15 Handling of Software Patents and Trademarks in Open Source Software Licenses(*)

Research Fellow: Masayuki Hatta

Open source software, such as Linux, can be freely used, modified, and re-distributed under an open source license. Its use is becoming widespread in Japan. Software licenses are, in principle, based on a copyright, but in recent years, active movements are taking place toward incorporating clauses concerning patent retaliation and trademarks into them, thereby controlling industrial property rights other than copyrights by using copyrights as leverage. In addition, open source communities, which have tended to avoid software patents, are now aiming to utilize patents strategically by forming patent pools designed for open source and working on open patent licenses, so as to clearly assure free development. Based on the latest research trends and the hearing survey targeting specialists in this field, this study closely examines and compares the details of patent and trademark clauses included in open source licenses, which differ from license to license, and comprehensively reviews how those rights other than copyrights are treated in the context of open source in actual cases, with the objective of finding out how the concept of open source will affect the future software development.

Ⅰ Introduction

Software, in general, consists of programs (algorithms) and data. Copyright protection shall be given to both programs and data if they are recognized as creative works in the legal sense. This is currently a matter of common sense, but at the beginning, software, and in particular, the use thereof, was not assigned so much economic value. In the 1970s, software was considered to be of no value. However, in the 1980s, its value was discovered, and people started to discuss the appropriate system for protecting it. In the end, it has come to be protected as a work under the Copyright Act.

In the past two decades, not only software but also creative works in general have been found to have considerable economic value. We should particularly note that software is a unique type of property which can be protected by a copyright, as well as by other rights of a completely different nature, such as a patent right and a trademark right. The issue of whether or not software itself should be protected by a patent was addressed in the U.S. Supreme Court judgment in Diamond v. Diehr,[1] and it is still being discussed in Europe. There are deep-rooted opposition movements against software patents. It is difficult to understand that issue without understanding the root of these movements. In Europe, the anti-patent group and the pro-patent group had a fierce debate over whether or not to adopt the directive on the patentability of software proposed by the European Commission.[2] On July 6, 2005, the European Parliament rejected the proposal by a substantial majority.

Under such circumstances, open source software exists as software which is developed basically for the purpose of licensing. It differs from a general type of software in many aspects in terms of the protection of rights.

This study aims to review how patents and trademarks, in addition to copyrights, are treated in open source software licenses, and to clarify the ideal form of a license for open source software.

Ⅱ What Does *Open Source* Mean?

Open source software, as used in this study, means software which is opened under a software license that complies with the criteria developed by Open Source Initiatives[3] under the title of the Open Source Definition.[4] In principle, anyone can freely modify or re-distribute open source software.

With GNU/Linux, an operating system that has been remarkably diffused in recent years,

(*) This is an English translation of the summary of the report published under the Industrial Property Research Promotion Project FY2010 entrusted by the Japan Patent Office. IIP is entirely responsible for any errors in expression or description of the translation. When any ambiguity is found in the English translation, the original Japanese text shall be prevailing.

listed first, open source software is now rapidly penetrating into our everyday life. Many home appliances and mobile phone units use some sort of open source software. In this respect, we have become licensees without knowing it.

The term *open source* itself is said to have first appeared in 1998,[5] but the origin of the very concept of making software legally available to anyone to use freely by means of an *open* copyright license goes back to the free software movement that took place in the 1980s.

There are various types of software that are called open source software. What is important is that under a software license, anyone can use the software *freely*. Software that cannot be used "freely" is called *proprietary* software.

The definition of free software, developed by Mr. Richard Stallman, the founder of the GNU Project and the President of the Free Software Foundation, is also widely used. To put it simply, according to Mr. Stallman's definition, open source software is, in most cases, software that anyone can use freely without charge. Representative examples of this kind of software include Linux, Mozilla Firefox (a web browser), and Android, which was recently released by Google as an operating system for mobile phone units.

While carrying out this study, I got the impression that the term *open source* is being used in two different ways. In my eyes, open source is attracting attention not only from the legal aspects as described above, but also as a development scheme. The specific example is what is generally called a bazaar development process, wherein, with no hierarchy or chain of order, a number of people can freely participate in the development of software. Such scheme is often called open source. Open source as a legal state seems to support open source as a development scheme and serve to facilitate the development process. I chose open source software licenses as the main topic for this study with the objective of finding out how it affects a bazaar development process.

