11

Free and Open-Source Software

ANDRÉS GUADAMUZ

I. INTRODUCTION

PERHAPS NO OTHER software-related topic is more written about, and more misunderstood, than that of free and open-source software (FOSS). The size of the software market is a good indication of the economic importance of software. For 2006, the total information technology (IT) market encompassed US$1.16 trillion, of which software made up US$394 billion.[1] While most of the market belongs to commercial and or proprietary software,[2] the growing importance of computer programs developed through non-traditional means is palpable. Not only has open-source software become a recognisable phenomenon outside of the techno-enthusiast circles, but some open-source programs such as Firefox have achieved a surprisingly high level of market penetration.[3] Google is the largest corporate user of FOSS in the world, and their developers are known to often state that 'Every time you use Google, you're using Linux.'[4]

Given the commercial relevance of software, the written instructions which make up a computer program, generally known as source code, have become a valuable commodity. The complexity and cost of a program can be measured in terms of its source lines of code (SLOC), which give an estimate of the amount of programming time required to create the product. A commercial operating system such as Windows XP can have as many as 40 million SLOC, and is developed by a team of 1,800 programmers.[5] By contrast, Debian 3.0, an open-source operating system, is said to have 105 million SLOC, and has been developed by a multinational

[1] Business Software Alliance, Research and Statistics, http://www.bsa.org/country/Research%20and%20Statistics.aspx.

[2] See below, page X.

[3] Sixteen per cent of the total browser market as of February 2008, according to thecounter.com, see: http://www.thecounter.com/stats/2008/February/browser.php.

[4] C DiBona, 'Joining OIN', Google Blog (2007), http://googleblog.blogspot.com/2007/08/joining-oin.html.

[5] V Maraia, *The Build Master: Microsoft's Software Configuration Management Best Practices* (Indianapolis, IN, Addison-Wesley, 2005) 78.

community without a commercial strategy.[6] Given such costly endeavour, open-source software development seems a counter-intuitive method of coding software. So why does it still exist? Why do thousands of programmers give their time to release source code to the wider community? And most importantly, how is it achieved? What are the legal implications of such a development method? This chapter will address some of these questions.

II. A BRIEF HISTORY OF FOSS

FOSS can be traced back to the creation of the Unix operating system[7] which was developed between 1969 and 1970 by a small team at AT&T Bell Labs.[8] AT&T had been the subject of an antitrust suit in 1949, which resulted, amongst other things, in the signing of a consent decree in 1956 to settle the suit in accordance with stipulations established by the Sherman Antitrust Act.[9] This consent required the company to reveal any patents it held and to license them to competitors. The practical effect of this was that AT&T could not profit from its work on Unix. It disbanded the team that had been developing the operating system and started selling it cheaply and with no guarantees, support or bug fixes of any kind.[10] As the source code was made available, this prompted users to band together and start working on fixes to known problems.[11] In 1974, AT&T was the subject of yet another antitrust suit, which resulted in it parting company with the Bell part of the enterprise, allowing it to commercialise Unix. Eventually, the company released a definitive version of the software in 1979, increased its price and ceased making the source code available to the public.[12]

During this process, and as early as 1973, the software had been rewritten to accommodate new hardware variations. Changes were constantly made by different teams within Bell Labs and the wider academic community—notably from the University of California at Berkeley. It was this unique environment of sharing between experts leading to the creation of Unix which set the tone for the future evolution of the free software and open-source movements.[13]

[6] JJ Amor et al, 'Measuring Woody: The Size of Debian 3.0' (2004) 5(10) *Reports on Systems and Communications* 1.

[7] An operating system, eg MS-DOS, UNIX, Linux, OSX or Windows, is a computer program that allows a computer to run; it serves as the basic interface between the user and the computer.

[8] G Moody, *Rebel Code: Linux and the Open Source Revolution* (London. Penguin, 2002) 13.

[9] Ch 647, 26 Stat 209, 15 USC s 1–7.

[10] Ibid, 14–16.

[11] J Naughton, *A Brief History of the Future* (London, Weidenfeld & Nicholson, 1999) 172–74.

[12] HE Pearson, 'Open Source: The Death of Proprietary Systems?' (2000) 16(3) *Computer Law & Security Report* 151–56.

[13] Moody, above n 8, 5–12.

The 1980s saw the culmination of the development of Unix. Many companies started selling their own versions of the operating system, and the academic community began distributing its own version called Berkeley Software Distribution (BSD).[14] In 1984, a software developer named Richard Stallman, who had been involved with MIT, formed the Free Software Foundation (FSF) to support the nascent ideas of sharing information in the shape of developing free software and to accommodate the GNU[15] project.[16] The 1980s saw the development of software under the auspices of the FSF, encouraging the sharing of code between developers who had never met each other. The FSF had been attempting to generate a new Unix system, but they were missing some key components, notably a kernel,[17] for their operating system.[18]

The movement gained mainstream recognition in 1990 with the development of a new Unix-based kernel[19] called Linux. This began as a hacker project by Finnish programmer, Linus Torvalds. He had been waiting for the developments coming from the FSF relating to their Unix-based clone system, but was impatient and wanted to run it right away. The fact that the FSF had not created a kernel prompted Torvalds to develop his own. He called it Linux and placed it on the Internet for free, asking programmers to improve it.[20] Torvalds secured feedback from other programmers by making the source code for the Linux operating system available to be examined by anybody who wished to do so. This led to the development of different versions (known as distributions) of Linux, giving this operating system an unparalleled amount of stability and security, as the community was in charge of its support.

The term 'open source' itself is relatively new; it was coined during a meeting in February 1998 in Palo Alto, California by a group of software developers with links to Linux.[21] The group met to plan a new strategy in response to the ground-breaking announcement by Netscape that it would be opening its operations and providing the source code of its popular Internet browser to the public. Netscape decided to do this prompted by

14 MK McKusick, 'Twenty Years of Berkley Unix: From AT&T-owned to Freely Redistributable', in C Di Bona, S Ockman and M Stone (eds), *Open Sources: Voices from the Open Source Revolution* (Sebastopol, CA, O'Reilly & Associates, 1999) 31–46.

