community. In addition to transforming Netscape Navigator into *open source software*, Netscape also began to develop a new browser, called Mozilla. Netscape designed NPL and MPL separately for Netscape Navigator and Mozilla. These two licenses are very similar, however; the main contents are the same, but MPL has some amendments that NPL does not have [9]. Because the NPL's amendments are not consistent with *OSD*, NPL is not considered an *open source software* license.

The most significant feature of MPL and NPL is that they reconcile the different characteristics of FOSS and proprietary software. Netscape Navigator was originally proprietary software. Some of the source code of Netscape Navigator was licensed from third parties, so it could not be open. The third-party code was not under the control of Netscape. Besides, Netscape's business strategy did not allow some specific code to be open. To solve this problem, MPL/NPL allow some source code to be open, but a certain portion of the source code to be closed. That is, MPL/NPL allow different licenses to be applied to different portions of the source code in one program, as long as the terms and conditions are compatible [10].

Special stipulations in MPL/NPL address possible future legal disputes: i.e., programmers who modify the source code of a program should specify the modification in a file, and those who has knowledge of what patent licenses maybe implemented should also specify the relevant detail information in a file.

Because of the above characteristics, MPL/NPL and other similar licenses are especially suitable when closed source code and open source code co-exist in a program.

Besides MPL, the Artistic license (Artistic), Common Public License v. 1.0 (CPL), and Qt Public License v. 1.0 (QPL) are also categorized to Others [7].

2.3 How to Select a License

To help programmers select an appropriate license, we propose a decision-making process that includes the following seven licensing considerations.

2.3.1 Whether the source code must be provided when the original program is redistributed

Some FOSS licenses require that licensees make sure that recipients can have access to the source code when the licensed program is redistributed, either by distributing the source code directly or by distributing the executable code but allowing recipients to download the source code from a specified location. This requirement can enable the source code to remain open and prevent licensees from keeping it closed.

GPL-type licenses demand that licensees provide the source code when they distribute a licensed program, but BSD-type licenses have no such requirement; they allow licensees to make their own decisions. Licenses in the "Others" category have different requirements but most of them require users to provide the source code.

If a program is developed for the purpose of research and the programmers wish to keep the source code open, they can choose a GPL-type license. If they do not want to impose any specific requirement and wish the program to be widely distributed, then a BSD-type license is a good choice.

2.3.2 Whether collection of royalties is allowed when a program is distributed

In general, one of the major characteristic of FOSS is that it is royalty-free. This avoids the situation where recipients are prevented from accessing source code because of high royalties. However, not all FOSS licenses ensure royalty-free redistribution of programs. Under BSD-type licenses, licensees can receive and use a program on a royalty-free basis; on the other hand, they are also allowed to collect royalties when they redistribute the program. On the contrary, GPL-type licenses stick to royalty free. As for licenses in the "Others" category, their requirements with regard to the collection of royalties during redistribution vary.

2.3.3 Whether a fee higher than the distribution cost can be collected when a program is distributed

Except royalties, FOSS licenses allow licensees to charge other fees. These fees can be charged to cover the cost of service, warranties, distribution, etc. However, GPL, LGPL, QPL, and CPL do not allow licensees to charge a fee that is higher than the distribution cost if the source code is distributed separately from the binary or executed form.

2.3.4 Whether a program can be sublicensed

Sublicensing means that licensees may sublicense a program to others as the status of this program's copyright holders. Continuous sublicensing results in a licensing chain. Licensees must make sure that the licenses they obtain are valid before they sublicense programs to recipients; otherwise the recipients may receive invalid licenses, which will break the licensing chain. Only a few licenses, such as MIT, MPL, and CPL, allow sublicensing, and their requirements vary[3].

[3] Any breach of MPL will cause termination of a particular licensee's rights, but will not affect sublicensing. Both MIT and CPL grant licensees the right to sublicense, but state nothing further.

