COPYRIGHT AND OPEN SOURCE SOFTWARE LICENSING

By

Chang Sau Sheong

ABSTRACT

Copyright and Open Source Software Licensing

By Chang Sau Sheong

The open source software movement has swept the software industry by storm in recent times, challenging many pre-conceptions about existing software development and licensing models. Copyright have protected software ownership and licensing of much of the closed source software in the market but how does copyright relate to open source software licensing? This dissertation describes the past and present of legal software protection and traces the history of the open source software movement from the Free Software Foundation and Open Source Initiative to the current state of the industry. The various open source licences are compared and explained. The discussion concludes with a discussion on the legal enforceability of open source licences.

TABLE OF CONTENTS

LIST OF TABLES

INTRODUCTION

The open source software movement has caused a large ripple in the software industry with its radical and seemingly counter-business licensing mechanism. Open source protocols and software have traditionally dominated much of what is known as the Internet, legacy from the days when the Internet was just a small collection of inter-connected networks, playground to the emerging hacker culture. This is clearly evident as most of the pre-dominant software used to operate the Internet is open source related. However, the Internet has evolved into a larger, more diverse and commercialized domain of big businesses and inevitably these social sub-cultures clash and legal issues arise as a result. This dissertation investigates the various aspects of open source software licensing, and examines the relationship between legal software protection, the laws that have been created to legislate software licensing, and open source software licences. This dissertation also investigates the enforceability of open source licences.

COPYRIGHT AND LEGAL PROTECTION OF SOFTWARE

Historical perspectives in legal software protection

Technology often advances faster than legal procedures that legislates the technology, and legal software protection is a prime example. Legal software protection falls under the umbrella category of intellectual property law, an area of law that deals with the legal rights associated with property that is intangible.

Protection mechanisms that come under this category includes copyright, patents, trademarks and others which are often not related to each other beyond being under the same group of property rights. Along with software intellectual

property laws covers a wide range of non-tangible assets such as literary and artistic works, designs, marks used by traders and even commercial goodwill.

Historically, the first form of intellectual property protection explicitly granted to software was patent. Applied Data Research (ADR) was to be the first company granted a software patent of commercial importance in 1968. However the practice of patenting software stopped in the 1970s as the courts and patent offices frowned on using patents to protect software. The European Patent Convention (1973) went as far as to exclude software from patentability in Europe.

At the same time, copyright gained gradual acceptance as the main means of legal software protection. In 1978, the US National Commission on New Technological Uses of Copyrighted Works (CONTU) recommended to apply only copyright to software protection. The US eventually became the first country to legislate software copyright protection in the Software Copyright Act in 1980.

Copyright

Copyright, is "the protection of works of artists and authors giving them exclusive rights to publish their works or determine who may so publish"[1]. It is a property right that protects certain types of works including original literary and artistic works, films and sound recordings, and typographical arrangement of published editions. Software is protected by copyright as a form of literary work. In some literature, depending on the period it is mentioned and the context it is mentioned in, software is also referred to as computer programs or computer

[1] Barron's Law Dictionary

software or the works. In this dissertation, all these terms are considered to mean the same thing.

Although copyright law is not uniform globally (intellectual property rights are territorial), there are common features that have been adopted as a result of international treaties and agreements. These provide copyright owners a number of rights relating to certain acts that be performed on the works protected by copyright:

- The right to copy the work and to issue copies
- The right to rent or lend the work
- The right to perform, show or play the work in public
- The right to broadcast the work
- The right to make a derivation of work and do any of the above in relation to such derivation

These are acts that are restricted by copyright, and anyone who does any of these acts without the permission or licence of the copyright owner, infringes copyright, barring any defensible exceptions.

In relation to software, the relevant rights restricted by copyright are mostly the right to copy, the right to issue copies (distributing) and the right to derive other works from the original work (adapting or modifying). However, copyright does not control how software is used, only how it is copied, modified or distributed. It is not the act of using the software but copying, distributing or adapting it that causes copyright to be infringed.

Copyright was traditionally used to protect artistic, dramatic and literary works. These are the expressions of the creativity and intellectual efforts of the authors.

Copyright was modified and adapted to be used for software, and was not even the recommended means of legal protection by the World Intellectual Property Organization (WIPO). In 1978, WIPO drafted a model law based on a *sui generis* approach that is neither copyright nor patent, but was later abandoned in preference for copyright.

However, copyright was never meant to protect concepts and ideas, which was the domain of patent protection. As a result, two important issues that challenged copyright as a means of legal software protection are: whether non-literal elements of software such as interfaces or program structures are copyrightable, and whether reverse engineering is permissible, neither of which were answerable or meaningful in previously existing copyright laws.

Copying software

Infringing copyright with literal copying

As with any other literary work, the copyright in software is infringed when a copy of it or a substantial part of it is made without the copyright owner's permission. One of the early cases of UK software infringement in which the software was copied was *Total Information Processing Systems Ltd v Daman Ltd*[2]. In this case the court decided that because a part of the software did not provide executable code, it was not a substantial part of the software and therefore copyright was not infringed. This decision is flawed as the significance of any part of the software should be determined by the functional importance of the software and not if it is executable or not.

[2] Total Information Processing Systems Ltd v Daman Ltd [1992] FSR 171

Fortunately this is not the currently accepted position in UK; in *IBCOS Computers Ltd v Barclays Mercantile Highland Finance Ltd* [3] the court determined that even though there might not be direct full copying of the software, there were enough significant portions of the code that was copied to have deemed copyright to be infringed.

Infringing copyright with non-literal copying

For software, copying is not restricted to literal copying. The non-literal copying of software can also be recognized as copyright infringement. For example, notes that are taken for the design of the software, or requirements specifications that are written to define the functions of the software can and do take on separate copyright on their own. Furthermore, if only literal copying is taken as infringement, copyright would be easily circumvented with the simple cosmetic alteration of variable names and function names. On the other hand, protecting non-literal elements of software takes a different light if we consider that copyright should not protect ideas and concepts and it gives too much monopolistic power to the copyright owners. In some legal literature, this is also known as the *idea/expression dichotomy*, and this is not unique to software.

Acknowledgement of the protection of expression, but not ideas, in is directly US copyright law whereas such explicit acknowledgement is not in the UK Copyright, Designs and Patents Act 1988[4]. The idea/expression dichotomy in the US originated in the United States Supreme Court case of *Baker v. Selden*[5]. Baker v. Selden established the principle that the exclusive reproduction rights in

---

[3] IBCOS Computers Ltd v Barclays Mercantile Highland Finance Ltd [1994] FSR 275

[4] See section on UK Copyright, Designs and Patents Act 1988 below

[5] Baker v Selden (1880) 101 US 99

copyright cannot prevent unprotected ideas or practical features of functional works from being re-used.

However, there may be occasions where it is not possible to separate the idea from the expression. This is true when the expression is the only way to express the idea. The *merger doctrine* then states that the expression and the idea 'merge' and the expression is not subject to copyright. This is because to grant copyright would be to grant a monopoly over the expression and that would be inconsistent with copyright laws.

The merger doctrine shows an interesting issue with open source software. As open source licensing depends on copyright to enforce its terms and conditions, if copyright does not exist with that particular software because of the merger doctrine, the terms and conditions lose effect. This does not affect closed source software as they can depend on contractual terms to safeguard their interests. However for open source software that is licensed with licences that prevent distribution if balked (for example the GPL) it effectively prevents the software from being distributed at all. However as mentioned it should be noted that while the merger doctrine is well-known in the US, it is not so evident in European or UK legislation.