15 GNU is a recursive acronym that means 'GNU is Not UNIX'.

16 R Stallman, *The GNU Project* (1998), http://www.gnu.org/gnu/thegnuproject.html.

17 The kernel is the fundamental part of an operating system. It is a piece of software responsible for providing access to the computer's hardware by other software applications.

18 R Stallman, 'The GNU Operating System and the Free Software Movement', in Di Bona, Ockman and Stone (eds), above n 14, 53–70.

19 The kernel is the central component in an operating system, which manages system resources and allocates memory and processor usage. {NOT QUITE WHAT IT SAYS n 17?}

20 Moody, above n 8, 31–35.

21 Open Source Initiative, *History of the OSI* (2001): http://www.opensource.org/docs/history.html.

fierce competition from Microsoft.[22] Netscape believed that this gesture would give them a precious opportunity to sell the open-source software development approach to the corporate world.[23]

The need to create a term to describe this approach had become pressing. Until then, the most common way to describe output produced through the non-proprietary approach was by using the expression 'free software'. It was apparent to many software developers that this movement had a tarnished reputation in the business world as a result of association with some of the more radical ideas held by people linked to Stallman and the FSF. In short, it was thought that trying to sell a more commercial non-proprietary approach would not work if they kept referring to the work as 'free software'. A more business-friendly philosophy was needed, along with a new name. Hence 'open source' was coined, a term that was considered less ideological.[24] Many developers welcomed the move, helped in great part by the Linux community using the existing network of websites, message boards and magazines.[25] The rest, as they say, is history.

III. KEY CONCEPTS AND DEFINITIONS

A. Common Characteristics of FOSS Licences

It will be clear from the above discussion that there are two names given to open-source developments: free software and open-source software, referred to together by the acronym FOSS. Contrary to popular misconception, it is important to note that FOSS does not necessarily mean free of charge, and it is not a movement opposed to traditional intellectual property protection. In its more general form, FOSS is simply software that is subject to later modifications by the user or other developers by allowing free access to its source code.[26] In this light, non-proprietary software is considered such if it is released under a licence that allows later modifications, also known as 'forks', together with legal documents which enable others to make their own modifications and distribute them accordingly. FOSS licences also allow a wide range of freedoms for consumers that they

[22] It may even be said that Microsoft's competitive tactics against Netscape were excessive and even predatory, and they prompted the anti-trust case brought by the US Department of Justice against Microsoft. A roadmap to the case can be found here: http://www.stern.nyu.edu/networks/ms/top.html.

[23] Open Source Initiative, above n 21.

[24] Moody, above n 8, 144–55.

[25] Open Source Initiative, above n 21.

[26] Source code is the programming statement expressed in a programming language that exists before the program is compiled into an executable application. The executable form of the software is generally known as the object code, and can only be read by the machine (see Figure 11.1).

Source code	Object code
#!/usr/bin/perl use LWP::Simple; use Math::BigInt; my $html = get my($prime) = $html =~ m{<blockquote>([^<]+)</blockquote>}; $prime =~ tr{0-9}{}cd; $prime = Math::BigInt->new($prime); my $binary = ''; while ($prime > 0) { $binary = pack('N', ($prime % 2**32)) . $binary; $prime /= 2**32; } $binary =~ s{^\0+}{}; local *FH; open(FH, '| gunzip -acq') or die 'cannot gunzip, $!'; binmode FH; print FH $binary;	7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 02 00 03 00 01 00 00 00 B0 D5 04 08 34 00 00 00 8C 10 0E 00 00 00 00 00 34 00 20 00 08 00 28 00 1F 00 1C 00 06 00 00 00 34 00 00 00 34 80 04 08 34 80 04 08 00 01 00 00 00 01 00 00 05 00 00 00 04 00 00 00 03 00 00 00 34 01 00 00 34 81 04 08 34 81 04 08 13 00 00 00 13 00 00 00 04 00 00 00 01 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 00 80 04 08 60 7A 0D 00 60 7A 0D 00 05 00 00 00 00 10 00 00 01 00 00 00 00 80 0D 00 00 00 12 08 00 00 12 08 A8 6E 00 00 58 FF 00 00 06 00 00 00 00 10 00 00 02 00 00 00 48 86 0D 00 48 06 12 08 48 06 12 08 10 01 00 00 10 01 00 00 06 00 00 00 04 00 00 00 04 00 00 00 48 01 00 00 48 81 04 08 48 81 04 08 20 00 00 00 20 00 00 00 04 00 00 00 04 00 00 00 50 E5 74 64 84 37 0C 00 84 B7 10 08 84 B7 10 08 6C 25 00 00 6C 25 00 00 04 00 00 00 04 00 00 00 51 E5 74 64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00 04 00 00 00 2F 6C 69 62 2F 6C 64 2D 6C 69 6E 75 78 2E 73 6F 2E 32 00 00 04 00 00 00 10 00 00 00

Figure 11.1 Source code and object code

would otherwise not have, such as making copies of the work, or installing and distributing the software.

It is important to stress that there are ideological differences in the choice of either open-source or free software. The FOSS acronym is a compromise between the different philosophies. This author prefers the use of the term 'non-proprietary software' as an umbrella definition that refers to the different subcategories encompassed by this movement. It is also covers different types of works, from those offered in exchange for payment, to those that are offered freely to the public. This would include works that are in the public domain,[27] something not included in the definitions of open-source software or free software. Another acceptable term is 'libre software'.[28] This leads to another common acronym: free, libre and open-source software (FLOSS).

Having discussed the general definition of the non-proprietary software model, it is necessary to explain how it fits with other types of software

[27] This is software that has been placed in the public domain specifically by their authors, and is known as public domain software to distinguish it from other types. In software development, public domain does not necessarily mean free; it is simply a legal term to refer to works that are not copyrighted. See Free Software Foundation (FSF), *Categories of Free and Non-Free Software* (2008), http://www.fsf.org/licensing/essays/categories.html.