2.3.5 Whether a modification[4] should be distributed under the same license as the original program

Once licensees receive the source code of a program, they can modify the source code and produce a *modification*; this is one of the features of FOSS licenses. If such a modification is recognized as a derivative of the original program under copyright laws, then copyright holders of the modification have the right to choose a license at their own discretion and need not adopt the same license as the original program. However, some FOSS licenses, such as GPL-type licenses, stipulate that a derivative work should use the same license as the original program, which places a constraint on the copyright holders of derivative works. On the other hand, BSD-type licenses impose no such restriction, so the copyright holders of derivative works may choose licenses at their own discretion. Licenses in the "Others" category have different requirements regarding license selection for a modification.

The purposes of requiring the author of a *modification* to use the same license as the original program are to ensure that the original program and its *modification* can be distributed royalty-free, and to prevent the source code of the *modification* from being closed. Therefore, programmers may choose GPL-type licenses if they wish the source code of their modifications to remain royalty free and open.

2.3.6 Whether the source code must be provided when a modification is distributed

The purpose of this source code requirement is to ensure that the source code of a *modification* remains open. Therefore, if programmers do not want the source code they have developed to be closed by licensees, they should choose a license containing this requirement. Some licenses, such as GPL-type licenses, require licensees to provide the source code when distributing a *modification*, but BSD-type licenses have no such requirement. As for other licenses in the "Others" category, their requirements vary: MPL, QPL, and CPL require that the source code of a *modification* be provided when it is distributed while Artistic does not.

2.3.7 Whether documentation must be provided with a modification

The documentation referred here is the record which is made when a program is modified. Such records can be helpful for subsequent modifications, because recipients can know from these records when a *modification* was made and which parts of the program were modified. In addition, when modifications are fed back to the program's original developers, this documentation will help them understand how and when the program was modified and what kinds of improvements were made to the original program.

GPL-type licenses require that such documentation be attached to a modification. MPL, Artistic, Apache v.2.0, and zlib also have this requirement, while BSD, MIT, Apache v.1.1, CPL, and QPL do not.

[4] The coverage of modifications in this paper can be extended to programs developed directly by modifying source code or programs that include part of the original program's source code. Even if a program contains only a small part of the source code of the original program, it will be defined as a modification.

2.3.8 Summary

The above seven considerations can be further categorized based on the purpose of program development. We will now use a common practice in academic research as an example: For the purpose of academic research, programmers may prefer that the source code of a program and its modifications remain open and royalty-free, so as to let more people study and use the program and its modifications. To ensure continuous program development, discussion and research, it is often necessary to impose the same license of the original program on subsequent modifications. In addition, to avoid the problem of broken licensing chains, programmers may choose a license which explicitly prohibits sublicensing.

Based on the above discussion, we will now present a structured decision-making process that programmers can follow to choose a suitable license from among the 11 commonly-used FOSS licenses mentioned in this paper.

Step 1: If programmers want the source code of their programs to be provided during distribution, then they should consider GPL, LGPL, MPL, QPL, CPL, or Artistic.
Step 2: If programmers insist that modifications of their programs be under the same licenses as the original programs, then only GPL, LGPL and MPL are suitable.
Step 3: If programmers insist that no royalties should be collected when their programs are redistributed, then GPL, LGPL and MPL are possible choices.
Step 4: If programmers do not want to allow licensees to sublicense their original programs, then only GPL and LGPL are suitable.
Step 5: If the developed program is a library, then programmers can choose LGPL; otherwise, GPL may be a better choice.

If programmers do not care whether the source code of their programs remains continually open or whether their programs are used for commercial purposes, then BSD-type licenses may be good choices. If programmers have other considerations in mind, then they can choose a license in the “Others” category that meets their specific requirements.