An important case in non-literal copying is *Whelan Associates, Inc v Jaslow Dental Laboratory*[6]. In this case, the claimant claimed that the defendant's Dentcom computer program infringed the claimant's copyright on its Dentalab computer program, even though there was no issue of any literal code being copied. (Dentcom was written in Event Driven Language (EDL) while Dentalab was

[6] Whelan Associates, Inc v Jaslow Dental Laboratory [1987] FSR 1

written in BASIC). To reach a decision, the court used what is subsequently known as the *structure, sequence and organization (SSO) test* to determine the gap between idea and expression. The court decided that the purpose or function of the software is the idea, and that everything not necessary to that purpose or function is the expression of the idea. As a result, the court held that copyright in software protects the non-literal structure, sequence and organization of the software and therefore copying these non-literal components of the software is infringing:

> "We hold that ... copyright protection of computer programs may extend beyond the programs' literal code to their structure, sequence, and organization."

This case seems to contradict the merger doctrine by seemingly allowing a monopoly on the idea behind the computer program. However, the Jaslow v Whelan decision was subsequently criticized and rejected in the landmark *Computer Associates International v Altai*[7] case which proposed another test for software copying. In this case, an employee of the defendant who was previously an employee of the claimant took parts of the software which he wrote for his previous employer and re-used them in his new job. However, once this was discovered, the code was re-written and the new code no longer infringed. The claimant sued as the functionalities of both software components are the same. The court criticised the Whelan judgement and rejected the SSO test, using the *abstraction-filtration-comparison (AFC) test* instead to test for infringement.

[7] Computer Associates International v Altai (1992) 20 USPQ 2d 1641

This test first abstracts the software by breaking it down in several levels, through the overall purpose of the software followed by the individual modules that serve the purpose and then the source code implementing the modules. These granular pieces are then put through a filtration process that removes elements that cannot be protected such as elements that are dictated by efficiency (this is basically the merger doctrine), external factors such as industry standards and common programming practices, materials from the public domain and factual materials. Finally the elements that remained after the filtration process are compared with the alleged infringing materials to see if they are substantially similar. In the Computer Associates International v Altai case, the court decided that there was no infringement. As of current writing the Computer Associates International v Altai decision is the current authority on non-literal copying of software and the AFC test has been used in subsequent cases in US.

In UK the most important case to date involving non-literal copying of software is the *John Richardson Computers v Flanders*[8] case in which the court partially used the AFC test to come to a decision. In this case, the defendant was the ex-employee of the claimant company, in which he co-wrote the software in question. Eventually after leaving the claimant company the defendant re-wrote the software in another programming language for a different platform. Unfortunately the claimant company was also engaging in porting the software to the same platform and so sued for copyright infringement. The court adapted the AFC test in identifying non-literal parts of the infringed software and compared them for similarities.

[8] John Richardson Computers v Flanders [1992] FSR 497

The current decision is favourable to the open source software movement. There are a large number of open source software today that provides alternatives to existing closed source software, for example Open Office[9], an office productivity suite that is an open source alternative to the popular but closed source Microsoft Office productivity suite. If copyright is allowed to protect non-literal components of software such as user interface or program structure, it will stifle the growth and perhaps even kill off open source projects before they come to being.

Reverse engineering and making adaptations of software

As mentioned earlier, copyright was adapted to protect software as a form of literary work. For other common forms of literary works such as novels, plays or articles, the ideas and concepts behind them are apparent since the plots or messages are readily conveyed by reading them. However in software, this is not so evident. Ideas within software are not immediately apparent to consumers of the expression. The algorithms, principles and concepts behind software are usually not distributed to the end-users of the software. The source code for closed source software is almost never provided to the end-users without heavy confidentiality protection. This would make the monopoly granted by copyright on software too strong if not for certain exceptions that are specific to the legal protection of software.

Current copyright laws provide for exceptions to copyright infringement in software. The special permitted acts are reverse engineering of software, making back-up copies of software and making adaptation of software, provided the

[9] http://www.openoffice.org

users are lawful users. This provision in the UK Copyright, Designs and Patents Act is provided by the amendment in the 1992 Regulations under sections 50A – 50C. In the US, these exceptions are in the Copyright Act 1976 where the relevant section is in Chapter 1 section 117 (a) and Chapter 12 section 1201 (f) (also known as the Digital Millennium Copyright Act (DMCA)). These exceptions are also found in the European directive on the legal protection of computer programs.

The ability to reverse engineer and to make adaptations of existing software is important in open source, as open source projects often offer alternatives to existing commercial software with similar functionalities. At the same time, open source licences offer the actual source code without the need for reverse-engineering. This is probably something not anticipated within the copyright framework, and it will be interesting to know how the law will reconcile between these two in cases of conflicts. For example, the UK Copyright, Designs and Patents Act allows for the adaptation of the software for error correction (section 50(C) (2)). GPL on the other hand prohibits copying or distribution of derivative works unless it is also licensed under GPL. If a software developer develops a fix for a GPL licensed software and decide not to release it under GPL, the developer can use this as a possible defence.

Today, copyright is a stable and accepted means of legal protection for software, and the interoperability debate has largely been resolved as legislation in the copyright laws.

Singapore Copyright Act (Chapter 63)

The Singapore Copyright Act is essentially based on the Australian Copyright Act 1968, which was in turn based on the United Kingdom Copyright Act 1956. The Copyright Act was amended in 1999 to implement Singapore's obligations under the TRIPS[10] Agreement. The amendments were made on the recommendations of the Electronic Commerce Committee (ECC) on intellectual property rights. The ECC was set up in 1998 to enhance Singapore's copyright law to promote electronic commerce and encourage a knowledge-based economy.

The Act explicitly includes computer programs as a literary work (in section 7A part 1a) that is protected but does not attempt to define what a computer program is. There is also no clear definition between a compiled program and the source code.

In 2004, Singapore overhauled its intellectual property laws, including the copyright law. Most of these amendments implemented obligations undertaken by Singapore under the bilateral free trade agreement with the US signed in 2003. Many of these changes import some concepts from US IP law, including an extension of copyright term to 70 years and increasing the coverage of criminal offences. Copyright enforcement has been made stricter with this overhaul and a statutory damages system for copyright infringement was introduced. It should be noted that this brings the copyright law in Singapore closer to what is practised in US although the main substance of the copyright laws are still rooted in older English system.

[10] See TRIPS section below

In Singapore, copyright and other intellectual property laws are managed through the Intellectual Property Office of Singapore (IPOS)[11].

UK Copyright, Designs and Patents Act 1988

Copyright law originated in the United Kingdom from a concept of common law; the Statute of Anne 1709. It became statutory with the passing of the Copyright Act 1911. The Copyright, Designs and Patents Act 1988 is the current legislative source of copyright law in the UK. In the Act, computer programs subsist as a form of literary work. However, this has not always been so. The Copyright Act 1956 made no mention of computers or computer programs. The Copyright (Computer Software) Amendment Act 1985 first made it clear that computer programs were protected by copyright as literary works, and the latter Copyright (Computer Programs) Regulations 1992 included computer programs as part of the protected works under literary works.

Included as part of the amendments are specific exceptions to copyright infringement. The special permitted acts are for reverse-engineering (or de-compilation), making back-up copies and making adaptations of the computer programs. An interesting aspect of the changes made in the Copyright (Computer Programs) Regulations 1992 is the change to permit certain acts for computer programs, including reverse-engineering, making back-up copies and making adaptations of computer programs, which are the amendments made to follow the directive on the legal protection of computer programs.