[28] Libre is a word present in various Romance languages that means free as in freedom, not free as in having no monetary cost. For more on this use, see Working Group on Libre Software, *Free Software/Open Source: Information Society Opportunities for Europe?* (2000), http://eu.conecta.it/paper.pdf.

development, particularly commercial software ownership and proprietary software. Proprietary software is usually defined as a computer program, '[the] use, redistribution or modification [of which] is prohibited, or requires you to ask for permission, or is restricted so much that you effectively can't do it freely'.[29] This is the opposite of non-proprietary software, for which there is a possibility of having access to the code and changing it. It must also be stressed that commercial software is a subset of proprietary software, but not all proprietary software is necessarily commercial.

Commercial software is a program that is created specifically to be marketed and sold.[30] There are several types of software that are offered free of charge, but cannot be changed. Examples of this would be freeware and shareware. Freeware is software that is offered to the public free of charge, but cannot be changed in any way because it is protected by copyright and closed such that the user cannot incorporate its programming into anything else they may be developing. Shareware is software distributed free on a trial basis with the understanding that if the user wants to continue using it, they must acquire a licence. Some software developers offer a shareware version of their program with a built-in expiration date (eg after 30 days, the user can no longer get access to the program). Other shareware (sometimes called liteware) is offered with certain capabilities disabled in the hope that the user will buy the complete version of the program.[31] Another type of proprietary software that should not be confused with non-proprietary software is called a demo. This is software that presents a limited edition of a program, distributed at no cost over the internet, usually before the general commercial release of the software. The objective is to promote the program by presenting some of its features in the hope that users will later buy the full version.

B. Free Software

The free software movement is centred on the concepts and philosophies of developing programs and distributing them freely. As described earlier, the term arose from Richard Stallman's own experiences as a programmer in the 1980s. For a while, the term 'free software' was synonymous with the non-proprietary philosophy of software development. As personal computers became widespread, software programmers continued to exchange pieces of code amongst themselves, providing better ways of developing software in a more efficient manner. Sharing code is an efficient way of programming, as it brings together the work and experience of pro-

[29] FSF, see above n 27.
[30] Ibid.
[31] Ibid.

grammers around the globe, reducing costs and making it easier to find errors. Other factors that have served as an important motivation for sharing code stem from the fact that programmers engaged in the movment mostly worked for non-profit organisations and academic institutions. Ownership of the intellectual property was thus less important than it might be now. Stallman described it as follows:

> Whenever people from another university or a company wanted to port and use a program, we gladly let them. If you saw someone using an unfamiliar and interesting program, you could always ask to see the source code, so that you could read it, change it, or cannibalize parts of it to make a new program.[32]

The decision to create the FSF and the GNU project came from the personal disillusionment felt by Stallman after the collapse of the early software-sharing community, and a notable increase in the development of proprietary software. Stallman explains that software began to have restrictions imposed in the shape of proprietary licences telling users they could not access the source code to modify the software, or share it with other people with a view to enhancing its functionality. If the user engaged in any tinkering with the code, then he stopped being a hobbyist and became a pirate.[33] Eventually, Stallman and other like-minded programmers created a powerful software development force under the general principles of non-proprietary software.

Stallman defines free software as having the following four characteristics:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.[34]

As understood by the proponents of free software, programmers and other developers can charge for the software if it is their desire to do so, but the same underlying freedoms must exist whether or not it is acquired for a monetary fee. The user must still have all of the freedoms described, with access to the source code as the most basic requisite.[35]

According to the GNU project, there are several types of free software, some that conflict with the values advocated by the FSF, and some that do

[32] Stallman, above n 16.
[33] *Revolution OS*, directed by JTS Moore (2001).
[34] FSF, *The Free Software Definition* (2007), http://www.fsf.org/licensing/essays/free-sw.html.
[35] FSF, *Selling Free Software* (2006), http://www.fsf.org/licensing/essays/selling.html.

not. The main category is an overarching free software definition, which states that the software qualifies as free software if it 'comes with permission for anyone to use, copy, and distribute, either verbatim or with modifications, either gratis or for a fee. In particular, this means that source code must be available.'[36]

The FSF has promulgated a policy of making available software termed 'free software', stating that the software must follow the philosophy described. If the program comes with too many restrictions, it will not be granted certification. Certain restrictions are acceptable if these are not excessive.[37] One of the main restrictions is that of subsequent licensing, under which software must be maintained as 'free'.

C. Open Source

In its broadest sense, open source is the opposite of 'closed source', the traditional proprietary approach to software development in the commercial world. Closed source is software 'in which the customer gets a sealed block of bits which cannot be examined, modified, or evolved'.[38] The common thread in both free software and open-source software is that the source code remains available for examination, modification and peer review.

As mentioned, the official definition was based on the Debian Free Software Guidelines, a licensing model written by software developer Bruce Perens and which accompanies the Debian GNU/Linux system, a Linux distribution.[39] These existing documents were improved and modified by developer Eric Raymond and form what is known as the 'open-source definition' (OSD). The definition not only requires that open-source software should make available the original code, but also sets the following principles underlying all open-source software:

> *Free Redistribution:* this means the software will have no restrictions regarding further distribution as part of another package.
> *Source Code:* the source code will be made available for examination, either by including the software in the software package or by making it available at a public location.
> *Derived Works:* the licence must allow modifications and the development of derived works.
> *Integrity of The Author's Source Code:* the licence may allow restrictions about changes to the original source code only if the distributor assumes the responsibility of fixing any problem found with the software.

[36] FSF, above n 27.
[37] Ibid.
[38] E Raymond, 'Keeping an Open Mind' [1999] *Cyberian Express* April, http://tuxedo.org/~esr/writings/openmind.html.
[39] The guidelines can be found at: http://www.debian.org/social_contract.html#guidelines.