2.4 Comparison of *Openness* Based on Specific Considerations

Although Stallman has proposed four kinds of freedom [3], using freedom as a criteria for examining FOSS licenses’ degree of *openness* is not sufficient. Therefore, we use the seven considerations mentioned in section 2.3 as criteria for examining FOSS licenses’ degree of *openness*. We define the *openness* of FOSS licenses as follows: 1) the source code is provided during distribution; 2) no royalties are paid; 3) the first two requirements are retained for all future derivatives. We will now compare the 11 FOSS licenses based on their degree of *openness*. Through the comparison, we hope to help programmers understand FOSS licenses better so that they can more easily decide which licenses they should use. In our comparison, we further divide the considerations discussed in subsection 2.3.3 into 2 cases, i.e. considerations (5) and (8). The eight considerations as listed below:

(1) Whether collection of royalties is allowed when a program is distributed?
(2) Whether the source code must be provided when the original program is redistributed?
(3) Whether the source code must be provided when a modification is distributed?
(4) Whether a modification should be distributed under the same license as the original program?
(5) Whether a fee higher than the distribution cast can be collected when a program is distributed without the source code?
(6) Whether a program can be sublicensed?
(7) Whether documentation must be provided with a modification?
(8) Whether a fee higher than the distribution cost can be collected when a program is distributed with the source code?

For considerations (2), (3), (4), and (7), a "Yes" answer represents a higher degree of *openness* (+), while a "No" answer represents a lower degree of *openness* (-); for considerations (1), (5), (6), and (8), a "Yes" answer represents a lower degree of *openness* (-), while a "No" answer represents a higher degree of *openness* (+). The above considerations do not carry the same weight. Consideration (1) weighs more heavily than consideration (2), consideration (2) weighs more heavily than consideration (3), and so on. Therefore, a license has more pluses (+) may not rank higher in terms of openness if all of these pluses (+) are gained from the later considerations. Two considerations deserve extra discussion. Consideration (4) means that for a modification, the source code must be open and royalty-free when it is distributed; therefore, a "Yes" answer represents a higher degree of *openness*. Of course, there may be a license which allows the source code of a modification to be closed and/or allows royalties to be collected but requires that the modification to use the same license as that of the original program. However, since none of the 11 FOSS licenses match these conditions, we can exclude this situation from our discussion in this paper. Since sublicensing may cause a licensing chain to be broken, a "No" answer represents a higher degree of *openness* for consideration (6).

Based on the above considerations, the comparison on the degree of *openness* of the 11 FOSS licenses is given as Table 1.

3. CREATIVE COMMONS

We will now turn our attention to another licensing model, Creative Commons (CC), which is closely related to copyright laws. The current copyright laws stipulate that any works eligible for copyright are automatically afforded full copyright protection upon completion. However, many people may not want to exercise all of the rights granted by the copyright laws, especially if they are engaged in creative activities on the Internet, as full copyright protection may prevent extensive diffusion and exposure of their works. To address this dilemma, a group of researchers from the fields of cyber-law, intellectual property rights, and information technology dedicated themselves to the formation of CC, and founded this non-profit organization in 2001. Motivated in part by the *FSF*'s GPL, CC has established a licensing model under the current copyright laws that allows authors to license the free use of their works to other people under certain conditions [11].

Table 1. Ranking of FOSS licenses' degree of *openness* based on eight specific considerations.