The Act does not state expressly that ideas are not protected by copyright. This compares significantly with the US Copyright Act where it stipulates in section

[11] For more information refer to http://www.ipos.gov.sg

102: "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." UK law do not explicitly make the distinction between idea and expression, either in legislation or in case law. However, cases have been decided that produced the same results, for example, in *Page v. Wisden*[12], a cricket scoring sheet was held as not protected by copyright.

The Patent Office[13] is responsible for developing and carrying out UK policy on all aspects of intellectual property including copyright. The Intellectual Property & Innovation Directorate (IPID) is responsible for formulating and implementing all new UK legislation involving intellectual property. The UK is party to the Berne Convention and is also a member of WTO therefore adheres to the TRIPS agreement. The UK has also implemented the EU Copyright Directive in 2003 and complied with the WIPO Copyright Treaty.

US Copyright Act 1976

The power to enact United States copyright law is granted in the United States Constitution, which states: "… the Congress shall have power . . . to promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries."[14] The copyright tradition follows from the English common law tradition but the key difference is that copyright is written in the US constitution itself.

---

[12] Page v Wisden (1869) 20 LT 435

[13] Referenced from http:// www.patent.gov.uk

[14] United States Constitution Article 1 Section 8 clause 8

The US Congress first exercised its power to enact copyright legislation with the Copyright Act of 1790. The Act secured an author the exclusive right to publish and sell "maps, charts and books" for a term of 14 years, with the right of renewal for one additional 14 year term if the author was still alive. Today, copyright last for 70 years after the death of an author, or 75 to 95 years in the case of works of corporate authorship and works first published before January 1, 1978. The term of copyright is also a key difference from European and UK copyright law which protects copyrights for 50 years.

In the US, copyright law is administered by the United States Copyright Office[15], a part of the Library of Congress. The US became a Berne Convention signatory in 1988, and the treaty entered into force with respect to the US on March 1, 1989. The US is also a party to TRIPS, which itself requires compliance with Berne provisions, and is enforceable under the WTO dispute resolution process.

The US copyright law is a major influence for open source licences as most of the open source licences are originally written in US and assume US copyright law. This however becomes an issue that is explored in a later section.

Copyright in the international context

Berne Copyright Convention[16]

The Berne Convention for the Protection of Literary and Artistic Works, adopted at Berne in 1886, first established the recognition of copyrights between sovereign states. Prior to the adoption of the Berne Convention, copyright

[15] http://www.copyright.gov

[16] Referenced from the WIPO Berne Convention website at http://www.wipo.int/treaties/en/ip/berne/index.html

provided only national protection. The Berne Convention required that each signatory state to recognize the copyright of works created by citizens of other signatory states. Copyright under the Berne Convention is automatic, no registration or copyright notice is required. Additionally, signatories to The Berne Convention were prohibited from requiring any such registration on citizens of other signatory states.

The Berne Convention provided for a minimum term of copyright protection of the life of the author plus 50 years, but signatories were free to provide longer terms of copyright protection. The European Union extended copyright protection with the 1993 Directive on harmonising the term of copyright protection while the US followed with the Sonny Bono Copyright Term Extension Act of 1998.

The US initially refused to become a party to the Convention, as it required major changes in its copyright law especially to moral rights and the registration of copyright works. As a result, the Universal Copyright Convention (UCC) was adopted in 1952, to cater to its objections. In 1989, the US became a party to the Berne Convention. Since 1967, the Berne Convention has been administered by WIPO, the World Intellectual Property Organization.

TRIPS Agreement (Trade Related Aspects of Intellectual Property Rights)[17]

The Treaty on Trade Related Aspects of Intellectual Property Rights (TRIPS) was signed on 15 December 1993 as a constituting document of the World Trade Organization (WTO), and entered into force on 1 January 1995. TRIPS sets

[17] Referenced from the WTO TRIPS website at http://www.wto.org/english/tratop_e/trips_e/trips_e.htm

minimal rules for every member nation's national intellectual property law to prevent member nations from using intellectual property as a hidden trade barrier against other nations.

TRIPS is a compulsory requirement of WTO membership and any country that seeks to join WTO must change their national intellectual property law to comply with TRIPS. Unlike other international intellectual property agreements like the Berne Convention or the UCC, TRIPS has a powerful enforcement mechanism because states which do not adopt TRIPS-compliant intellectual property laws can be disciplined through the WTO's dispute settlement mechanism, which can authorize trade sanctions against non-compliant states.

Copyright in TRIPS were imported from the Berne Convention. Copyright terms must extend to at least 50 years after the death of the author and must be automatic, that is without the need for registration or any other forms of formalities. TRIPS has a national treatment principle in which member states are not allowed to offer any intellectual property benefits to local citizens which are not available to citizens of other TRIPS signatories. TRIPS also follow the most favoured nation principle in which citizens of all member states are treated equally. In addition, TRIPS has an important principle, that is, intellectual property protection should contribute to technical innovation and the transfer of technology. TRIPS mention explicitly that software is protected under literary works.

WIPO Copyright Treaty[18]

The WIPO Copyright Treaty, adopted by the World Intellectual Property Organization (WIPO) in 1996, provides additional protections for copyright. It ensures that software is protected as literary works (Article 4) and that the arrangement and selection of material in databases is protected (Article 5). It also provides authors of works with control over their rental and distribution (Articles 6-8) which they may not have under the Berne Convention alone. At the same time it prohibits circumvention of technological measures for the protection of works (Article 11) and unauthorised modification of rights management information contained in works (Article 12). The WIPO Copyright Treaty is implemented in United States law by the Digital Millennium Copyright Act [19](DMCA) and in Europe with the EU Copyright Directive[20].

---

[18] Referenced from WIPO Copyright Treaty online at http://www.wipo.int/treaties/en/ip/wct

[19] Referenced from the US Copyright Office at http://www.copyright.gov/legislation/dmca.pdf

[20] Referenced from the Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society. Full text of this directive can be found at European Union Online web site at http://europa.eu.int/information_society/eeurope/2005/all_about/digital_rights_man/doc/directive_copyright_en.pdf

THE OPEN SOURCE SOFTWARE MOVEMENT

The open source software movement began in the programmer culture of US computer science laboratories in the 1960's and 1970's. The community of programmers was small and close-knit and code passed back and forth between the members of the community easily. Keeping code to yourself was considered unfriendly – after all, you benefited from the work of your friends, you should return the favour.

Richard Stallman, a graduate student at the Massachusetts Institute of Technology Artificial Intelligence lab, was part of this community. Soon however, as computers grew in importance, the community grew larger and commercial remunerations tempted programmers to sell their software to companies who in turn guarded their intellectual property jealously. Determined to return to community of cooperating programmers he was used to, Stallman decided to devote himself to creating free software. In 1984, Stallman resigned from MIT so that the university would have no claims on the software he created, and started a Unix-compatible operating system project called GNU (which stands for Gnu's Not Unix). In 1985, Stallman created the Free Software Foundation (FSF), a tax exempt charity, to support his work and that of his collaborators. To ensure that his code would always be freely modifiable and distributable, he created the GNU General Public License (GPL). The FSF proposed a radical concept of 'free software', with their slogan of 'free' as in 'free speech' not as in 'free beer'.

In 1992, a second year graduate student at the University of Helsinki named Linus Torvalds wrote a Unix-like kernel (which is the core of an operating system) called Linux. Eventually, Linux became the de facto kernel for the GNU operating system, licensed with GPL. Today Linux is the best known and most

visible software that is part of the free and open source software movement, spawning a multi-billion industry for alternative operating systems.