No Discrimination Against Persons or Groups: OSS can be used both for 'abortion clinics and anti-abortion activists'.[40]
No Discrimination Against Fields of Endeavour: the licence will not discriminate the usage of the software for specific fields of work.
Distribution of License: there will be no need for the development of additional licences for those who receive the software from any party other than the licensee.
Licence Must Not Be Specific to a Product: if the software is distributed within a larger software bundle, the software will still be subject to the larger product license.
The Licence Must Not Restrict Other Software: this means that there will not be any restrictions placed on other software being distributed under the same software bundle.[41]

The main characteristic of open source as underpinned by these points is the idea of peer review of a work. By allowing more people access to the code that makes up a software, that software will gain in dependability, stability and security. In the words of Raymond, 'open source puts the software customer in the driver's seat, dramatically lowers total cost of ownership, and is the only recipe that works for high reliability'.[42]

Since the original coining of the term, open source has gained substantial recognition in technical circles. But the success has come at a price. As the term gained more credibility and popularity, there was nothing to prevent a software developer releasing a software program and labelling it 'open source' as a marketing ploy. This was possible without the software actually fulfilling any of the requirements under the definition, or even where it fell into the proprietary category. The lack of an enforcement mechanism prompted several activists to create the Open Source Initiative (OSI) to analyse software licenses and measure them against the OSD, and hence to certify software as genuinely open source.[43] The OSI maintains a public list of all software that it has certified, thus enabling consumers and others to know whether the software they are using is indeed open source.[44]

D. Copyleft

The third most important concept to understand in FOSS licences is that of copyleft. Copyleft is a legal mechanism contained in a licence which maintains the general freedoms awarded to users of software licensed as FOSS,

[40] Interview with Bruce Perens. *Revolution OS*, above n 33.
[41] Open Source Initiative, *The Open Source Definition*, Version 1.9 (2006), http://opensource.org/docs/osd.
[42] Raymond, above n 38.
[43] Open Source Initiative, *Certification Mark* (2006), http://opensource.org/docs/certification_mark.html.
[44] The list is maintained at: http://opensource.org/licenses.

but by acquiring a program released as copyleft, the user agrees that the software will not be used to develop closed-source applications derived from it.[45] This is done via a clause that requires all derivatives arising from the original code to be released under the same freedoms as those under which they were received.

Copyleft was developed from a perceived need to protect the fruits of non-proprietary development. After several years of producing computer programs through sharing expertise and offering the code to the public, some programmers started taking the source code, tweaking it and selling it as commercial proprietary software. Their development costs were thus very low.[46] Copyleft licensing became the only means of stopping companies from profiting from non-proprietary products and then creating products that went against the spirit of the free software movement.

Copyleft is an elegant legal solution that imposes a restriction through a chain of software distribution. The contractual clause ensures the propagation of a licence through that same chain, aptly named 'viral contracts' by Radin, who defines them as 'contracts whose obligations purport to "run" to successor of immediate parties'.[47] These contracts spread in a viral form as the licensee must include the terms of the licence in any subsequent 'fork' distributed because that obligation is part of the contract. Subsequent licensees will have to impose the same contractual terms in further licences they promulgate in perpetuity.

Despite the fact that copyleft licences tend to promote the free software principles and the definitions drafted by the FSF and Stallman, some of the contractual restrictions in copyleft licences have prompted criticism from enterprises and commercial users. Copyleft is a strong tool to keep code open, but it may also prove overly restrictive for enterprises wishing to profit from those derivatives.

IV. FOSS LICENCES

A. Licence Ecology

This chapter has so far described the underlying philosophies and principles that determine the development strategy known as FOSS. These are implemented and enforced through legal documents. Although the definitions and principles are important, one could argue that licences are

[45] LE Rosen, *Open Source Licensing: Software Freedom and Intellectual Property Law* (Upper Saddle River, NJ, Prentice Hall, 2004), 105.

[46] FSF, *Copyleft: Pragmatic Idealism* (2005), http://www.fsf.org/licensing/essays/pragmatic.html.

[47] MJ Radin, 'Humans, Computers, and Binding Commitment' (2000) 75(4) *Indiana Law Journal* 38.

the one thing that separates proprietary and non-proprietary software models.

The inclusive nature of the definitions of free and open-source software mean that there is a large number of software licences that fall into one or other camp.[48] At the time of writing, the OSI listed 72 certified licences while the FSF identifies more than 100.[49]

As FOSS licences cover such a broad spectrum, classification is difficult. One could distinguish them as either FSF or OSI certified, but as has been pointed out, there is substantial overlap between the definitions and hence the licences. One way to categorise them is as either copyleft or non-copyleft. The common elements of FOSS licences are:

- *Attribution*. Copyright notices are to be kept intact, and the author(s) will be attributed in the code.
- *Access to the source code*. This is the most basic common element in all licences. The source code will be included either with the distribution, or is to be made available to the public in an open source repository.[50]
- *User rights*. Users are granted a non-exclusive right to use, copy and distribute the work.
- *Derivatives*. All open-source licences allow developers to make modifications to the source code and make those modifications available to the public. This modification may come with restrictions, such as the one present in copyleft licences.

The most basic FOSS licence is the BSD licence,[51] which is short and concise. There are several variations of the licence, evidencing its popularity within the FOSS community. The main part is the assignation of rights, which states:

> Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
> —Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
> —Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
> —Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.[52]

[48] Research by the author has uncovered a total of 129 unique FOSS licences.
[49] Some of them are on both lists. See http://www.fsf.org/licensing/licenses/.
[50] Such as SourceForge, located at http://sourceforge.net.
[51] http://www.opensource.org/licenses/bsd-license.php.
[52] Ibid.