Actions are taking place toward applying a concept, similar to open source as a development scheme, for purposes other than software development. A famous example is Creative Commons,[6] advocated by Professor Lawrence Lessig. The core concept of Creative Commons is to make works such as books and other text materials available so that anyone can use them, with the goal of promoting the reuse and secondary use of works as well as the creation of innovation.

Recently, considerable attention is also being paid to a new concept, open hardware. Along with this new trend, for example, the Open Source Hardware (OSHW) Statement of Principles and Definition v1.0[7] is being established.

Another example of action under the influence of the principle of open source is a community activity called Peer-To-Patent,[8] which is designed to enable anyone to participate in the process of examining the patentability of inventions, which is usually conducted by patent offices.

Ⅲ Earlier Studies on Open Source

The concept of open source influences economics as well. Following the great success of Linux and GNU/Linux, the views that open source methodology would work, or would work at least under certain conditions, spread widely. It is natural that such views led to the attempt of gaining some knowledge from this methodology.

Chesbrough (2006) states that *open innovation*, which involves open source as a development scheme and also legally open, is considerably helpful in promoting innovation and it is necessary to develop business models based on this new concept. His view became very influential. Others followed him and started to talk about open innovation or open business models.

Katz & Allen (1982) discusses the Not Invented Here (NIH) Syndrome, and to breakthrough this way of thinking is another main theme of this study. When launching an open source project as a development scheme for open innovation, we have to procure part of the human resources and other development resources from the outside. In this context, it is necessary to consider what we should do to collaborate with parties outside the project. One desirable approach may be to develop a license that can facilitate collaboration with outside parties. This view is heard from the field of economics as well. In conjunction with studies on NIH, it is now impacting discussions on various issues, including the open licensing strategy and standardization strategy.

Ⅳ Characteristics and Classification of Open Source Software Licenses

Open source software can be protected by a copyright. In general, in order for a third party to

use such software, a copyright license is granted. The copyright holder, who is the producer of the software in most cases, grants permission for reusing the software under certain conditions of license. The current common practice is to choose the one that best suits the relevant license from among some typical, widely-used samples of open source software license agreements.

There are several types of licenses used for open source software. In most cases, they satisfy or at least do not violate the criteria specified as the Open Source Definition. GNU General Public License (GPL)[9] is a highly notable type of open source software license. This license is applied to Linux Kernel. Berkeley Software Distribution (BSD) License and Apache License are also frequently used. Among these major types of licenses, GPL is actually used for 50 to 60% of all open source software licenses, BSD License for 10 to 20%, and Apache License for 10 to 20%. The rate of use of Apache License is recently showing high growth.

These open source software licenses are also drawing attention from the software industry and those actually engaged in software development, and they are also considered to be worthy of note from a legal perspective. Since the beginning of the year 2000, studies on open source software licenses have progressed in various fields. Having reviewed earlier studies, I found that the former scholars and legal professionals were interested especially in copyright licenses. Considering that software can be protected by copyright, this may be a matter-of-course phenomenon. The object that attracted the most interest was the feature of copyleft. *Copyleft* is a coined word that means the reverse of copyright. It refers to the concept of reciprocity, that is, if any modification is made to the software covered by an open source software license, the licensee who has made such modification must open the relevant source code. This rule is generally considered to be a feature of GNU GPL, but actually, similar conditions are included in other licenses. Lerner & Tirole (2002) conducted a leading study that focused on this feature.

Ⅴ Open Source Software and Software Patent

1 Treatment of software patents in open source software licenses

An open source software license is exactly an approach of providing legal assurance that anyone can freely use software, on the basis of a copyright. On the other hand, a patent is a legal framework of granting an exclusive right to the inventor, or in other words, preventing anyone from freely using the invention, thereby returning the benefit to the inventor so as to increase the incentive to disclose inventions. Thus, open source and patents are incompatible by nature. This problem has been too serious to ignore, as open source software became significantly valuable in economic terms and profit-making enterprises with large patent portfolios started to take part in the development of open source software.

What makes the problem more complicated is that a copyright and a patent right, both means of legal protection for software, are completely independent from each other. Even in the case of open source software covered by a copyright license, if it involves a technology protected by a software patent, it may not be made available to everyone to use freely without obtaining a separate license regarding the patent.

In the context of open source, the most popular existing way to deal with the issue of a software patent is to include Patent Clause concerning the treatment of a patent in the conditions of an open source software license.