In 1997, Eric Raymond published a landmark essay entitled *The Cathedral and the Bazaar*[21] explaining why open source licensing of software will result in higher quality, less expensive software. Later, a coalition of individuals, led by Eric Raymond, Bruce Perens, and Tim O'Reilly, formed the Open Source Initiative (OSI) to promote the pragmatic benefits to the business community, and to certify free/open source licenses that meet the Open Source Definition. The Open Source Definition defines the general requirements for a licence to be considered open source. This was the start of the open source software movement.

The emergence of the OSI and the more pragmatic way of looking at open source software did not go very well with the FSF. Although not confrontationally opposing, the FSF has been known to disagree publicly on the principles and philosophy behind the OSI. The FSF believes that all software should always be accompanied by its source code, and the user has the right to modify and extend that source code. The OSI people share that goal, but do not define this as a moral right, but instead focus on the pragmatic benefits of source sharing. Also, while the FSF defines the GNU GPL and its derivatives, the OSI attempts to capture the commonalities across a range of different licenses. All free software licenses are open source, but not all open source licenses are free software.

---

[21] This was later expanded into a book with the same title, see Raymond 1999

Although the FSF and the OSI has famously debated over their differences[22] in philosophy in promoting the free and open source software movements, the desired goals are the same and both parties are not mutually exclusive. In this dissertation, the term 'open source' is used to mean both free and open source movements.

Open Source Licences

There are a large number of open source licences in the market today, derived from various historical background and usages. The Open Source Initiative has accepted variety of open source licences as certified open source licences[23], and the Free Software Foundation has also defined a number of free software licences[24]. However the objectives of all the licences are very similar except for some minor differences, mainly to do with the issue of reciprocity.

The sections below are short descriptive analyses of the few most commonly used open source software licences.

GNU Licences

The GNU Project[25] was launched in 1984 to develop a free, complete UNIX style operating system, the GNU system (GNU is a recursive acronym for GNU's Not UNIX). Variants of the GNU operating system, using the Linux kernel, are now widely used; though these systems are often referred to as Linux. The Free Software Foundation (FSF) manages the GNU project.

---

[22] See Klang (2005)

[23] See http://www.opensource.org/licenses for a full list of open source licences

[24] See http://www.fsf.org/licensing/licenses/index_html for a fill list of free licences

[25] Referenced from http://www.gnu.org

The first version of the GNU General Public Licence appeared in 1988, and it has continued to evolve into its current form, version 2, released in 1991. The GPL version 2 is the most popular open source licence to date, with close to 70% of the software on Sourceforge, the largest repository of open source software in the world, licensed using it.

The GPL begins with a pre-amble that, though not a part of the licence, describes the spirit of what the licence wishes to achieve. Firstly, the licence allows the software to be distributed and modified without additional permission from the licensor. Secondly, the licence ensures that the licensees are aware that the software is distributed without warranty. Thirdly, the licence frees the software from restrictive patents.

One of the key concepts in the GPL is the concept of *copyleft*, a concept that places a reciprocity obligation on software developers to license works that are derived from GPL licensed software under GPL as well. The FSF defines copyleft as "a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well." The rationale behind copyleft, according to the FSF, is to encourage collaboration and increase the freedom of the software. Not surprisingly, this reciprocity obligation is also one of the most criticised features of the GPL.

The fear for some of the closed source software development companies, especially the larger ones, is that some of the developers (which could sometimes range in thousands, from different locations around the globe) could accidentally include GPL licensed code and inadvertently cause the closed source software to

be forcibly made GPL. This obligation has often been dubbed by the media as 'viral'[26].

However, the fears are unfounded as the GPL has no special powers beyond what is given in the copyright laws. Software developers who do not wish to be restricted by this licensing clause can refrain from deriving from GPL code and if GPL code is accidentally included, the act of removing the offending code will rectify the situation quickly. After all, no matter if the code is GPL-licensed or not, if licensed code is included in any software without proper authorization the same copyright infringement penalties apply. The label of being 'viral' is descriptively wrong as a license cannot be 'infectious' the way a virus (organic or otherwise) is. It takes the conscious effort of deriving source code from the original GPL code, deliberately copying and distributing it, and wilfully including GPL code with non-infringing code that causes copyright infringement.

The GPL, though often criticized, is a well written legal document with clear upfront concepts. Its enforceability is something that has been often debated and questioned as well. Although Eben Moglen, the general counsel for the FSF explains that the GPL is not a contract[27], the contractual limitations of the GPL have also often been subjected to scrutiny.

The definition of the scope of the GPL clearly limits the licence to copying, distribution and modification activities only. At the same time, only software that

---

[26] Typical commentaries on GPL's 'viral' properties are as in this article from Microsoft cautioning the public on the usage of GPL
http://www.microsoft.com/resources/sharedsource/Articles/LicensingOverview.mspx

[27] In Moglen (2001) he explains "Licenses are not contracts: the work's user is obliged to remain within the bounds of the license not because she voluntarily promised, but because she doesn't have any right to act at all except as the license permits."

is integral to or derived from software that is GPL-licensed is affected. The GPL creates a relationship between the licensor and each of the licensees regardless of the number of distribution layers that have gone through between them. It also bars licensees from imposing additional restrictions on the recipient's rights to the software.

In addition, the GPL prevents any dilution of its effectiveness. If in any case where the licensee is not able to adhere to the terms of the GPL, either by contract or by court enforcement or any other constraints, the licensee loses the right to perform the activities. For example if the licensee is not able to distribute the software without charging for the licence, then the licensee loses the right to distribute the software, but he retains other rights as long as he conforms to the conditions. This effectively prevents any software licensed with GPL to be combined in any way with another piece of software not licensed with GPL or not compatible with GPL. This has serious repercussions, as GPL in-compatible licences are not just closed source licences. FSF also considers some open source licences like the Apache licenses (1.1 and 2.0) as well as the MPL in-compatible with GPL.

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

The licensor is explicitly allowed to modify the GPL such that certain jurisdictions that restrict the GPL can be excluded from GPL allowed activities. For example, if some countries disallow licensing and distributing some software in GPL due to pre-existing patents, then the licensor is allowed to limit the distribution of the software in those jurisdictions.

Another popular open source licence from the GNU project is the Lesser General Public Licence (LGPL). Previously known as the Library General Public Licence, this licence was created especially to overcome a technical problem with GPL when used with certain software libraries that requires the software to be linked to be used. LGPL specifically allows LGPL-licensed software libraries to be linked with non-GPL licensed software, including closed sourced software. However especially to be noted is that although the GNU Project provides the LGPL, it encourages the GPL to be used over LGPL. The LGPL is the second most popular open source licence, with 11% of all software hosted in Sourceforge being licensed under the LGPL.

Combined, the two GNU licences provide licensing to approximately 80% of all open source software in the world. It is not surprising then to know that most of the criticism focused on open source software and open source licensing centres around these two licences, and most debates on open source take GPL as the prime example of an open source licence.

Open Source Definition

The Open Source Definition (OSD) defines the general requirements in order for a licence to be considered as open source. The OSD is managed by the Open Source Initiative[28] (OSI), an organization formed in 1998 to promote the more practical usage of open source software. The OSI also certifies license to indicate if they fall under the OSD. Effectively the OSD is a guideline for licences that wishes to be open source.