This is a permissive licence with regard to the range of rights granted to the user, as it allows all redistribution of the software both in binary and source code. Other short and elegant OSS licences, eg the MIT licence, take a similar approach.[53]

The most comprehensive non-copyleft, open-source licence is the Apache 2.0 licence,[54] which is more restrictive than both the BSD and MIT licences. This is an important legal document because the Apache Software Foundation produces the Apache HTTP Server software, one of the most widely used open-source programs in the world.[55] The Apache licence maintains the freedom to redistribute the software in binary or source code form, the freedom found in most FOSS licences, but adds the right to create derivative works from the original.[56] The redistribution and modification of the work are allowed provided the copy or derivative work is included with proper attribution of the originator(s) of the program, and that the copyright notices attributing ownership of the code to the original programmers are attached. This approach falls short of the viral clause included in copyleft licences and demonstrates one of the main differences between the licensing models. Another interesting feature included in the Apache licence is the assignment of copyright and grant of patent licence despite the fact that the Apache Software Foundation (drafters of the patent) have stated that that they do not own, and have not applied for, any software patents.[57]

A new development in FOSS licensing is that some developers have been using open licences not originally designed for software. One example is Creative Commons, a project started by Lawrence Lessig following the licensing ideals of FOSS, but directed towards the protection of creative works, such as literary, artistic and musical creations.[58] Although Creative Commons licences are not designed to cover software and source code, there is nothing in the way in which they are drafted that excludes them from being used to licence software.

53 http://www.opensource.org/licenses/apache2.0.php.

54 See full text at http://www.opensource.org/licenses/apache2.0.php.

55 As of February 2008, Apache commands 51 per cent of the total web server market, while some competitors, such as Google Server, are based on Apache code. See Netcraft, *February 2008 Web Server Survey* (2008), http://news.netcraft.com/archives/2008/02/06/february_2008_web_server_survey.html.

56 In UK law, this would be known as 'adaptations'.

57 Apache Software Foundation, Apache License v2.0 and GPL Compatibility, http://www.apache.org/licenses/GPL-compatibility.html.

58 For more about Creative Commons, see N Elkin-Koren, 'What Contracts Can't Do: The Limits of Private Ordering in Facilitating a Creative Commons' (2005) 74(2) *Fordham Law Review* 375; and S Dusollier, 'The Master's Tools v the Master's House: Creative Commons v Copyright' (2007) 29(3) *Columbia Journal of Law & the Arts* 271.

B. GNU General Public Licence Version 2

The GNU General Public License (GPL) is the most important licence in the FOSS movement. At the time of writing, 68 per cent of all projects listed in the SourceForge open-source repository are released under the GPL.[59] But what makes the GPL the licence of choice for non-proprietary development?

The GPL was first drafted in 1985[60] by Richard Stallman in order to accommodate the ideas of free distribution of source code implemented by the FSF.[61] The GPL is the first copyleft licence, and as such is designed to maintain the four freedoms in the FSF definition, but it also contains the copyleft clause.[62] Version 2 of the licence, drafted by Stallman and Moglen, was released in 1991.[63]

The GPL is the legal framework that sustains most of the copyleft system.[64] It reads as a mixture of a legal contract[65] and an ideological manifesto. The preamble to the work restates clearly some of the most common beliefs of free software movement and non-proprietary approach, with several admonitions about the meaning of the word 'free'. The key point is that the source code must be made available to the users. The preamble states:

> For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.[66]

The licence specifies that this is achieved by two means: by protecting the software through copyright; and by providing the users with a licence that gives them the freedom to use and modify the software in any way they see fit if they meet the stated conditions. The main body of the licence reiterates these ideas. Section 1, for example, states:

> 1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence

[59] Out of 73,978 listed projects, 50,013 were released with the GPL. See http://sourceforge.net/softwaremap.

[60] However, the official dating of version 1 is 1989. This is because the GPL was known originally as the EMACS General Public License.

[61] Stallman, above n 18, 59.

[62] Moody, above n 8, 26–29.

[63] Ibid.

[64] The full text of version 2 of the licence can be found at http://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[65] Some people do not believe that the GPL is a contract, a question that will be discussed in detail later.

[66] FSF, GNU General Public License v2, Preamble.

> of any warranty; and give any other recipients of the Program a copy of this License along with the Program.[67]

This section also mentions that the user can make monetary charges when distributing the copy as long as the charges are for expenses in making copies of the software. This is consistent with the general free software characteristic that does not discriminate against commercial software as long as it is not proprietary commercial software.

Many of the provisions of the GPL can be found in other non-proprietary software licences. What makes the GPL different to most others is section 2(b) as this is where the restrictions against using the software to create commercial software are specified. The section reads:

> 2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions: . . . b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.[68]

This is the best example of the copyleft clause, as any software developed using the open-source code of the program must ensure that the GPL is transferred to further users of the derivative software. As evidenced by the widespread adoption of the GPL version 2, this is indeed a viral licence because derivatives must be released under the same licence, thus ensuring downstream use of the licence terms.[69]

Despite the fact that copyleft licences tend to promote the free software principles and the definitions drafted by the FSF and Stallman, some of the contractual restrictions in licences such as the GPL have come in for criticism from enterprises and commercial users.[70] Despite this, the GPL has stood well against most challenges, as exemplified by its longevity and the relatively small amount of litigation that it has generated.[71]

There is another version of the GPL, called the GNU Lesser General Public License (LGPL).[72] This version is almost identical to the GPL but does not contain a copyleft clause. This is because there is sometimes need for software to interact with non-FOSS code and to do so a non-copyleft

[67] Ibid, s 1.
[68] Ibid, s 2(b).
[69] See A Guadamuz, 'Viral Contracts or Unenforceable Documents? Contractual Validity of Copyleft Licenses' (2004) 26(8) *European Intellectual Property Review* 331.
[70] For some criticisms, see A Guadamuz, 'Legal Challenges to Open Source Licences' (2005) 2(2) *SCRIPT-ed* 301.
[71] See JS Miller, 'Allchin's Folly: Exploding Some Myths about Open Source Software' (2002) 20 *Cardozo Arts & Entertainment Law Journal* 491.
[72] http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html.

version is required or else the linking libraries would have to be released under the GPL.