Among many types of open source software licenses, many are not very recommended, and only seven types are in relatively frequent use. The BSD License is the only one that does not include any Patent Clause, whereas the Apache License, GPL, Lesser General Public License (LGPL), Common Development and Distribution License (CDDL), Mozilla Public License (MPL), and Artistic License include Patent Clauses (e.g. Section 11 of GPL and Section 13 of Artistic License).

2 Validity of Patent Clauses

We should first question whether or not those Patent Clauses can be regarded as being legally valid. While there is no precedent case which directly addressed the dispute over Patent Clauses, the legal validity of an open source license was disputed squarely before court in Jacobsen v. Katzer.

Professor Robert Jacobsen of the University of California, Berkeley is one of the major members of the team that developed open source software, the Java Model Railroad Interface (JMRI). JMRI was opened under the Artistic License.

Meanwhile, Matthew Katzer, who ran a company called KAMIND Associates Inc. in the State of Oregon, copied part of the source code of JMRI and incorporated it in his company's products, without complying with the clauses presented by Jacobsen as the conditions for granting a license.

Jacobsen brought a case before the U.S. District Court for the Northern District of California against Katzer for non-fulfillment of the agreement and also for copyright infringement, with a motion for a preliminary injunction. In 2007, the District Court dismissed Jacobsen's motion for a preliminary injunction for copyright infringement, holding that the relevant conditions of the license do not limit the scope of the license. Dissatisfied with this, Jacobsen appealed to the U.S. Court of Appeals for the Federal Circuit (CAFC).

The CAFC acknowledged that the open source license is enforceable, and determined that Jacobsen's license conditions are set as those for granting a license regarding the copyright for the software, and that any user of the software who fails to comply with these conditions shall be deemed to go beyond the scope of use permitted under the license and such use constitutes copyright infringement. In conclusion, the CAFC remanded the case to the District Court to examine Jacobsen's likelihood of success on the merits and irreparable harm.[10]

In the judgment of the remanded case, the District Court dismissed Jacobsen's motion for a preliminary injunction again, on the grounds that Jacobsen did not demonstrate that he would sustain any irreparable harm unless the alleged infringement were stopped.

This case finally came to an end through a settlement between the parties in 2010. From this precedent case, I drew a view that the CAFC took the stance to confirm the legal validity of an open source software license, and presumably would also take the same stance for Patent Clauses.

3 Two Major Types of Patent Clauses

Open source software licenses often include Patent Clauses. Apache License 2.0, Section 3, *Grant of Patent License*, may be a typical example of such clause, and can be divided into the following two parts.

(1) Automatic licensing

The first sentence of the Patent Clause, Section 3, *Grant of Patent License* can be paraphrased as stating that the copyright holder for the software covered by Apache License 2.0 or any part thereof (Work or Contribution) shall grant to the licensee (You) a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable patent license for the use (including modification and sale) of the software.

(2) Patent Retaliation

Patent Retaliation is a mechanism that has been seen in many open source software licenses these days.

The second sentence of the Patent Clause, Section 3, *Grant of Patent License*, provides, if the licensee institutes patent litigation, alleging that the software covered by Apache License 2.0 or any part thereof (Work or Contribution) constitutes direct or contributory patent infringement, any patent licenses granted to such licensee under Apache License 2.0 shall be put to an end. After that, the licensee might be sued by the licenser in a patent infringement lawsuit. Similar clauses are adopted in GPL v. 3 and other types of open source software licenses.

4 Discussions on Patent Clauses

An implied license is one of the topics taken up for discussions on Patent Clauses. This may also be regarded as an issue relating to legal estoppel. The origin of the concept of automatic licensing is *implied license*. Open source software, by its nature, must allow the licensee's free use, even where the licenser has not explicitly permitted, he/she shall not be allowed to subsequently deny its validity, which has been in effect permitted. Inhibiting such free use by way of a patent license constitutes legal estoppel.

GPL v2 does not include any such explicit permission as an automatic license, and adopts an implied license. Although various opinions are heard on this point, it is generally understood that an implied license is valid unless the original software is modified. If the licenser transfers the software to another person, without modification, he/she would not be allowed to subsequently apply any additional restrictions. However, it is very controversial whether or not an implied license is valid even when the licensee makes a modification to the software.