[28] Referenced from http://www.opensource.org

The OSD's main concepts for open source are free redistribution, access to source code and open modification of the source code. In addition, the OSD requires that the licences do not discriminate against persons, groups of persons or any fields of endeavour and be technology-neutral. The OSD also tries to close certain loopholes within existing open source licences.

The OSD is consistent with the older free software definition described by the FSF but is a looser and more business-friendly definition. It does not attempt as GPL does, to force derived works to follow the same licensing as the original work, but does not disallow that either. From a larger perspective the GPL is OSD certified but OSD certified licences are not necessarily GPL compatible.

The MIT[29] and BSD[30] Licences

The MIT and BSD Licences are the two earliest open source software licenses. The MIT Licence is a simple licence that basically grants all of the rights of a copyright holder including the exclusive right to commercially exploit and create derivatives from the software. The only two conditions imposed are that the copyright and permission notices must be included in the copies of the software and a general disclaimer of warranty.

The BSD Licence is only slightly more restrictive. Originally it carried a provision that the University of California, Lawrence Berkeley Laboratory must be acknowledged in all advertising and use of the software, but was this was removed later on. The only other restriction that is different from the MIT Licence is that the name of the organization that created the software or it

[29] Massachusetts Institute of Technology

[30] Berkeley Software Design (BSD is Unix variant first developed from the University of California, Berkeley)

contributors cannot be used to endorse or promote the software without prior written permission.

The MIT and BSD licences are both OSI-certified licences and GPL-compatible licences, although the original BSD licence is not GPL-compatible. The MIT and BSD licences are one of the most popular open source licences partly because of they have been around for a long time.

The Apache Licences

The Apache License has two major versions, the older 1.1 and the newer 2.0 released in 2004. The older Apache License version 1.1 is very similar to the BSD License, but includes a requirement for the acknowledgement of the creator's contributions of the software. The Apache Licence version 2.0 is a more complex and comprehensive licence. It includes provisions for patent rights granted by the licence and the use of other licences for derivative software based on the original software. The Apache Licence version 2.0 also explicitly defines 'Contributions' that are special modifications of the software provided to the licensor of the software for its inclusion into the original software. If accepted, the modifications will become part of the original software and will fall under the same licence.

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

The Artistic Licence

The original Artistic License was written by Larry Wall, the creator of Perl for use by Perl, a popular Internet programming language. It was designed to allow Larry Wall and his collaborators to maintain control over the Perl project while encouraging participation in the project and innovation outside the project. It is

often heavily criticised for being ambiguous, self-contradictory and virtually impossible to interpret. The FSF, who maintains a list of free software licences do not even acknowledge the Artistic Licence as free software licence although a later version of the Artistic Licence has been accepted. Although the OSI does acknowledge the original Artistic Licence, the Open Source Definition actually defines a point to close up a loophole found in the Artistic Licence. One problem with the Artistic Licence is that although it prohibits sale of the software it also allows an aggregate distribution of the Artistic Licensed software with another piece of software. Interpreted literally, someone can defeat the licence by merely including a trivial piece of software together with the licensed software.

Due to the popularity of the Perl programming language, the Artistic Licence is one of the more popular open source licences around.

The Mozilla Public Licence[31]

The Mozilla Public Licence (MPL) was drafted by Mitchell Baker, a lawyer who originally worked for Netscape Communications (and later joined and became president of the Mozilla Foundation). Established in July, 2003, with start-up support from America Online's Netscape division, the Mozilla Foundation is a non-profit organization founded to manage the Mozilla project, which in turn was founded after Netscape released its Communicator browser as open source.

The MPL was initially created along with the Netscape Public Licence (NPL) to address the issues relating to the decision that Netscape Communications took to release the binary and source code of the Netscape Communicator web browser for free. There are two versions – MPL 1.0 and MPL 1.1.

---

[31] Referenced from http://www.mozilla.org

Originally developed for software released under the Mozilla Foundation, it has since been adapted by others as a license for their software, most notably Sun Microsystems, as the Common Development and Distribution License for OpenSolaris (the open source version of Solaris 10, a popular UNIX-variant server operating system).

MPL can be loosely regarded as a hybrid of ideas between the GPL and the MIT/BSD licences. The FSF regards the MPL as a weak copyleft as it allows MPL-licensed code to be combined with code licensed under another license. MPL is not a GPL-compatible licence but it is OSI certified.

A distinguishing difference with the other licences analysed is that the MPL divides a software work into an Open Source part (called "Covered Code") and anything a contributor adds. This arrangement allows any developers to add his own files and distribute them with the covered code, provided he does not modify the covered code. However if he does modify the covered code, he must distribute the modified code under MPL. The license also shows its link to commercial software licenses with the standard licensing language covering such topics as liability and arbitration. The MPL is considered one of the better drafted open source licences and is used in many open source projects including the popular Firefox browser.

.

Open source licenses matrix

The matrix below shows a matrix of the different licences described above, showing the comparison of rights granted by the different licences as well as the size of the licence document itself. Public domain and closed source licensing are also compared.

Freedoms or Restrictions	**Public Domain** [32]	**MIT/ BSD**	**Apache 1.1**	**Apache 2.0**	**Artistic**	**MPL 1.1**	**GPL**	**LGPL**	**Closed Source**
Has copyright owner	✗	✓	✓	✓	✓	✓	✓	✓	✓
Copyright acknowledgement	✗	✓	✓	✓	✓	✓	✓	✓	✓
Freely copy and use as-is.	✓	✓	✓	✓	✓	✓	✓	✓	✗
Distribute modified versions with same licence	✓	✓	✓	✓	✓	✓	✓	✓	✗
Distribute modified versions under different licence	✓	✓	✓	✓	✓	✗	✗	✗	✗
Link with code under different licence	✓	✓	✓	✓	✓	✓	✗	✓	✗
Must include source code in the distribution [33]	✗	✓	✓	✓	✓	✓	✓	✓	✗
Grants licensee patent rights	✗	✗	✗	✓	✗	✗ [34]	✓	✓	✗
Disclaimer of warranty/limitation of liability	✗	✓	✓	✓	✓	✓	✓	✓	✓
Non-endorsement provision	✗	✗	✓	✓	✗	✓	✗	✗	NA
Reciprocity obligations for derivatives (copyleft)	✗	✗	✗	✗	✗	✓	✓	✓	✗
Number of words in licence document (complexity of licence)	NA	167/ 222	294	1,581	771	3,666	2,956[35]	4,020[36]	Varied

Table 1 - Free and open source licence rights matrix

[32] Public domain software does not require any licences

[33] Including in the distribution can also mean allowing the source code to be available for download from the same location as the binaries

[34] MPL only grants the patent rights to the original (covered) code

[35] Size of this document includes the preamble, which describes the philosophy behind the licence

[36] As above

Open source licenses popularity

Sourceforge[37] is the largest repository of open source software in the world, with more than 110,000 registered projects as of July 2005. From an investigation into Sourceforge, there were 65,362 OSI approved open source projects registered in Sourceforge in July 2005 (the rest are either proprietary licensed software that has not been approved by OSI, or projects under public domain).

The General Public Licence is used by the largest number of projects (69%) followed by the Lesser General Public Licence (11%). The combined numbers represent 80% of the projects registered in Sourceforge. Although Sourceforge is not the only repository of open source software, it holds the most significant number of projects. However it should be cautioned that many major open source projects are not hosted by Sourceforge (for example, the Apache Foundation hosts their own projects licensed under the Apache Licences).

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

Table 2 - Free and open source licence popularity in Sourceforge

[37] http://www.sourceforge.net

LEGAL ENFORCEABILITY OF OPEN SOURCE LICENSES

Open source licences – copyright licence or contract?