(+): higher degree of *openness*; (-) lower degree of *openness*

License type	Licenses	(1) Collection of loyalties is allowed when a program is distributed	(2) The source code is provided when the original program is redistributed	(3) The source code must be provided when a modification is distributed	(4) A modification should be distributed under the same license as the original program	(5) A fee higher than the distribution cost can be collected when a program is distributed without the source code	(6) The program can be sublicensed	(7) A documentation must be provided with a documentation	(8) A fee higher than the distribution cost can be collected when a program is distributed with the source code	Degree of *openness*
GPL type	GPL	No(+)	Yes(+)	Yes(+)	Yes(+)	No(+)	No(+)	Yes(+)	Yes(-)	1
	LGPL	No(+)	Yes(+)	Yes(+)	Yes(+)	No(+)	No(+)	Yes(+)	Yes(-)	1
Others	MPL	No(+)	Yes(+)	Yes(+)	Yes(+)	No(+) (source code is always redistributed)	Yes(-)	Yes(+)	Yes(-)	2
	QPL	No(+)	Yes(+)	Yes(+)	No(-)	No(+)	No(+)	No(-)	Yes(-)	3
	CPL	No(+)	Yes(+)	Yes(+)	No(-)	No(+)	Yes(-)	No(-)	Yes(-)	4
	Artistic	No(+)	Yes(+)	No(-)	No(-)	No(+) (source code is always Redistributed)	No(+)	Yes(+)	No(+)	5
BSD type	Apache v.2.0	Yes(-)	No(-)	No(-)	No(-)	Yes(-)	No(+)	Yes(+)	Yes(-)	6
	Zlib	Yes(-)	No(-)	No(-)	No(-)	Yes(-)	No(+)	Yes(+)	Yes(-)	6
	Apache v.1.1	Yes(-)	No(-)	No(-)	No(-)	Yes(-)	No(+)	No(-)	Yes(-)	7
	BSD	Yes(-)	No(-)	No(-)	No(-)	Yes(-)	No(+)	No(-)	Yes(-)	7
	MIT	Yes(-)	No(-)	No(-)	No(-)	Yes(-)	Yes(-)	No(-)	Yes(-)	8

3.1 CC Licenses

To help authors select the specific terms and conditions under which they can let users worldwide use their works, CC released a set of copyright licenses for creative works [11] in December 2002. Based on four elements [12] (three choices: No Derivative Works (No Derives), Noncommercial, Share Alike; and one implied: Attribution), CC developed six different licenses [13]. Authors can select a license they find most appropriate for their works from among the six CC licenses. When authors decide which CC license to use when they release their works, they can reserve some rights instead of all rights for their works. These six licenses are listed below:

(1) Attribution-Noncommercial-No Derives,
(2) Attribution-Noncommercial-Share Alike,
(3) Attribution-Noncommercial,
(4) Attribution-No Derives,
(5) Attribution-Share Alike,
(6) Attribution.

"Attribution" has been the default element of all CC licenses since v. 2.0 was released. It means that the licensee can copy, distribute, display, and perform the licensor's works, and derivative works based on it, provided that the licensee credits the licensor.

"Noncommercial" means that the licensee can copy, distribute, display, and perform the licensor's works only for noncommercial purposes.

"No Derives" means that the licensee can copy, distribute, display, and perform only verbatim copies of the licensor's works.

"Share Alike" means that the licensee can distribute derivative works only under a license identical to the license that the licensor chose for his/her works.

Unlike GPL, which is designed for software, CC licenses are designed mainly for other kinds of creative works, such as web sites, scholarship, writing, music, film, photography, literature, courseware, etc. However, the four elements of CC licenses address similar concerns for most programmers. Should a redistributed software/program contain a copyright notice specifying the relevant details of the copyright holder? Can the licensee use, copy, or distribute derivative works for commercial purposes? Is the licensee permitted to modify the program? Should modified works be regulated under the same license as that being used for the original program? In addition, CC has designed an online license wizard [14] to help authors obtain the most suitable license by answering three simple questions: 1) Will you allow commercial uses of your works? 2) Will you allow modifications to your works? 3) If you answer "yes" to 2), do you require other users to share alike?

3.2 The Perspective of CC Licensing Elements on Specific FOSS Licenses

To a certain extent, the creation of CC licenses was inspired by GPL of *FSF*. CC licenses mostly apply to creative works other than software, while licenses provided by *FSF* or *OSI* are designed specifically for programs. Based on the four licensing elements

"Attribution," "Noncommercial," "No Derives," and "Share Alike," 11 CC licenses v.1.0[5] were developed. As most early CC licenses v.1.0 adopters chose the licenses containing the element "Attribution," "Attribution" became a default element in CC licenses v.2.0, released in May 2004. We can see that all of the 11 FOSS licenses discussed in this paper have relevant terms about copyright notices, but there are variations. They are discussed in detail below.