An open source software license is generally defined as a software license that complied with the Open Source Definition. There is an argument that the compliance with the Open Source Definition is, in effect, the basis for asserting the validity of an implied license for all licenses. With regard to the BSD License, which does not involve patents, it is also argued that as long as it complies with the Open Source Definition, it could be deemed to grant an implied license. Items 6 and 7 of the Open Source Definition provide for the prohibition of discrimination by field and prohibition of distribution of rights.[11] Some argue that, under these provisions, putting restrictions on rights depending on the purpose of use or the licensee would be contrary to the Open Source Definition, and that an open source software license subject to such restrictions could never be deemed to be in compliance with the Open Source Definition.

There was controversy over the MPEG eXtensible Middleware (MXM) License's compliance with the Open Source Definition. The point at issue was what types of license should be applied when implementing technical standards which contain patents by means of open source software. The MPEG Working Group basically adopted the MPL but removed the Patent Clauses, in an attempt to design a license which requires the licensee to acquire patent licenses separately.

Another point is that there is considerable difference between licenses. For instance, under the Artistic License, if the licensee institutes patent litigation with regard to the software or any derivative software, the license shall terminate.[12] This clause has a broad scope. On the other hand, the Apache License and GPL v3 limit the scope of patent claims in some ways.

A possible future approach is to directly control the use of software via patent licenses. The licensing scheme for WebM, an open video format for the web by Google, may be a notable example. This new format involves another web video standard, H.264, which cannot be made open solely at Google's discretion. As the patent pool developed by MPEG LA contains various patents relating to WebM, Google properly ought to participate in MPEG LA. However, in that case, WebM might not be able to meet the conditions of open source software, i.e. non-exclusive and no-charge. Google seems to have devised the idea of licensing WebM in order to clear this problem.

A remarkable feature of the license of WebM is that Google intends to provide a copyright license and patent licenses in one package, while completely separating these two categories of licenses. More specifically, Google's plan is to apply the BSD License for the source code, and to grant patent licenses for the implementation of JavaScriptV8, which it has developed, under the scheme of Additional IP Rights Grant. It is of great interest that Google is to grant a license in the form of a Specification License for WebM even when a third party has re-implemented this technical standard independently.

As I see it, Google aims to avoid forcing the WebM licensees to enter into license agreements with MPEG LA to use H.264, and to this end, on the presupposition of the existence of YouTube, a widely spread web service, it makes a kind of threat to a third party that if the party sues a licensee for patent infringement with regard to WebM, Google would terminate the various licenses it has granted to the party, thereby trying to maintain an open source software license.

Ⅵ Trademarks under Open Source Software Licenses

In the context of open source, controlling the use of software via trademarks is a relatively new idea.

Like a patent license, a trademark license for open source software is basically regarded as a control means other than a copyright license. It is used directly for the purpose of maintaining some sort of quality of software.

For instance, with regard to Linux Kernel, a representative example of open source software, rights for "Linux" or any other similar trademark had not been obtained by any person in any country until 1994. On August 15, 1994, William R. Della Croce, Jr., an attorney living in Boston, filed a trademark application for "LINUX" in the field of computer operating systems (Serial No. 74560867), and this trademark was registered on September 5, 1995 (Registration No. 1916230).[13] This person had no relation with Linux. After that, he attempted to collect royalties (10% of the product sales) from Red Hat and other affiliated companies of Linux. In 1996, Linus Torvalds, who developed Linux, with other persons in the industry, brought an action to claim invalidation of the trademark registration of "LINUX." In 1997, this case was resolved by a settlement wherein Della Croce, Jr. should assign the ownership of the trademark to Torvalds. Since this case, Torvalds has held the ownership for Linux-related trademarks registered in the United States, Germany, EU, and Japan, while the Linux

Trademark Institute (LMI), set up within the Linux Foundation (a consortium for the promotion of Linux), has actually performed the management of these trademarks. The LMI's policy is to grant no-charge, permanent, and worldwide sub-licenses for Linux-related trademarks, on condition that the licensees should avoid using the trademarks in a manner that could jeopardize Torvalds' trademark rights, and should indicate those trademarks properly.

The Linux trademark case can be referred to as the endeavor to realize open source via a trademark. Meanwhile, a trademark is also used in order to put some restrictions and exert influence on open source development.