The idea of the validity and legal enforceability of open source licences often argue around 2 central ideas on how software can be legally protected. The first relates to intellectual property, that is, the inherent copyright that the software acquires as soon as it is written. The second concerns the legal contract that is drawn up between the software copyright owner and the recipient of the rights. These can be any of the rights granted to the owner and is acquired by the recipient including the right to modify or distribute the software. The legal contract defines the scope and rights that are traded in exchange for the consideration provided by the recipient.

While no-one debates the rights granted by copyright law, that a legal contract exists at all in open source licences is sometimes challenged and argued for and against the legal enforceability of the licences.

Copyright in a work is infringed when a person performs an act restricted by copyright, unless the licence of the copyright owner is obtained. A licence can be defined as "an agreement between the owner of the copyright (the licensor) and another person (the licensee) whereby that person is permitted to do certain acts in connection with the work involved that would otherwise infringe the copyright in the work."[38] However, a licence is not necessarily contractual as there are a number of conditions that must be met before it is considered a contract.

---

[38] Bainbridge (2002) Chapter 4, page 88

A contract is "an agreement giving rise to obligations which are enforced or recognized at law"[39] and the heart of a contract is an agreement. Essentially, the parties to a contract must have a meeting of minds, and there are four key elements to forming a valid contract. The first is an offer, that is, the willingness of a party to perform a promise or a task. In the case of a licence as a contract, the offer is to licence the software to the licensee. The second is the existence of an acceptance, which is an unconditional assent to the terms of the offer. The third is sufficient consideration. Consideration is the compensation for the promise, in this case what the licensee provides to the licensor in return for the licence. Lastly there must be an intention to create legal relations.

Generally, there is no doubt an offer exists by copyright owners to licence their software. The acceptance of the licence however is debatable. Comparison is often made with click-wrap and shrink-wrap closed source licences. In the case of shrink-wrap licences, the buyer purchases software that is shrink-wrapped in a box. The licence is only available for the buyer after he pays for the software and opens it. It has been argued that since the buyer has not seen the licence after the sale is complete, the licence is not part of the contract. In the case of click-wrap licences, the licence is again shown to the buyer after he purchases the software and is usually during the installation of the software. However despite these conditions, click-wrap and shrink-wrap licences have been accepted as valid licences in the US through the *ProCD v Zeidenberg*[40] landmark case and the UCITA[41] has confirmed these licences in the US legislation.

---

[39] Treitel GH, Law of Contract 11th ed

[40] ProCD, Inc. v. Zeidenberg (1996) F.3d 1447

[41] Uniform Computer Information Transactions Act

In UK, the validity of shrink-wrap licences was first upheld in the Scottish case of *Beta Computers (Europe) Ltd v Adobe Systems (Europe)*[42].

Although consideration is often associated with economic compensation, the promise of doing something in return can be argued as sufficient consideration in most cases. In all open source licences there is at least some promise that the licensee needs to keep, whether it is to include the licence in its re-distribution or to license derivatives using the same licence. The issue however is that of adequacy of consideration. This means that each side must promise to give or do something for the other as part of the contract. Without consideration, that is, if the software and source code is given freely as a gift, there is no contract.

There is usually no monetary consideration given in exchange for open source licensed source code, although open source licences do not prevent anyone from charging a fee for the physical transfer of the source code. This is a point that is sometimes mentioned by persons not familiar with open source concepts, because 'free' is often confused as equivalent as 'without cost'. However, if money is not involved, what are the considerations exchanged in order to form a contractual relationship between the open source software developer and the user? The answer is that the developer grants the licence to his program in exchange for the user's promise to follow the terms and conditions stated in the licence. However, this begs another question – is this consideration adequate?

Consideration does not necessarily need to be money. However, the consideration needs to be sufficient, that is, it needs to have some real value. Lush J. in *Currie v Misa*[43] referred to consideration as consisting of a detriment to

---

[42] Beta Computers (Europe) Limited v. Adobe Systems (Europe) Limited. FSR (1996) 367

[43] Currie V Misa (1875) LR 10 EX 153

the promisee or a benefit to the promisor. From this point of view, the promise to follow the terms and conditions in open source licences can be argued as sufficient consideration.

Finally, the tricky rule of intention to create legal relations can be debatable. In the GNU GPL, Eben Moglen, the general counsel for the FSF declared that "Licenses are not contracts"[44]. As Moglen clearly states, the GPL is not considered a contract, but a copyright licence. Following the rule of intention to create legal relations it would seem that the GPL cannot be considered a legal contract. However there are no such statements in other open source licences and it is arguable if such statements can be considered as proof of non-intention at all.

The issue of privity of contract is another issue that has been brought up by a few writers[45], especially when discussing the GPL, which is taken as the most representative of all the open source licences. Privity of contract is the relationship that subsists between the two contracting parties. Essentially, no one but one of the parties can go to court and enforce the contract even if the contract was to operate to a third party's benefit. In the case of secondary distribution where an original recipient of the software re-distributes the software, the issue is if the terms and conditions of the GPL still valid for the secondary recipient as there is no privity of contract between the original licensor and the secondary recipient.

On analysis, GPL is a non-exclusive, transferable licence (a licence that allows sub-licensing). The distribution of the GPL code is in fact a transfer within the

---

[44] See Moglen (2001)

[45] See Merges (1997), McGowan (2001) and Ravicher (2000)

terms of the licence. Distribution is allowed and governed with terms and conditions within the licence itself. In paragraph 1, the GPL specifies that the code can be distributed provided the copyright notice and the disclaimer of warranty are conspicuously published. This effectively means that the distribution of the code establishes a relationship between the secondary recipient and the original licensor. Paragraph 6 of the GPL confirms this: "Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions."

Privity of contract also applies when the recipient creates derivative works from the original GPL code. However, since the GPL requires derivative works to be distributed under the GPL the relationship is between the creator of the derivative work and the secondary recipient.

It would seem that a clear-cut affirmation if open source licences are copyright licences or legal contracts is yet to be determined. Although both legal contracts and copyright licences have different enforcements, one which is through covered by the contract itself, and the other is through copyright legislation, both enforces the same terms and conditions on the licensees. However, one noticeable difference is that without a legal contract, licensors can revoke their licences at any point in time, subject to equitable rules. This has some serious repercussions if the software is already well known in the market as the licensor is not obliged to continually provide the software under the same licence.

However, the initial creator of the code can only terminate the open source licence rights he has granted but not in the derivative works. The creator of the derivative works can continue to distribute the derivative works own his own

code but not the original code, as is clearly seen from *Steward v Abend*[46], which states that "the aspects of a derivative work added by the derivative author are that author's property, but the element drawn from the pre-existing work remains on grant from the owner of the pre-existing work.". In the section 103 (b) of the US Copyright Act, it is also stated that

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

Also, in many open source software projects the final software incorporates code from more than one creator therefore unilateral termination of rights seems unlikely as there are little to no benefits to any single one creator to terminate the rights. On the other hand projects with a single creator or a small number of contributors have been known to form commercial enterprises to use the software for commercial licensing. However, in these cases, the software is usually dual-licensed instead.

Another deterrent for creators to retract or terminate the rights is usually the pressure from the open source community itself. As with any companies, good will from the community is important for continual business, especially technology companies. Technology companies depend on the community to supply the credibility, marketing and resources to fuel the running of the

[46] Stewart **v**. Abend, (1990) 495 US 207

business. In the recent case of *SCO Group v IBM*, the fall-out from the technology community that supported the software industry has caused tremendous damage to the SCO Group. The SCO Group's stocks slid down from a high of more than US$20 in the aftermath of the suit, to the current price of less than US$5.