Attribution The terms and conditions of GPL only stipulate that a licensee may copy and distribute verbatim copies of a GPL-licensed program's source code, on the condition that an appropriate copyright notice is placed on each copy[6]. LGPL includes similar terms[7]. Neither license gives a detailed definition of "appropriate copyright notice." However, in "How to Apply These Terms to Your New Programs," we find: "… each file *should* have at least the "copyright" line …" along with examples showing that the copyright line should contain the program's name, copyright©<year><name of author>, etc[8]. Similar terms can also be found in LGPL[9]. As such, we can infer that GPL/LGPL require that a licensee must credit the original author of the program/library.

According to [7], BSD and MIT licenses allow a licensee to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the software, provided that the copyright notice (copyright©<year><copyright holders>) is included in all copies or substantial portions of the software. The zlib license allows licensees to use, alter, and redistribute the software freely, provided that they do not claim that they wrote the original software and that the origin of the software is not misrepresented.

The QPL, CPL, and Artistic licenses all have provisions related to copyright notices but do not specify the contents of such notices[10]. Apache v.1.1 requires that redistributions of the source code or redistributions in binary form retain/reproduce "Copyright © 2000 The Apache Software Foundation. All rights reserved."[11] Apache v.2.0 clearly allows a licensee to copy and distribute an Apache-licensed work or its derivative works. However, the licensee must retain all copyright, patent, trademark, and attribution

[5] CC Licenses cannot contain the elements "No Derivs" and "Share Alike" simultaneously, because "No Derivs" prohibits users/licensees from making any modifications to the original works, while "Share Alike" requires that the derivative works be distributed under the same CC licenses as the original works.

[6] See article 1 of GPL.

[7] See article 1 of LGPL.

[8] See "How to Apply These Terms to Your New Programs."

[9] See "How to Apply These Terms to Your New Libraries."

[10] QPL requires that when a licensee copies and distributes an unmodified software package, the entire package released by the initial developer must be distributed, including but not limited to copyright notices. Also, the licensee may modify the QPL-licensed software and distribute such modifications, provided that the copyright notices in the software are not altered. The modification must also be kept separate from the software. QPL does not specify whether copyright notices must include the name of the initial developer of the software. CPL, like QPL, requires that when modified code or unmodified code is distributed, copyright notices cannot be altered or removed. Artistic stipulates that a licensee may copy and distribute verbatim copies of the source form of the unmodified files distributed by the copyright holders, or derivatives of those files modified according to the wishes of the copyright holder, provided that the licensee reproduces the copyright notice. Artistic does not specify what content the "original copyright notice" should include. For further information, see http://www.opensource.org/licenses/qtp1.php, http://www.opensource.org/licenses/cp11.0.php, and http://www.opensource.org/licenses/artistic-license.php, (last visited April 29, 2005).

[11] See article 1,2 of Apache v.1.1.

notices from the source form of the Apache-licensed work in the source form of the derivative works that the licensee distributes[12]. In addition, from the "APPENDIX: How to apply the Apache License to your work of Apache v.2.0," we can see that a copyright notice should contain the "name of the copyright owner." Compared with Apachev.2.0, MPL states more clearly that each time a licensee modifies the Original Code[13], a prominent statement that includes the name of the Initial Developer must be shown.

In summary, eight licenses (GPL, LGPL, MPL, Apache v. 1.1, Apache v. 2.0, zlib, BSD, and MIT) require that licensees credit the authors of programs.

Noncommercial Another issue important to most programmers is whether a program or any modification derived from it can be used for commercial purposes.

GPL does not prohibit commercial use, but a licensee must ensure that the derivative works follow terms and conditions of GPL. That is, for instance, a licensee must provide the source code of derivative works to recipients and allow recipients to modify and redistribute the derivative works [15]. LGPL has the same regulation [7]. The BSD, MIT, Apache v.1.1, Apache v.2.0, zlib, and QPL licenses do not have provisions explicitly restricting commercial use; hence, our interpretation is that these licenses do not prohibit commercial use. Although MPL has a special provision defining "Commercial Use[14]" with relevant regulations[15], the "Commercial Use" provision does not refer to the commercial application of derivative works. Therefore, we infer that MPL does not prohibit commercial use, either. Artistic licenses stipulate that a licensee may charge a reasonable copying fee or any fee for supporting the Package[16]. CPL explicitly states that commercial use of CPL-licensed programs is allowed[17].