In the case of Firefox, a famous open source web browser, a software license is granted in the form of an open source software license, Mozilla Public License (MPL). The Mozilla Foundation, which is in charge of the development of the software, has laid down the Mozilla Trademark Policy for Distribution Partners. According to this policy, distributors may use the name, logo or any other artwork of "Mozilla Firefox" only if they distribute unaltered official binaries provided by the Mozilla Foundation. The official purpose of this restriction is to maintain and improve the brand image of Mozilla by quality assurance given by the Mozilla Foundation itself. At the same time, the Mozilla Foundation receives as much as 60 million dollars from Google in exchange for designating Google as the default search engine of Firefox, and for this reason, the foundation presumably wishes to keep part of the official binaries of Firefox unchanged, while providing it as an open source web browser.

A Linux distributor, Debian, declared that it would not accept Mozilla's trademark policy, and has decided to use "Iceweasel," instead of "Firefox," in logos, artworks, filenames, and any other kinds of names to be attached to its own, modified versions of Firefox that it distributes.

This is not unusual in the context of handling trademarks. However, I find it strange that the Mozilla Foundation allows distributors to modify Firefox but restricts them from using the name "Firefox," which seems to be contrary to the concept of open source. Here is the limit of Mozilla's trademark policy. While Debian's case was settled by changing the name, some cases have been brought to court in trademark litigation. Oracle, which bought Sun Microsystems, sued Google, alleging that Google directly copied Java code and used it in its mobile phone platform, Android. In this noteworthy case, the parties disputed various issues, including those concerning a copyright, patents, and trademarks. Oracle uses the "Java" trademark as a means to control the use of Java technology. Google wished to use Java in Android, but faced difficulties in implementing Oracle's Java Platform, Micro Edition (JavaME), so it developed another Java implementation called Dalvik. However, Google is not allowed to call Dalvik a Java implementation because it bypassed the Java Community Process (JCP), which Oracle requires for Java processing.

Ⅶ Other Measures for Open Source

In recent years, movements toward protecting the development of open source software by systems or organizations set up outside the framework of licenses are taking place. In such situation, *collaborative licensing* is being adopted as a research and development model. When several parties work together to develop something, they would have to go through too many steps if they were to license the relevant copyrights, patents, and trademarks, one by one, to one another. The collaborative licensing model has been developed to make this process more simple and easy. In this section, collaborative licensing is roughly divided into two types: licensing based on weak collaboration and licensing based on strong collaboration.

1 Patent Pools

A patent pool is an example of weak collaboration. In principle, participants in a patent pool are supposed to contribute their patents voluntarily. They may select only some of their patents that are less important and contribute them to the patent pool.

Patent pools have been diffused as a licensing scheme in the fields of home appliances and software. They have been introduced in the context of open source only relatively recently.

The Patent Commons Project is a well-known example. It was launched by the Open Source Development Labs (OSDL) on November 15, 2005, and it is currently being managed by the Linux Foundation, which was established through reorganization of the OSDL.

2 Defensive Patent Licensing

A specific example of licensing based on strong collaboration is Defensive Patent License (DPL).

DPL has recently been proposed by Jason Schultz and Jennifer Urban, both of the UC Berkeley. It is a patent license in which the concept of GPL is incorporated. It is designed for directly controlling the use of software using patent licenses, rather than indirect control by way of a copyright license.

3 Contributor License Agreement

When an open source software user has modified the existing source code, etc., he/she has two options for opening the modified version. One is to open only the portion that the user has modified,[14] and the other is to contribute the modified portion to the entity that takes the lead in the development of the software (e.g. the development project), and have it merged into the existing source code (the code base of the main line) under the management of the entity. More specifically, when a Linux Kernel user has developed a new device driver, he/she can simply open the source code of the device driver on his/her own website or by other means, or may contact the Linux Kernel Mailing List, etc. and have the device driver accepted as part of Linux Kernel standard distributions, with the consent of Linus Torvalds and the major development staff.[15]

Where the latter option was chosen, how to deal with a copyright for the contributed portion often becomes a problem. If the development project does not designate any particular condition upon merging the contributed code into the existing source code, the copyright for such contributed code remains in the hand of the person who developed that code. For instance, there is no particular condition set for the merge into Linux Kernel (except for the requirement of the quality of codes), and as a result, Linux Kernel can be regarded as a work under joint authorship owned together with a huge number of copyright holders. Theoretically, when making a decision that could affect the entire work or Linux Kernel in whole (e.g. a change to the license), it is necessary to confirm the intention of all of these copyright holders. Furthermore, someone might later come forward to claim to be the *true* author of the merged code, alleging that the development project has merged his/her work into the existing source code without his/her consent.

To avoid such situation, some development projects require contributors of codes to submit a copyright assignment.

