Western University
Scholarship@Western

Computer Science Publications | Computer Science

1-1-2007

Open Source Software Licensing Patterns

Halina Kaminski
University of Western Ontario, hkaminsk@csd.uwo.ca

Mark Perry
University of Western Ontario, mperry@uwo.ca

Follow this and additional works at: http://ir.lib.uwo.ca/csdpub

Part of the Computer Sciences Commons, and the Contracts Commons

Citation of this paper:
Kaminski, Halina and Perry, Mark, "Open Source Software Licensing Patterns" (2007). *Computer Science Publications.* Paper 10.
http://ir.lib.uwo.ca/csdpub/10

Open Source Software Licensing Patterns

Halina Kaminski, Mark Perry
University of Western Ontario
{hkaminsk, markp} @csd.uwo.ca

Introduction

Over the last two decades there have been thousands of software releases with ever increasing complexity. One division between software types is whether it is proprietary type software, such as Windows and DB2, or Free/Libre and Open Source Software (FLOSS), such as Linux and MySQL. Both types have associated licenses that define the terms and conditions of use, reuse and adaptation. The FLOSS term is convenient shorthand to encapsulate the various flavours of open source. In previous work, we have identified a number of patterns that can be used in developing a license for proprietary software. Here we show licensing patterns for FLOSS, and will provide a set of patterns that can be added to the existing software licensing pattern language [1].

To be categorized as FLOSS, the software license must grant certain rights to the user. These rights range from the basic access to the software's source code to the rights to make copies and distribution of the program. There has been much debate in the FLOSS community as to the extent of the rights, duties and privileges that are required to fall within various categories of FLOSS.

Background

Free/Libre and Open Source Software has raised much interest and been gaining widespread acceptance over the last two decades. This can be seen through the work of millions of programmers around the world who dedicate significant amounts of their time developing FLOSS. Although FLOSS has a lot benefits, there is a number of risks that are associated with this kind of software. One of the greatest risks is potential liability for intellectual property infringement. Almost every open source project is a collection of contributions from many people. Contributors do not guarantee the cleanliness of the code they contribute to the project. The standard open source license is designed to be protective of the contributor. Most often FLOSS license does not include any intellectual property representations or warranties in favor of the licensee. It usually contains a broad disclaimer of all warranties that benefits the licensor/contributors. It is very important to remember that there is a difference between the software that has a source code available to anyone and the software that is freely distributed, but still has its code closed to others. This paper concerns only the product that has a source code open to the public.

In the United States there are over 45 FLOSS licenses in common use. Sometimes the type of license can be crucial to its end use, and whether one type can co-exist with software of another type. The main division in FLOSS is between Copyleft and Non-Copyleft Licenses. The simpler forms of these licenses, for example the revised BSD and

MIT/X11 licenses, allow redistribution and use in both source and binary forms, with or without modification, on the condition that the copyright notice is retained, and that applicable warranties are disclaimed. The original BSD license had an 'obnoxious advertising clause' that required attribution to be displayed on all materials for the software, such as advertising. However, when there were many contributors to a project the attribution material quickly became large and unwieldy. Current versions of this license do not include the clause, but there are still many examples of software products released under the original license or modified versions of the original license.

There is no requirement that derivatives of the free software be free themselves. On the other hand, the copyleft licenses, like the GNU General Public License (GPL), attempt to create a contributory commons by requiring that any re-distribution of the software or its derivatives is released under the free license. The deciding factor in that segregation is the right to re-distribute the software. In a Copyleft type, any redistribution of a code or its derivatives must be released under the same free license.

Non-Copyleft licenses allow redistribution and use with or without modification, on the conditions that the copyright notice is retained and that any applicable warranties are disclaimed. In this type the derivatives of such software does not have to be free. In order to make a program free, a programmer should attach "License.txt" file to the source code containing the text of the license. Most programmers will not try to write an original license. Not only is it very difficult to come up with a well written document that covers all legal issues, but also the use of an established license has many advantages: people are familiar with it and know the implications; the license will have been tested; and common licenses make it easier to share code between FLOSS software projects.