From the above discussion, we can see that neither GPL nor LGPL licenses prohibit commercial use. The BSD, MIT, Apache v.1.1, Apache v.2.0, zlib, and QPL licenses do not explicitly prohibit commercial use; on the contrary, CPL clearly states that commercial purpose use is allowed. "Commercial Use" defined under MPL is not related to the commercial use of derivative works. Artistic allows authors to charge a reasonable distribution fee but prohibits charging a fee for modified/unmodified files themselves.

No Derives We will now address the issue of whether the 11 FOSS licenses have any regulation concerning modifications of original works by licensees. *OSI*-approved licenses should follow the 10 *OSD* criteria [5] set by *OSI*. In the terms for Derived Works, the third *OSD* criterion states that, "*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*." As such, the effect of the "No Derives" on all *OSI*- approved licenses is to make "to allow modifications of the works" a default element. Hence, "No" is shown in all fields of the "No Derives" column. Nevertheless, it is noteworthy that QPL

[12] See article 4 of Apache v.2.0.
[13] "Original Code" is defined as article 1.10. of MPL.
[14] See article 1.0.1. of MPL.
[15] See article 2.2.(c) of MPL.
[16] "Package" is defined within "Definitions" of Artistic.
[17] See article 4 of CPL.

does not allow users to directly modify original works. Instead, it requires that all modifications should exist separately from the original works (e.g., in the form of a patch)[18].

All of the 11 FOSS licenses allow modification of original works.

Share Alike Finally, we will examine the effect of the element "Share Alike." One of the characteristics of the distinctive *copyleft* mechanism in GPL and LGPL is that derivative works must be distributed under exactly the same license as the original works[19]. Therefore, derivative works that adopt a portion of or all of GPL-licensed software will become GPL-licensed software. Furthermore, MPL stipulates that modifications derived from MPL-licensed programs must adopt MPL. This restriction is identical to that of GPL and LGPL. Only GPL, LGPL and MPL have regulations similar to "Share Alike"; the other eight licenses do not specifically stipulate that derivative works must only be distributed under the same license as the original works.

3.3 Using CC Elements to Compare the Openness of Specific FOSS Licenses

In this section, we will attempt to use CC's four elements to examine the degree of *openness* of FOSS licenses. By answering "Yes," "No," and "Contingent" for each CC element, we can assign a specific score and ranking for each license. Thus, programmers can learn how to select a license more easily.

3.3.1 Ranking of eleven FOSS licenses

In the following discussion, we will adopt the same criteria listed in Table 1: if a license is beneficial for the *openness* and distribution of software, it will obtain a higher ranking; otherwise, it will receive a lower ranking.

For the elements "No Derives," "Noncommercial," and "Attribution," a "No" answer represents a higher degree of *openness*, while a "Yes" answer represents a lower degree of *openness*. The element "Share Alike" applies to the opposite situation. Furthermore, each license can gain 1 point if it gets a "No" answer for "No Derives," "Noncommercial," or "Attribution"; but if a license gets a "Yes" answer for "Share Alike" it will gain 1 point as well; the answer "Contingent" means that the license does not have an expressly positive or negative attitude toward the CC elements, and thus, gains 0.5 point.

According to the above criterion, GPL, LGPL and MPL get 3 points and rank highest for openness. QPL, CPL and Artistic get 2.5 points and are ranked in the middle. Each of the remaining five licenses gets 2 points and rank lowest.

3.3.2 How to select a license based on the four CC elements

From Table 2, we can draw the following conclusions: If programmers require that their programs be distributed under the same license as the original and that copies of the

[18] See article 2,3 of QPL.
[19] See article 2(b),(c) of GPL.

Table 2. Ranking of FOSS licenses' degree of *Openness* based on CC elements.