For such process of copyright assignment, a contributor license agreement (CLA) has been becoming popular in recent years.

Under this new type of agreement, the contributor of a code who originally holds a copyright permits the development entity to use the contributed code without restrictions, instead of assigning the copyright to the entity.

Ⅷ Conclusion

Those who have read my discussions on the treatment of patents and trademarks in open source software licenses, might pose a question as to why open source software development projects are trying so hard to restrict patents. While carrying out this study, I went to Europe and talked with various persons engaged in this field, and found that engineers originally had deep-rooted opposition to software patents, which has nothing to do with open source. The great success of open source initiatives launched by those opposed to software patents somewhat encourages those engineers. It is revealed that the raison d'être of software patents has been shaken to a great extent in recent years.

It is also revealed that in the fields of economics and business management, patents have been justified by few scholars. Representative anti-patent studies are those by Boldrin & Levine (2008) and Bessen & Meurer (2008). The latter study, as its title "Patent Failure" clearly signifies, denies the patent system itself.

A typical example frequently mentioned in criticizing patents is the development of the steam engine by James Watt. Watt obtained Patent No. 913 and had adequate funds. Patent supporters argue that the research and development of the steam engine required large capital, and a patent was indispensable to recoup the huge initial investment. Contrary to this argument, Boldrin & Levine state that, after obtaining the patent, Watt became enthusiastic about collecting money from users of the patented technology; he did not himself keep producing steam engines until the expiration of the patent. Viewed from society as a whole, Watt's patent did nothing but impede technological development. This is a criticism against patents in general. Studies in favor of software patents are far fewer.

The Japan Patent Office is said to be examining applications for software patents, in particular business model patents, rather strictly,

in terms of the disclosure requirement and the enablement requirement.

It should be understood that the underlying tone about the treatment of patents in open source software licenses has always been negative, because of the anti-software patent concept that has affected open source software licenses. It can be considered to be the approach to solve problems with software patents by using patent rights as leverage, in a similar manner that copyleft clauses have substantially incapacitated copyrights by using copyrights as leverage.

As the principle of open source is to grant a license without charge equally to anyone, it would be difficult to reconcile this principle with the nature of a patent as an exclusive right. The treatment of patents in open source software licenses represents the efforts to ensure coherence between the two somehow.

The same applies to the treatment of trademarks, as a means to effectively control the use of software while maintaining the basic features of an open source license. Those who lack an understanding on these points might be involved in serious trouble when they intend to license their software or use open source software especially for profit-making purposes. I hope this study can help promote better understanding.

---

[1] DIAMOND v. DIEHR, 450 U.S. 175 (1981)

[2] The proposed directive on the patentability of computer implemented inventions, Commission proposal COM(2002) 92

[3] For Open Source Initiative, see the official website: http://www.opensource.org/osd.html

[4] For Open Source Definition, see the webpage on the Open Source Initiative website: http://opensource.org/docs/osd

[5] See the Open Source Initiative website: http://opensource.org/history

[6] See the Creative Commons website: http://creativecommons.org/

[7] http://freedomdefined.org/OSHW

[8] See the Institute of Intellectual Property website: http://peertopatent.jp/

[9] The original text of GNU GPLv3 is available at GNU Website http://www.gnu.org/licenses/gpl-3.0.html. An unofficial Japanese translation of the original text is available at the Website of the:Information-Technology Promotion Agency (IPA) http://ossipedia.ipa.go.jp/legalinfo/gpl-3.0J.html.

[10] The text of the CAFC judgment on this case is available at: http://www.cafc.uscourts.gov/images/stories/opinions-orders/08-1001.pdf

[11] The Open Source Definition

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.

[12] Artistic License 2.0
(13) This license includes the non-exclusive, worldwide, free-of-charge patent license to make, have made, use, offer to sell, sell, import and otherwise transfer the Package with respect to any patent claims licensable by the Copyright Holder that are necessarily infringed by the Package. If you institute patent litigation (including a cross-claim or counterclaim) against any party alleging that the Package constitutes direct or contributory patent infringement, then this Artistic License to you shall terminate on the date that such litigation is filed.

[13] USPTO Assignments on the Web (http://assignments.uspto.gov/assignments/q?db=tm&rno=1916230), as of March 25, 2011.

[14] In this case, the user him/herself adds the modified portion to the main body (usually by applying patches and building the code).

[15] TOMOYO Linux, developed in Japan, was merged in the main body of Linux Kernel in 2009.