The difference between open source and free software is at a philosophical level of abstraction. Because the definition of 'open source' is somewhat broader than the definition of 'free software', it is clear that all free software is open source, but not all open source software is free. In practice, however, most licenses that satisfy the OSI definition will also be considered 'free'. It should also be noted, as Stallman pointed out that "free software does not mean that the software is free, as in requiring no payment. When I speak of free software, I'm referring to freedom, not price. So think of free speech, not free beer." [2]

There are many kinds of FLOSS licenses, but most common are four major types: the GNU General Public License (GPL), the GNU Lesser (or Library) General Public License (LGPL), the MIT license, and the BSD license. The Open Source Initiative refers to these four types of licenses as the classic open source licenses [3]. The GPL and LGPL are copy-left licenses, which means that the software cannot become proprietary (although the LGPL, for libraries, can allow linking to proprietary code). The MIT and BSD licenses let anyone do almost anything with the code except sue the authors. Recently, GNU (GPL) text has been revised and a draft of new version (v3) of that license has been released for public review in January 2006. The new version outlaws the use of the license in restrictive technologies such as Digital Rights Management. The draft also includes a clause that any software licensed under the GPL must offer free and unrestricted use of any patented technology that it may contain.

Patterns in language

Over the years programming experts have developed and selected many ways to solve the licensing problems, although these have typically been static they have addressed the main requirements for FLOSS. Here we concentrate on illustrating the open licensing practices that have proved to be useful over the last decade. It is recommended that any time you are thinking of editing the licensing text you should consult your legal advisor to make sure that the changes will still comply with the legal requirements of your jurisdiction or product environment.

Although there is a growing trend to move beyond traditional technologies, the stand-alone licensing methods still provide the basis for most license managers. In this paper, we present four open source software licensing patterns that are the basic types of open source license, and form an extension to an existing software licensing pattern language presented in [1]. Our previous work identified fifteen basic software licensing patterns that can be used by novice programmers to select the licensing model that would provide the best solution for their proprietary software under a non-open license. Figure 1 provides a high level overview of the existing language and the FLOSS licensing patterns added to the diagram.