On the other hand, closed source licences are almost always written as a legal contract between the licensor and the licensee, and additional terms and conditions are almost always added into the contract. These terms are often additional safeguards that define the boundaries and scope of the relationship between the licensor and the licensee. In some instances, the legal contract itself tries to restrict copyright laws in certain areas. For example, most closed source licences explicitly disallow reverse engineering of software although copyright law equally explicitly allows that.

From the analysis it is not clear if all open source licences can be considered valid contracts but similar closed source licences have been accepted as valid contracts albeit controversially. Interestingly if a contract does not exist for open source licences, sometimes the copyright laws of certain countries impose a harsher criminal offence on copyright infringers[47]; therefore it would seem that it is to the benefit of the licensee not to use this as a defence against enforceability of open source software licences.

Copyright law in open source licences

The act of creating useful and non-trivial software, and later marketing and distributing it to a larger audience is often a capital-intensive activity. In the early

[47] In the Singapore Copyright Act Section 136 mentions "… shall be guilty of an offence and shall be liable on conviction to a fine not exceeding $10,000 for the article or for each article in respect of which the offence was committed or $100,000, whichever is the lower, or to imprisonment for a term not exceeding 5 years or to both"

days of software development, software is tightly integrated and tied to specific hardware. The source code and the software itself is often given freely to anyone interested enough to use or modify it. In most cases, the hardware itself is large, expensive and is usually owned by large corporations, research facilities or universities. The marketing and sale of software as a separate industry is non-existent and software is distributed as part of the sale of the computer hardware. Legal property protection of software was non-existent.

As technology advanced and hardware became more widely used especially with the advent of personal computers, software became eventually more de-coupled from the hardware. As a result, the marketing and sale of computer software became firstly a separate business activity and eventually became a distinct and even larger industry than computer hardware. With the increase of business activity it is inevitable that software is treated as property that is bought and sold. Legal property protection for software was something new, and copyright was eventually used for the protection of software. However, copyright does not fit entirely – it was created to protect creative works that are static and software is more than that. In addition, the problems of copyright protecting more than the literal representations of software became debatable.

Copyright was eventually chosen as the main legal vehicle for software protection, although patents are currently becoming the new battleground for further legal protection. However, because the current ruling for interoperability removed copyright protection for interfaces and structure as well as for reverse engineering, closed source software licences often tries to compensate by adding such rules in the licence itself as part of a contractual agreement. For example, in the end-user licensing agreement distributed along with Adobe Premiere, a popular video editing application from Adobe Systems, there is such a clause:

> "You also agree not to reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software except to the extent you may be expressly permitted to decompile under applicable law, it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Adobe to provide the information necessary to achieve such operability and Adobe has not made such information available. Adobe has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Adobe or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software."

Such a clause obviously extends the reach of copyright using a contractual agreement. There is even a clause at the beginning of the agreement:

> "YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. IF YOU DO NOT AGREE, DO NOT USE THIS SOFTWARE."

Most software developers (companies) resort to such licensing agreements to add additional weight of a legal contractual agreement to the protection of their software.

Copyright for open source software is even trickier. Although the software itself is not different but the intention of the copyright owner is to essentially revert

back to the original days of freely shareable software. However, the environment today is significantly different from the past and the copyright owner can no longer just release the software into public domain. This is because public domain software can be easily subverted into closed source software if another party takes the code, modifies it and re-licenses it.

To achieve the intentions of the copyright owner, the software needs to be licensed to prevent this, and open source software licences has this as a basic objective. Open source software copyright then involves the liberalization of the following copyright rights:

- Copying the software
- Distributing the software
- Distributing the source code
- Modifying the source code

All of these activities are prohibited in most closed source software, as the closed source software licence either explicitly prohibits it or do not mention it, therefore allowing copyright to protect it by default. If there is no specific licensing agreement in closed source software, the copyright owner owns all the rights to the software, and no-one else has any rights except those mentioned in copyright law. In contrast, open source licences gives these rights away to the licensees in exchange for relatively minor promises.

Enforcing open source licences therefore lies strongly in copyright protection. The existence of copyright laws allows for the protection of software as property and this applies to both closed source software as well as open source software. Issues relating to copyright that is seemingly ambiguous for open source licensing seem a moot point as copyright applies uniformly for any software regardless of the licensing that it uses. Effectively if copyright cannot be enforced on open

source software, it cannot be enforced on closed source software and if copyright can be enforced on closed source software there is no reason why it cannot be enforced on open source software.

Cross-jurisdictional issues in open source licences

A major issue with copyright is the issue with the applicability of the law in different countries. Copyright laws are territorial and generally do not cross borders. Although treaties such as the Berne Convention, TRIPS and other international agreements provide guidelines and directives for member signatories to adhere to, copyright is a local law that can behave differently in different jurisdictions.

Closed source software often overcomes this by mentioning explicitly how the licence will apply in different countries, and often has a general rule to define its applicability. For example, in Microsoft's end-user licensing agreement, the following clause explains how applicable laws may direct the licence:

> "If you acquired this Software in the United States, this EULA is governed by the laws of the State of Washington. If you acquired this Software in Canada, unless expressly prohibited by local law, this EULA is governed by the laws in force in the Province of Ontario, Canada; and, in respect of any dispute which may arise hereunder, you consent to the jurisdiction of the federal and provincial courts sitting in Toronto, Ontario. If you acquired this Software in the European Union, Iceland, Norway, or Switzerland, then local law applies. If you acquired this Software in any other country, then local law may apply."

Open source software however generally does not have such clauses. In addition, open source software has other issues relating to cross-jurisdiction applicability of the licence.

A first problem is with the language of the licence itself. Most open source licences are written in English and in fact assumes certain facts that are only applicable in US laws (where most of the open source licences are written). However in many countries, there are laws that mandate the use of the national language for legal documents including licences and contracts. For example, under the German Civil Code section 205 paragraph 2, there is a provision for consumer contracts to be in German in order to be valid. In France, there are laws relating the mandatory usage of French in the description of the scope and conditions of a warranty of goods, products and services. It is therefore conceivable that a software licence written in English such as most of the open source licences, is not legally binding in France or other countries that have similar restrictions. Assuming the software licence is a valid contract in the first place, it can be held not binding under particular circumstances because it is not written in the correct language.

Another problem arises in the legal background which copyright law is derived from. In countries that derive laws from the English legal system, copyright arose from the economic rights of copywriters and publishers while most continental European countries (for example Germany and France) derive copyright from the concept of *droit d'auteur*, which focuses on the moral rights of the original author. "Author's right" – *droit d'auteur* in French – is founded on the idea that a work of creation is intimately linked with its creator like a child from his father. The "copyright" concept on the other hand stems from the common law tradition stating that authors hold a property or economic right to their creations that can be traded on the basis of economic principles. Although the *droit d'auteur*

principle has been included in copyright concepts in common law countries as moral rights, the most important being attribution and reputation, the concepts are not exactly the same.

The sticking point is that in the *droit d'auteur* system, the author cannot waiver his moral rights without knowing the changes done to the source code, as it will affect the reputation of the author. Without taking care of such a provision, open source licences stand a serious chance in being repudiated when it comes to the modification and distribution of derivative works. This is particularly serious in GPL, which prevents the distribution of the software altogether if the issue of source code modification and the moral rights of the author is not correctly dealt with.