C. GPL Version 3

While the longevity and success of the GPL have been a testament to its legal validity,[73] the FSF felt that it was time for an updated version, giving the opportunity to fine-tune some legal issues and to respond to developments in the free software community. Prior to the release of the draft, Stallman and Moglen identified key issues they believed needed to be addressed by the licence.[74] These were as follows:

- *Internationalise the licence*. While GPL version 2 was drafted with US law in mind, it aimed to meet the international copyright principles present in the Berne Convention.[75] While most issues of enforcement have taken place outside of the US,[76] it was felt that the licence required clarity in international compliance.
- *Standard setting*. Stallman and Moglen argued that the GPL has become an international standard for the FOSS industry, so any update should take this factor into consideration.
- *Responding to changing circumstances*. The FOSS community, and in particular the principles of free software, are faced with new threats, such as patentability of software, trusted computing and digital rights management (DRM). The GPL must change to meet those threats.[77]

So, what is contained in the new licence? There have been some major changes in both style and substance. The GPL has never been known for its conciseness and clarity, so it is unfortunate that version 3 is longer and more opaque than its predecessor.[78] This is a real problem, as even before being updated, the GPL was subject to much differing legal interpretation.[79]

The drafting process was an interesting exercise in governance. The FSF must be commended for the inclusive and open process through which it undertook discussions on drafting. The first draft of the GPL version 3 was

[73] For further legal analysis of its validity, see R Gomulkiewicz, 'De-bugging Open Source Software Licensing' (2002) 64 *University of Pittsburgh Law Review* 75.

[74] E Moglen and R Stallman, 'GPL Version 3: Background to Adoption' (5 June 2005), http://www.fsf.org/news/gpl3.html.

[75] Berne Convention for the Protection of Literary and Artistic Works 1886.

[76] J Höppner, 'The GPL Prevails: An Analysis of the First-ever Court Decision on the Validity and Effectivity of the GPL' (2004) 1(4) *SCRIPT-ed* 662, at http://www.law.ed.ac.uk/ahrb/script-ed/issue4/GPL-case.asp.

[77] Stallman and Moglen, above n 74.

[78] GPL v3: http://www.fsf.org/licensing/licenses/gpl.html.

[79] R Gomulkiewicz, 'General Public License 3.0: Hacking the Free Software Movement's Constitution' (2005) 42(4) *Houston Law Review* 1015, 1034–36.

made available for public discussion in January 2006. The website on which it was made available was open for public comment, and hundreds of notes from the community were shared for all to see. Two more drafts were made available which were the subject of closed and open meetings, conferences and discussion that included stakeholders and GPL users in general. Although the participation was unprecedented in such a drafting exercise, it must be said that the end result seems to have suffered from too much compromise. The text of the GPL version 3 is nearly unreadable in places.

The document which identified the need for a redraft prompted expectations that the GPL v3 would overhaul the protection of GPL software vis-à-vis software patents and it would update the copyleft clause present in the existing section 2(b). Although these are topics covered in the final text, the real surprise came with regards to DRM and in particular technical protection measures (TPMs). The new GPL makes a statement on the future of open-source usage in the entertainment industry by including strong wording against TPMs. One of the most contentious parts is section 3 on DRM:

> No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.[80]

This paragraph specifically excludes all works distributed under the GPL from the anti-circumvention measures in the WIPO Copyright Treaty[81] (WCT) by stating that the licensed software shall not constitute 'an effective technological protection measure', thus making it inapplicable for protection. In other words, the distribution of derivatives with works that contain certain restrictive types of DRM is prohibited. This is an elegant legal solution to the perceived problem, as it excludes all relevant software from anti-circumvention legislation by contractual means.

Surprisingly, and contrary to what many expected, GPL version 3 seems to direct all of its power against TPMs but not software patents. This is perhaps a reflection of the fact that some of the biggest free and open-source software players in the US are acquiring patents as well.[82] Stallman and Moglen seem pragmatic regarding software patents and recognise that FOSS developers may be involved in complex patent licensing transactions. Hence their implicit recognition of the status quo.[83]

[80] S 3 of GPL v3.

[81] Specifically, Art 1 WCT, which states that 'Contracting Parties shall provide adequate legal protection and effective legal remedies against the circumvention of effective technological measures.'

[82] Apache, IBM and Novell have applied for thousands of software patents in the US. For more about this, see A Guadamuz, 'The Software Patent Debate' (2006) 1(3) *Journal of Intellectual Property Law & Practice* 196.

[83] See their draft rationale at http://gplv3.fsf.org/rationale.

Version 3 expands on the implicit patent licensing in GPL version 2, making it an explicit patent grant in section 11. The new patent grant reads:

> Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In addition to this, GPL version 3 adds paragraphs relevant to patents that are drafted with specific issues in mind. In particular, there is a paragraph that seems to be drafted to respond to patent licence agreements between Microsoft and some FOSS developers, such as Novell.[84] Another issue-specifc section relates to so-called Tivoisation of code, which is an existing loophole present in previous versions by which modified code released under the GPL is distributed to the public but locked using hardware (such as the copyleft code contained in TiVO players).[85]

These drafting practices are, in the author's opinion, the biggest problem with the GPL. A document that is supposed to act as the constitution of the free software ideals should not be bogged down in these details.

Another big change in GPL version 3 is that it revamps the old copyleft section 2(b). As explained earlier, under the current GPL the copyleft aspects only apply to derivative works that are distributed to the public. In other words, you take a work under the GPL, change it and then distribute your own adaptation. Then you have to redistribute it under the GPL. This simple rule generally resulted in clear-cut cases in which the GPL would apply, and those where it would not. For example, imagine that a hardware developer creates a driver module that interacts with the Linux kernel (distributed under the GPL). It is not part of the kernel, but it interacts with it. Under GPL version 2, it is clear that this module is not a derivative, and therefore it does not need to be distributed under the GPL.