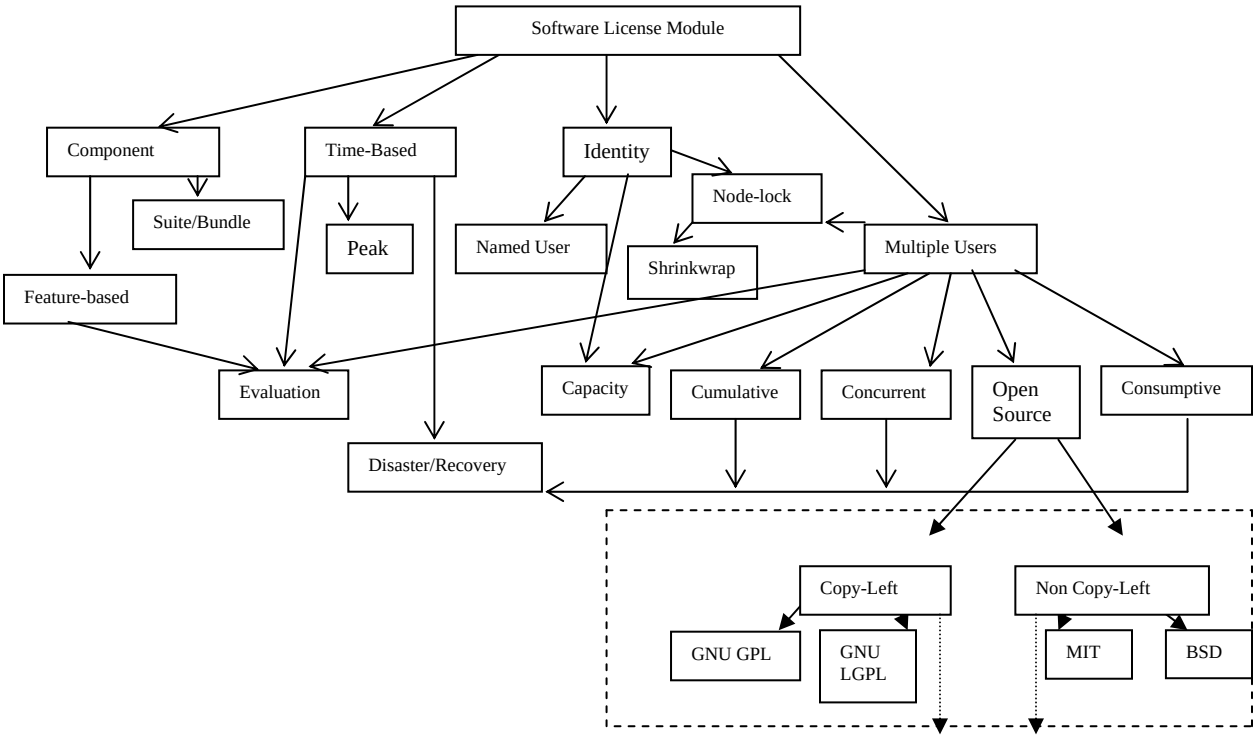

Figure 1: Software licensing patterns

Pattern Reference table

Pattern Name	Problem	Solution
Software License Module *(Top level pattern)*	There is a need in software development to ensure the rights of creators, intellectual property owners.	To meet the needs of software protection a separate licensing module should be provided to manage (legally) the use of software.
GNU GPL (General Public License)	To ensure maximum contributions with the greatest feedback without proprietary lockup: an open paradigm.	Use the GNU GPL license. It is the best possible way to achieve a permanently open source code base, involve open collaborative development.
GNU LGPL (Lesser General Public Software License)	To ensure maximum contributions for a subroutine library with the greatest feedback without proprietary lockup but with flexibility: a flexible open paradigm.	Use the Lesser General Public License (LGPL), which has been derived from the GPL and has been designed for software libraries. Unlike the GPL, an LGPL-ed program can become a part of a proprietary program.
Berkley Software Distribution (BSD) license	How to acknowledge the original authors of the software with no restrictions on how the source code is used or distributed. How to prevent the user from including an original author's name in the product promotions.	Use the BSD license. It carries very liberal clauses with it. If you want to spread your ideas and you do not mind someone else to accumulate the financial gains for your contribution you should use BSD license.
MIT license	You would like to allow that your name and software be used for the product promotion.	Use MIT license. Under MIT license the source code of your program can be used (integrated) in another computer program.
The following **proprietary** patters can be found in H Kaminski & Mark Perry "The Pattern Language of Software Licensing", *Proc. 10th European Conf. Pattern Languages of Programming* EuroPLoP 2005, (Konstanz: UKV Konstanz GmBH, 2006)		
Identity Software License	There is a need to restrict the authorized use of software to a specified user or to a specific machine. Sometimes a vendor needs to specify the hardware component that can be used to run an application.	Assign the license to the specific individual or to the identified machine or its hardware
Multiple Users Software License	A company needs a predefined number of software licenses to be available at all times. It provides an opportunity for the vendor to gather single licenses into one group and put the restriction on the whole group.	Assign a number of the licenses allowed for the concurrent use. Every time a user requests a license one should be issued to him/her if and only if the number of the licenses in use does not exceed the number of licenses allowed.
Time- Based Software License	Sometimes there is a need to have an application that should be used only for a specified period. It is important to prevent the customer from running the software after the agreed period.	When the application is first installed, it should make an entry in the system's registry providing the necessary information about the time restrictions on the operation of the software. The user should not be able to run the software after the expiry date

Pattern Name	Problem	Solution
Named User Software License	It is useful to have a mechanism that restricts the use of software to one person especially in a setting where software is Identity based (e.g. e-mail applications or business transaction applications).	Include the exclusive user name in the licensing module.
Node-lock (named host) Software License	There is a need to restrict the use of software to a specific piece of hardware.	Provide a set of parameters that uniquely identifies that computer into the software license module. On installation, identifiers from the hardware are written into the executable object file, which can then only be run on that particular hardware.
Capacity Software License	To restrict the use of software to the characteristics of the machine where that software is executed.	The predefined number of allowed capacity units should be written into the license application.
Concurrent Software License	It is beneficial for a company to have a predefined number of software licenses available at all times. There is a need to restrict the use of software to a defined number of concurrent users.	The number of the licenses allowed for the concurrent use should be indicated in the License Manager data table. Every time a user requests a license one should be issued to him/her if and only if the number of the licenses in use does not exceed the number of licenses allowed.
Consumptive Software License	To provide a number of software licenses to specified number of users.	A license used once cannot be retrieved or used again. The licensed software cannot be further used once the number of allowed uses is exceeded.
Cumulative Software License	Let the customer to use the software and calculate the payments based on the actual usage. You need to provide a software license that will allow you to know how many times the users run the application.	Where a vendor uses the number of executions as the billing system basis then the counter has to be implemented such that it counts the number of executions and records them into the database where the information about the total usage is kept. If the vendor specifies the time units to be used as the base for the payment then the database file will hold the number of total time units used since the last billing.
Feature- based Software License	To restrict the software use to the pre-defined set of functions.	The licensing module developer has to specify a list of features (functions) of the application that can be locked and unlocked for the user. This licensing model has to contain a detailed description about the availability of the features. The function locking mechanism has to be implemented for each function from the list in the License Manager.