A third problem arises from the question of warranties and disclaimers. In certain countries, especially continental European countries, general disclaimers are not valid in a contract due to provisions for unfair terms in contracts. As a result, the disclaimers which are a part of all open source licences can be made non-binding. This has strong implication on the entire open source software movement, as one of the basic tenets of open source is the clear disclaimer of liability and warranty, as no software developer will be willing to be liable for voluntary and cost-free software. GPL is again implicated strongly as it explicitly mentions in paragraph 7 that the distribution rights will be cancelled if the conditions in the GPL are not met.

A fourth problem is in the interpretation of derivative works across different jurisdiction. In the US, derivative works are defined in the Copyright Act under section 101, which states that a derivative work is "a work based upon one or more pre-existing works." In the Singapore Copyright Act, adaptation is defined in section 7 (1) as "a version of the work (whether or not in the language, code or

notation in which the work was originally expressed) not being a reproduction of the work". An analysis of both definitions can bring two different interpretations of the definitions in the different countries. The Singapore definition is broader in which any modification in the computer program, no matter how little (which will make it not a reproduction of the work) will be considered an adaptation while in US the interpretation of minor usage of the code might not be considered as a derivative work. For example, a small piece of code taken from an open-source licensed software might not be considered as derivative work in US, but can be considered as an adaptation in Singapore.

It seems clear that most open source licences are not ready for cross-jurisdictional applicability outside of the US or other common law countries and they stand a possible chance of being set aside as non-binding for particular clauses mentioned in the licence. However it is interesting to note that the Munich Court[48] of Appeals in Germany has ruled that the GPL is generally valid although it is only legally binding in its English version and only parts of it (the exclusion of liability and warranty as explained earlier will not be valid in Germany).

CONCLUSION

Open source or the concepts behind it are not new, in fact, software was originally entirely shareable among the close community of software developers. Copyright did not apply to software in those early days of software development and legal protection of software became necessary only with the gradual commercialization of software as a separate intellectual commodity. In this dissertation, we went through how copyright was modified to be used for software protection and the problems that ensued as a result of this development.

[48] Harald Welte v. Sitecom, District Court of Munich, (2004) 21 O 6123/04. Also see Höppner 2004

We also discussed what it means for software to be open source licensed and analysed some of the most popular and common open source licences today.

With the background information and the analysis, we went on to investigate if open source licences are enforceable in a court case and discussed the various issues that can possibly influence the outcome. However because there has not been a court case that deals fully with the issues with open source licensing, the points discussed are nothing more than anticipation based on interpretation of current legislation and related cases on legal software protection.

As this investigation shows, there are numerous points that weaken the enforceability of open source licences, especially those that debated the validity of an open source licence as a legally binding contract and points that discussed on cross-border issues with open source licences. The interesting point to note is although the open source licence might potentially be set aside as a non-legal biding contractual agreement as a result of its validity as a contract, copyright still provides significant weight in ensuring its enforceability. Unfortunately the cross-border issues that are inherent in most open source licences today might prove to be a stumbling block for enforcement outside of the US. However there is no conclusive evidence that any open source licence would not be enforceable in any court today; in fact GPL was considered valid in a Munich court. How other cases turn out in the future remains an interesting development to be followed on.

The open source software movement is an exciting alternative development in the fast-paced software industry, one that promises a change in the way we look at how software is developed, distributed and sold. As its major premise lies within the legal protection of software licensed with this unique licensing

mechanism, it is important that the evolution of the legal processes that governs it matches the ongoing developments in order to facilitate its progress.

REFERENCES

- David Bainbridge, *Intellectual Property* 5th Edition (2002)
- Patrick K. Bobko, *Linux and General Public Licenses: Can Copyright Keep Open Source Software Free?*, 28 AIPLA QJ 81 (2000)
- Patrick K. Bobko, *Open-Source Software and the Demise of Copyright,* 1 Rutgers Computer & Tech. LJ 51 (2001)
- Costello, S., *Settlement nears in open source GPL suit*, NetworkWorld Fusion News (2002) at <http://www.networkworld.com/news/2002/0305settle gpl.html>
- Hahn, R.W., *Government Policy toward Open Source Software*, Aei-Brookings Joint Center for Regulatory Studies
- Ira V. Heffan, *Copyleft: Licensing Collaborative Works in The Digital Age*, 49 Stan. L. Rev. 1487 (July. 1997)
- Höppner, J. P., *The GPL prevails: An analysis of the first-ever Court decision on the validity and effectivity of the GPL*, 1:4 *SCRIPT-ed* 662 (2004) at <http://www.law.ed.ac.uk/ahrb/script-ed/issue4/GPL-case.asp>
- Hannu Järvinen, ***Legal Aspects of Open Source Licensing***, University of Helsinki, Department of Computer Science (2002)
- Dennis M. Kennedy, *A Primer on Open Source Licensing Legal Issues: Copyright, Copyleft, Copyfuture,* 20 St. Louis U. Pub. L. Rev. 345 (2001), available at <http://www.denniskennedy.com/opensourcedmk.pdf>
- Kennedy, G., *New Codes and Protocols for Cyberspace: Current Issues in Internet Governance*, C.T.L.R. 2000, 6(8), 223-229
- Mathias Klang, *Free software and open source: the freedom debate and its consequences*, First Monday (2005), at <http://www.firstmonday.org/issues/issue10_3/klang>

- Daehwon Koo, *Patent and Copyright Protection of Computer Programs*, 2 Intell. Prop. Qtrly. 188 (2002)
- Paul B. Lambert, *Shareware: Problems of Definition and Legal Nature After The Ozemail Decision,* 22 Eur. Intell. Prop. Rev. 595 (2000)
- Paul B. Lambert, *Copyleft, Copyright and Software IPRS: Is Contract Still King?,* 11 Eur. Intell. Prop. Rev. 165 (2001)
- Lawrence Lessig, *The Future of Ideas*, (2002)
- David McGowan, *Legal Implications of Open-Source Software*, 2001 U. Ill. L. Rev. 241 (2001)
- Robert P. Merges, *The End of Friction? Property Rights and Contract in the 'Newtonian' World of On-Line Commerce*, 12 Berkeley Tech. LJ 115 (1997)
- Axel Metzger , *Free Content Licenses under German Law*, talk given at the Wissenschaftskolleg, Berlin, June 17, 2004, at <http://lists.ibiblio.org/pipermail/cc-de/2004-July/000015.html>
- Eben Moglen, *Enforcing the GNU GPL*, at <http://www.gnu.org/philosophy/enforcing-gpl.html>
- Maureen O'Sullivan, *Making Copyright Ambidextrous: An Expose of Copyleft*, 2 J.I.L & Tech. (2002) at <http://elj.warwick.ac.uk/jilt/02-3/osullivan.html>
- Daniel B. Ravicher, *Facilitating Collaborative Software Development: The Enforceability of Mass-Market Public Software Licenses*, 5 Va. J.L. & Tech. 11 (2000)
- Eric S. Raymond, *The Cathedral and the Bazaar*, (1999)
- Andrew M. St. Laurent, *Understanding Open Source and Free Software Licensing*, (2004)
- Richard Stallman et al., *The GNU Operating System and the Free Software Movement Open Sources: Voices from the Open Source Revolution*, (1999)
- Mikko Välimäki, *The Rise of Open Source Licensing: A Challenge to the Use of Intellectual Property in the Software Industry* (2005)

- Henning Wiese, *The Justification of the Copyright System in the Digital Age,* 24 Eur. Intell. Prop. Rev. 387 (2002)