	Share Alike	No Derives	Noncommercial	Attribution	Ranking of *Openness*
GPL	Yes	No	No	Yes	1
LGPL	Yes	No	No	Yes	1
MPL	Yes	No	No	Yes	1
QPL	No	No	No	Contingent[20]	2
CPL	No	No	No	Contingent	2
Artistic	No	No	No	Contingent[21]	2
Apache v.2.0	No	No	No	Yes	3
zlib	No	No	No	Yes	3
Apache v.1.1	No	No	No	Yes	3
BSD	No	No	No	Yes	3
MIT	No	No	No	Yes	3

programs include their names but are not concerned about whether their programs are modified or used for commercial purposes, then GPL, LGPL, or MPL licenses will be good choices.

If programmers are willing to allow their programs to be distributed under a different license from the original and do not care whether their programs are modified or used for commercial purposes, then QPL, CPL, or Artistic licenses may be good choices. If programmers are willing to allow their programs to be distributed under different licenses and do not care whether their programs are modified or used for commercial purposes BUT do require that copies of their programs give them credit, then they may consider the Apache v.2.0, zlib, Apache v.1.1, BSD, or MIT license.

Comparing Table 2 with Table 1, we can observe the following. Table 2 has less number of rankings than Table 1. The reason is that in Table 2, we adopt the four CC elements to examine the degree of openness of 11 FOSS licenses, while we use 8 considerations as criteria to view how open these licenses are. Although CC's elements are easily understandable, these elements are not precise enough to provide new participants with a thorough understanding of FOSS licenses. Based on the four CC elements, these licenses fall into 3 ranks (1 to 3).

MPL deserves special discussion. In Table 2, MPL has a rank of 1, but it falls into the category of "other" licenses, which are ranked in the middle in Table 1. The reason is that MPL requires that modifications derived from MPL-licensed programs be governed by MPL. This characteristic makes MPL a GPL-type license and can be displayed via the element "Share Alike". In addition, MPL allows sublicensing, but GPL and LGPL do not. This feature makes MPL less open than GPL and LGPL. Therefore, MPL is given a rank of 1 in Table 2, but it belongs to the "others" category in Table 1.

[20] More information, see footnote 9.
[21] More information, see footnote 9.

4. CONCLUSIONS

The licensing model applied to FOSS is different from proprietary software models. This licensing model has opened up new and fresh options for programmers and users under the increasingly rigid copyright regime. However, the huge number of FOSS licenses often confuse new participants. In comparison, the CC licensing model is based on four simple, clear elements, and is easier for new participants to understand. We, thus, have used eight specific considerations and the four CC elements to analyze and compare 11 commonly-used FOSS licenses.

In the past, most programmers have chosen GPL-type licenses for their programs. The reason is that GPL and LGPL licenses were among the earliest ones and are still the most widely-known FOSS licenses. Yet, this does not mean that GPL-type licenses are suitable for all FOSS. Programmers should also consider BSD-type licenses, since they impose almost no restrictions on users and also allow programs to be distributed freely. Therefore, if a program is developed for no specific purpose, then BSD-type licenses may be better choices than GPL-type ones.

We hope this paper can help programmers, especially new participants, make decisions about licenses and, in addition, help foster a spirit of sharing and collaboration on the Internet.

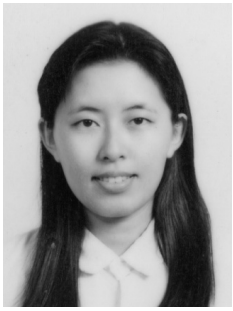
ACKNOWLEDGMENT

We especially thank Mr. Han-Teng Liao for his contributions to this paper.