Shrinkwrap Software License	To restrict the software use to one computer system the vendor places the license in the package along with the software and documentation.	When the application is first installed, it should write an application key (also known as License key or Serial Number) into the system's registry. Every time the user requests an operation of the software the license manager should call upon the system registry database and check if the serial number for that application is correct.
Evaluation Software License	Many software vendors let their customers try software before they enter into some other kind of license. There is the need to provide 'try-before-you-buy' software license.	The Evaluation Licensing Model is usually implemented as Time_based, Feature_based or Consumptive License.
Suite/Bundle Software License	To provide a license for a set of software application. To restrict the use of two or more products which are licensed individually to the limited number of concurrent uses?	The license module should link the products in the bundle by a common license key and the password. It is possible to represent all the products in the system by a single bundle license. The number of the available licenses should be indicated in the License Manager's module.
Peak Software License	To manage software use based on the different time of the day.	The time checker has to be included in the licensing module. Since the usage price depends on the time of the day the time checker has to be able to recognize and return the exact time meaning hour and minutes values. The time checker has to communicate with the system's clock when it requests the current time and then it sends that information to be recorded in the usage log file.
Component Software License	To restrict the use of software to pre-defined components.	The licensing software developer should create several licenses that each carry unique license key. Each module should carry a separate license that can be combined with other product's license.
Disaster-Recovery Software License	To manage software use in the disaster-recovery scenario.	It can be implemented as a Time_based, Cumulative or Consumption based.

Category A. Copyleft

Pattern 1: GNU GPL (General Public License) Software License (version 2)

Problem: *Ensuring maximum contributions with the greatest feedback without proprietary lockup: an open paradigm.*

Context: The code must be 'free': freely available source code, useable by anyone, adaptable by anyone and redistributable on the same terms by anyone. Adapted code for distribution requires the same licensing. The free terms are inseparable from the code.

Forces:

- The GPL is a straightforward yet powerful licence. When code is released under GPL, there is an obligation that the source code is accessible and any software deriving from that code. A developer who takes code under a GPL licence and incorporates it into their own code is obliged to make the source code of the entire product available to its recipients upon distribution under the GPL. The license cannot be separated, or split from the code, nor can it be revoked if its terms are met. The copyright in the code is held by the authors of this code, but this copyright is limited by the terms of the license, i.e. no rights other than those under the license can be enforced.
- Code forking is not a problem with FLOSS, but license forking can be [4]; the GPL prevents code licence 'forking', where there is more than one kind of licence for the same code, by locking the license to the code, unlike some other types of license. However, some licences offer a choice of GPL or other licence type for the user.
- This licence choice of GPL for developers means that their source code will always be available freely and never tied into closed proprietary code. Other license types can give more 'downstream' flexibility for inclusion.
- The GPL allows for the code package to be sold, but not licensed under other terms, so that a charge could be made for supplying the GPL code by download or on CD but no further charges (such as an ongoing licence fee). To prevent even this charge a different licence would need to be used.
- The GPL prevents the monitoring or auditing of the distribution of the code under the license; other license types allow this.
- Under the GPL, terms anyone can redistribute and change/or modify the source code, and then distribute the new version, so long as it is under the GPL. There are some within the community that look to enforce the license terms to prevent 'free riding' on code development without giving back any changes to the community. [5]
- The GPL ensures that you are recognised through attribution as the contributor of the code, and you want to keep copyright in your code. Copyright is the legal

force that supports GPL, as copyright means that the code can only be replicated under a license from the copyright owner, so it is the underlying backbone to the license and enables its enforcement. Without having copyright, simply placing the code in the public domain, would allow for appropriation of the code [6].