What happens with GPL version 3? The situation with derivatives is less straightforward. The copyleft section in the new GPL has been given a boost. Users and developers still have the right to install and use GPL software without restrictions. Developers also have the right to privately modify the software, unless they have initiated a patent suit 'against anyone for making, using or distributing their own works based on the Program'. The problem is that of modifications that are distributed. Consider section 5(c) (the old section 2.b):

[84] This is the Patent Cooperation Agreement—Microsoft & Novell Interoperability Collaboration, signed October 2006, see http://www.microsoft.com/interop/msnovellcollab/patent_agreement.mspx.

[85] C O'Riordan, 'Tivoisation Explained—Implementation and Harms', FSF Europe Briefing paper (December 2006).

> (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.[86]

This would certainly apply to anyone who includes any sort of modification of code into her work. Imagine, for example, that you include modified modules from the Linux kernel in your work so as to be compatible with the kernel. An initial reading of section 5(c) would lead one to believe that the entire program would need to be licensed under the GPL. However, there is an exception for compilations. Section 5 reads in a later paragraph:

> A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an 'aggregate' if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.[87]

This seems like excessive protection. The wording generates legal uncertainty—something picked up by commentators of the draft.[88] The paragraph tries to rationalise specific cases in which the revamped GPL copyleft section will apply by inventing a new definition as to what a compilation is. Software distributed in the same distribution medium has to be GPL if it is a 'compilation', but not if it is an 'aggregate'. Why create the new terminology?

All of the above points make for an interesting legal mechanism, but one that seems unnecessarily complex. Those developing small and medium FOSS projects looking for a licensing scheme may very well think twice about migrating to the new version. Although the licence was issued last year, adoption has been slow. According to a survey by Evans Data, by September 2007 only 6 per cent of FOSS projects had migrated to GPL version 3. More worryingly, 43 per cent of respondents claimed that they would never implement the new licence.[89] Time will tell if this approach is maintained.

86 S 5(b) draft GPL v3 (emphasis added).
87 S 5 draft GPL v3.
88 For a series of comments, see: http://gplv3.fsf.org/comments/.
89 E Corradetti, 'Open Source Developers Staying Away From GPLv3, New Evans Data Survey Shows', Evans Data Corporation (25 September 2007), http://www.evansdata.com/press/viewRelease.php.

V. SOME LEGAL ISSUES

A. Contract or Licence? The Problem with Consideration

The issue of the legal nature of open-source licences has generated controversy in legal circles. Some FOSS proponents, particularly those in the free software camp, are adamant that FOSS licences are not contracts, but copyright licenses. This may seem like an arcane legal distinction, but it raises important questions about enforcement, jurisdiction, applicable law and even about the validity of the licence. There may also be different legal effects in some jurisdictions depending on whether a contract is a contract for sale of goods, or a contract for sale of services.[90]

A licence documents a legal relationship through which the licensee is granted permission to perform acts that could not otherwise be done legally. When one buys commercial software, the licence allows the licensee to install a copy of the program on a computer—an act that would otherwise infringe copyright. In some jurisdictions, licences are contracts, and are classified as a specific type of contractual obligation. However, in other countries, licences are not contracts.

The question at the heart of this dichotomy between contracts and licences rests on the issue of reciprocity, known in common law jurisdictions as consideration.[91] A typical argument is presented by FOSS advocate and blogger Pamela Jones, when she asks: 'Why isn't it a contract? Because there are no further agreed-upon promises, no reciprocal obligations.' [92] The lack of reciprocity is also mentioned by Moglen:

> A contract . . . is an exchange of obligations, either of promises for promises or of promises of future performance for present performance or payment. The idea that 'licenses' to use patents or copyrights must be contracts is an artifact of twentieth-century practice, in which licensors offered an exchange of promises with users: 'We will give you a copy of our copyrighted work,' in essence, 'if you pay us and promise to enter into certain obligations concerning the work.' With respect to software, those obligations by users include promises not to decompile or reverse-engineer the software, and not to transfer the software.[93]

[90] For a discussion of a distinction between software as sale of goods or sale of services in software, see A Taubman in the Introduction to this edition and HL MacQueen, 'Software Transactions and Contract Law', in L Edwards and C Waelde (eds), *Law and the Internet: Regulating Cyberspace* (Oxford, Hart Publishing, 1997).

[91] Famously defined in *Currie v Misa* (1875) LR 10 Ex 153 as 'some right, interest, profit or benefit accruing to one party, or some forbearance, detriment, loss or responsibility given, suffered or undertaken by the others'.

[92] See P Jones, 'The GPL Is a License, not a Contract' [2003] *Linux Weekly News* 3 December, http://lwn.net/Articles/61292/.

[93] E Moglen, *Enforcing the GNU GPL* (2001), http://www.gnu.org/philosophy/enforcing-gpl.html.

The problem with this interpretation is that it comes from a jurisdiction-specific analysis of contract law. In most civil law and mixed legal systems, such as Scotland, contracts are present when the requirements of offer and acceptance have been met.[94] This means that unilateral acts can constitute contracts under the appropriate conditions.[95] However, in common law jurisdictions the additional requirement of consideration must be met. This is the reason why Jones and Moglen place so much emphasis on the issue of reciprocity in FOSS licences. It is argued that open-source systems are usually offered for free, which would mean that the important contractual step of consideration is missing. Therefore, FOSS licences should be classified as unilateral legal acts, and not contracts as such.