REFERENCES

1. GNU Project: http://www.gnu.org/.
2. FSF: http://www.fsf.org/.
3. See http://www.fsf.org/licensing/essays/free-sw.html, last visited April 26, 2005.
4. OSI: http://www.opensource.org/.
5. The open source definition, available on http://www.opensource.org/docs/definition.php.
6. See http://www.gnu.org/licenses/license-list.html.
7. See http://www.opensource.org/licenses, last visited May 12, 2005.
8. Available on http://www.mozilla.org/MPL/NPL-1.1.html, last visited May 12, 2005.
9. Available on http://www.oreilly.com/catalog/opensources/book/netrev.html.
10. Available on http://www.oreilly.com/catalog/opensources/book/netrev.html.
11. Available on http://creativecommons.org/about/history, last visited May 3, 2005.
12. Available on http://creativecommons.org/about/licenses/, last visited May 3, 2005.
13. The explanation of 6 different CC licenses is available on http://creativecommons.org/license/meet-the-licenses, last visited May 3, 2005.
14. CC license wizard is available on http://creativecommons.org/license/?lang=en, last visited May 3, 2005.
15. See http://www.gnu.org/licenses/gpl-faq.html#GPLCommercially, last visited May 2, 2005.

Yi-Hsuan Lin (林懿萱) is the legal lead of Creative Commons Taiwan. She received her Master of Law degree (LL.M.) from Southern Methodist University, Dallas, Texas in 2001 and her B.L. degree from Fu Jen Catholic University, Taiwan. During 2002-2003, she worked in MediaTek Incorporation of Taiwan, which is the world's leading digital media solution provider. She works for Institute of Information Science, Academia Sinica, since 2004.



Tung-Mei Ko (葛冬梅) is a project manager at Open Source Software Foundry (OSSF), Institute of Information Science, Academia Sinica, since 2005. Her research area focuses on the legal topics about the Free/Open Source Software. She received her Master of Law degree (LL.M.) from the Institute of Tax Law, Münster University, Germany in 2002 and her B.L. degree from Chung Yuan Christian University, Taiwan. During 2003-2005, she worked in the Science and Technology Law Center (STLC) of Institute of Information Industry (III) in Taiwan.

Tyng-Ruey Chuang (莊庭瑞) is an Associate Research Fellow at the Institute of Information Science, Academia Sinica, Taipei, Taiwan. His research areas include functional programming, programming languages, XML and Web technologies, and social implications of information technologies. He received his PhD degree in Computer Science from the Courant Institute of Mathematical Sciences, New York University and his BS degree in Computer Science from National Taiwan University.

Kwei-Jay Lin (林桂傑) received the B.S. degree in Electrical Engineering from National Taiwan University in 1976, and the M.S. and Ph.D. degrees in Computer Science from the University of Maryland, College Park, in 1980 and 1985, respectively. He is currently a Professor in the Department of Electrical Engineering and Computer Science at the University of California, Irvine. He was an Associate Professor of Computer Science at the University of Illinois at Urbana-Champaign from 1985 to 1993. His research interests include service-oriented systems, e-com-

merce and enterprise computing, real-time systems, scheduling theory, distributed computing and operating systems. He has published more than 150 papers in academic journals and conference proceedings. He is the principal investigator of the RED-Linux project, an open source real-time Linux kernel project.

Dr. Lin is an Associate Editor of the IEEE Trans. on Parallel and Distributed Systems. He has been Co-Chair of the IEEE Technical Committee on E-Commerce since 2004. He was a guest editor for the IEEE Computer Special Issue on Web Services published in October 2003 and the IEEE Software Special Issue on Real-Time Systems Development published in September 1992. He was an Executive Committee Member of the IEEE Technical Committee on Real-Time Systems from 1998 to 2002. He has served on committees for many international conferences and workshops; for example, he was General Chair for the 1998 IEEE Real-Time Systems Symposium held in Madrid, Spain; General Chair for the 2003 IEEE Conference on E-Commerce held in Newport Beach, CA; Program Vice Chair for the 2004 IEEE Conference on Distributed Computing Systems held in Tokyo, Japan; General Co-Chair for the 2004 IEEE Conference on e-Technology, e-Commerce and e-Service (EEE-04) held in Taipei, Taiwan; Program Co-Chair for the 2005 IEEE Conference on E-Commerce held in Munich, Germany; and General Co-Chair for the 2006 IEEE Conference on E-Commerce. Dr. Lin received the 1990 NCR Award of Excellence from the University of Illinois and received the IBM Faculty Research Award twice in 1997 and 1998.