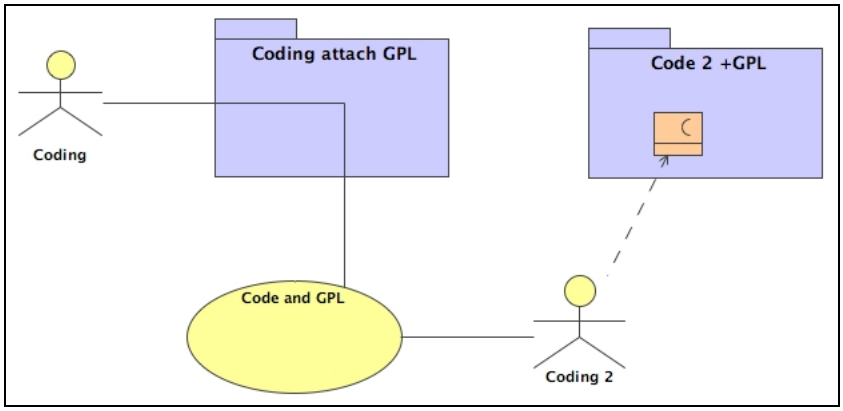

Figure 2 Whatever the input, the GPL remains

Solution:

Use the GNU GPL license. This is the best possible way to achieve a permanently open source code base, involve open collaborative development, and satisfy the forces. The full text of a GNU GPL license is available at [7]. Although having a full text of GPL license available is seen as a good programming documentation practice, it is not necessary to attach it to the software. It is enough to inform a potential user that this license type applies to your program.

To license your software with a GNU General Public License you should also attach the following notices to the start of each source file of your program:

1. Author's name
2. Notice saying that this program is free software
3. That the program can be redistributed it and modified under the terms of the GNU General Public License as published by the Free Software Foundation
4. Note saying that you do not offer any warranty for this software
5. The information that a full copy of the GNU General Public License can be obtained by writing to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. [4]

If you are employed and the code is part of your work or you are working on a research project, you should also get your employer or your supervisor, to sign a "copyright disclaimer" for the program, if necessary.
The GNU GPL does not permit incorporating your program into proprietary programs.

[note: GPL version 3 is still under development]

Pattern 2: GNU LGPL (Lesser General Public Software License)

Problem: *Ensuring maximum contributions for a subroutine library with the greatest feedback without proprietary lockup but with flexibility: a flexible open paradigm.*

Context: The code must be 'free': freely available source code, useable by anyone, adaptable by anyone and redistributable on the same terms by anyone, but it can be used with proprietary code. Adapted code of that library for distribution requires the same licensing. The free terms are inseparable from the library.

Forces:

- Sometimes there is a need to allow use of a particular library in non-free programs, which enables a greater number of users to benefit from the open source software paradigm. For example, many proprietary programs use the GNU C Library which in turn enables many people to use software with the whole GNU operating system, as well the variant GNU/Linux operating system. Another licence, such as GPL, could be used to prevent use with proprietary programs.
- The LGPL requires that there is no requirement to charge for redistribution, if a charge for the library is required, then another license should be used.
- The determination of intellectual property rights in a library may be uncertain, the LGPL disclaims any warranty, although you can choose to give a warranty if you wish to.
- You want other people to incorporate your library software into proprietary programs, but the LGPL maintains the openness of the library.

Solution:
Use the Lesser General Public License (LGPL) that has been derived from the GPL and has been designed for software libraries. Unlike the GPL, a LGPL-ed program can become a part of a proprietary program.
As in a GNU GPL license it is enough to inform a user that such license applies to your program. To license your software libraries with a LPGL you attach the following notices to the start of each source file of your program:

1. Author's name
2. Notice saying that these libraries are free.
3. That the program can be redistributed it and modified under the terms of the GNU Lesser General Public License (LGPL) as published by the Free Software Foundation
4. Note saying that these software libraries can be incorporated into proprietary software.
5. Note saying that you do not offer any warranty for these libraries.

6. The information that a full copy of the GNU Lesser General Public License can be obtained by writing to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. [4]

Like in a general GNU GPL license pattern, in this license type, if you are employed and the code is part of your work or you are working on a research project, you should also get your employer or your supervisor to sign a "copyright disclaimer" for the program, if necessary.

Category B: Non Copyleft

Pattern 3: Berkley Software Distribution (BSD) license

Problem: *How to acknowledge the original authors of the software with no restrictions on how the source code is used or distributed. How to prevent the user from including an original author's name in the product promotions.*

Context: You have created some software that you want others to use as widely as possible, but with no restrictions on the downstream use of the software.

Forces:

- You do not want the software to be used in any advertising.
- License forbids the use of software to promote the product on a market.
- You do not want your name to be used by anyone to promote their program.
- You do not want someone else to incorporate your program into their software and claim that they wrote it.
- You do not mind if others making money out of some combination of programs that include yours.

Solution:

Use the BSD license. The Berkeley Software Distribution license, known as BSD license, gives the most freedom to the software user. It carries very liberal clauses with it. If you want to spread your ideas and you don't mind someone else to accumulate the financial gains for your contribution you should use BSD license. All users are free to integrate your code into their own software without giving you a credit for your work. This license forbids the user to use the developer's name in public to promote the product. The license also carries a denial of any warranty clause. This protects the author from any damage caused by the software. [8] Any BSD code can be sold or included in proprietary products without any restrictions on the availability of your code.

To include BSD license with the program the software developer needs to fulfill three requirements that claim, that the license has to be retained.

It has to contain:

1. A note, that a redistribution of source code must retain the copyright notice, the list of conditions and the license disclaimer.
2. A note that a redistribution in binary form must reproduce the above copyright notice,
3. The list of conditions and the disclaimer in the documentation and/or other materials provided with the distribution. [9]

The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission. You need to include a full text of the license with your software. See: (Appendix 1)

Pattern 4: MIT license

Problem: *You allow that your name and software will be used for the product promotion.*

Context: You need to preserve the license with no restrictions on the downstream use of it.

Forces: License allows the use of software for promotion. You might benefit from the product being used in advertisement.

Solution:

Use MIT license. Under MIT license the source code of your program can be used (integrated) in another computer program. If you don't mind the software creators to use your name in their products' promotion you should use MIT license. MIT open source software license originated at Massachusetts Institute of Technology. It is very similar to the BSD license except it contains no restrictions as to the use of the original author's name in product promotions. The text of the MIT license includes the permission to any person obtaining a copy of software to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the product. To license your product with MIT license you should include its text with the software (Appendix 2)

Conclusion

The FLOSS model offers some exciting opportunities for the software market. For computer scientists and developers, open source licenses offer the opportunity to get their ideas out to the world, with the opportunity for collaborative feedback.

With a careful choice of license types, it is possible to select the most appropriate environment for development, and often the greatest benefit is to be had from using open source software. Naturally, choice of license should be primarily determined by the intentions of the software developer and the desired outcomes. Although FLOSS is seen by some as a "long-term developer mindshare threat", it does offer an alternative means of software licensing over the proprietary system, and can offer a beneficial and even profitable environment for many players.

References:

[0] Thanks to Natural Science and Engineering Research Council for funding and shepherds Eugine Wallingford and Robert Hanmer.

[1] Kaminski, H. Perry, M. “Software Licensing Pattern Language”, EuroPLoP ‘05 Conference, Irsee, Germany, 2005

[2] Richard M Stallman, “Free Software: Freedom and Cooperation”, Speech at New York University, New York, 29 May 2001, http://www.gnu.org/events/rms-nyu-2001-transcript.txt (27 August 2001).

[3] http://www.opensource.org/licenses/

[4] FLOSS encourages forking of the code base, and although code forking is seen as undesirable in many proprietary development situations (as it is wasteful of resources), with FLOSS it enables a number of ‘bake-off’ versions to be developed, with the best surviving. Although initial license forking is possible (for example MySQL AB offers their software under GPL or under an OEM Commercial License), once GPL is chosen then downstream versions remain under GPL.

[5] Most GPL enforcement involves persuading companies to put code that embodies GPL code under GPL. See < http://www.gnu.org/philosophy/enforcing-gpl.html >.

[6] There have been few cases taken as far as court action, but the threat of court action is usually enough to get the parties to comply with the license. See < http://gpl-violations.org/ >

[7] GNU General Public License v.2 at http://www.gnu.org/copyleft/gpl.html

[8]Weissmann, Marcus, “Open Source Software Licensing”, 2005, http://www.-mweissmann.de/downloads/OSS-Licensing-Paper.pdf

[9] http://www.opensource.org/licenses/bsd-license.php

Appendix 1 **(BSD License)**

Copyright (c) <YEAR>, <OWNER>All rights reserved.

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

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

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.[2]

Appendix 2 (MIT License)

Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.[2]