One could argue, however, that there is consideration in some FOSS licences, particularly in copyleft licences. Risking oversimplification of the rich case-law dealing with consideration and contract formation in common law, one could generalise the concept as one of reciprocity, as has been expressed earlier. If one defines consideration as such, then it would be possible to see how copyleft clauses would fulfil the requirement of consideration in jurisdictions where it is required. Copyleft clauses impose an obligation to release modifications under the same licence as the one under which it was obtained. The contract then would be formed like this: making the software available under a FOSS licence would be the offer, using the software under those conditions would be the acceptance, while consideration would be met by the obligation imposed in the copyleft clause. However, some scholars disagree that copyleft clauses meet the requirement of consideration. Giles, for example,[96] argues that copyleft is, at best, illusory consideration, and has found several cases to support his view. Of note he cites the case *British Empire Films Pty Ltd v Oxford Theatres Pty Ltd*,[97] where the courts found unilateral promises that depended entirely on the will of one of the parties as illusory consideration. As expressed by O'Brian J:

> It is common ground that the plaintiff is obliged to supply nothing, and a supposed consideration which is entirely dependent upon the will of the plaintiff whether it will ever become operative is illusory.[98]

However, this does not seem to be a useful analogy, as the promises dealt with in illusory consideration case-law are very specific, and seem not to translate well into the realm of open-source software. For example, participants in the software development community may have their options

[94] HL MacQueen and JM Thomson, *Contract Law in Scotland*, 2nd edn (Edinburgh, Tottel, 2007) 54–56.
[95] Ibid, 56–59.
[96] B Giles, '"Consideration" and the Open Source Agreement' (2002) 49 *NSW Society for Computers and Law*, http://www.nswscl.org.au/journal/49/Giles.html.
[97] [1943] VLR 163.
[98] Ibid, 168.

seriously curtailed if they use copyleft software, as they will be under a very real obligation to release modifications under the same licence as the one under which they received it. True, they may choose not to use the software, but is that not the case in all contracts? In other words, once they have accepted the terms and conditions of the licence by using the software and modifying it, the obligation imposed on them seems very real, and not illusory at all.[99]

The contract/licence dichotomy was discussed at length in a recent case in the United States, *Jacobsen v Katzer*,[100] which dealt precisely with this question.

The case involved Robert Jacobsen, an open-source developer participating in an open-source project called the 'Java Model Railroad Interface' (JMRI), a model train software released under the Artistic License.[101] Jacobsen received a letter demanding licence fee payments from a company named Kamind Associates, owned by Matthew Katzer, which had obtained software patents over model railroad software.[102] Jacobsen decided to pre-empt legal action and sued Katzer first, alleging that the patent was invalid on the grounds of obviousness, and for failure to meet disclosure requirements. He later amended the complaint to include copyright infringement, claiming his software pre-dated Katzer's.

The US District Court for the District of Northern California ruled on a motion to dismiss by the defendants, and on a motion for preliminary injunction from the plaintiff. The District Court granted some of the motions to dismiss, denied others and denied the claim for preliminary injunction. The important part of the decision for the current discussion is the analysis of the copyright infringement claims. The District Court declared that because the software was released to the public online through an open-source licence, there was therefore permission to use the software. The Artistic License is not a copyleft licence: it allows modification and the creation of derivatives, provided that those doing so doing insert prominent notices on each file, and perform one of the following:

> (a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as ftp.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.
>
> (b) use the modified Package only within your corporation or organization.
>
> (c) rename any non-standard executables so the names do not conflict with

[99] Credit must go to FOSS legal expert Robert Gomulkiewicz, who explained some of these ideas in an e-mail discussion in the Cyberprof mailing list. Other FOSS experts who come on the side of thinking of the licences as contracts is OSI legal counsel Larry Rosen. See Rosen, above n 45, 59–66.

[100] 2007 US Dist. LEXIS 63568; 535 F 3d 1373 (Appeal).

[101] http://www.opensource.org/licenses/artistic-license.php.

[102] US patent 7,216,836.

> standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.
>
> (d) make other distribution arrangements with the Copyright Holder.[103]

The District Court made it clear that such restrictions are not copyright restrictions but contractual obligations.

> Based on the both the allegations in the amended complaint and the explicit language of the JMRI Project's artistic license, the Court finds that Plaintiff has chosen to distribute his decoder definition files by granting the public a non-exclusive license to use, distribute and copy the files. The nonexclusive license is subject to various conditions, including the licensee's proper attribution of the source of the subject files. However, implicit in a nonexclusive license is the promise not to sue for copyright infringement. . . . Therefore, under this reasoning, Plaintiff may have a claim against Defendants for breach the nonexclusive license agreement, but perhaps not a claim sounding in copyright. . . . However, merely finding that there was a license to use does not automatically preclude a claim for copyright infringement. . . . The condition that the user insert a prominent notice of attribution does not limit the scope of the license. Rather, Defendants' alleged violation of the conditions of the license may have constituted a breach of the nonexclusive license, but does not create liability for copyright infringement where it would not otherwise exist.[104]

The District Court stated that there should be no presumption of a copyright infringement claim, and that such claim should be proven before the plaintiff can make its case. If the plaintiff cannot provide evidence that such a claim may be successful in court, then Jacobson could only rely on the contractual elements of the licence in order to seek redress—in other words the failure to place attribution notices is not enough to make a copyright claim, only a contractual one. Interestingly, the District Court understood perfectly the trade-off in open-source licences that rests at the very heart of the contract/licence dichotomy: if the user complies with the licence, there is permission to use the software, and therefore there is no copyright infringement. But if there is no claim for copyright over the work, the only claim possible is breach of contract.

Katzer appealed the District Court ruling, and it made its way to the Court of Appeals for the Federal Circuit (CAFC), which overturned the decision holding that open-source licences set out permissions to use the work, and if the licence disappears, then the user would be infringing copyright. The ruling makes for interesting reading:

> In this case, a user who downloads the JMRI copyrighted materials is authorized to make modifications and to distribute the materials provided that the user follows the restrictive terms of the Artistic License. A copyright holder can grant the

[103] http://www.opensource.org/licenses/artistic-license.php
[104] {ADD EXPLICIT REF} (emphasis added)

> right to make certain modifications, yet retain his right to prevent other modifications. Indeed, such a goal is exactly the purpose of adding conditions to a license grant. The Artistic License, like many other common copyright licenses, requires that any copies that are distributed contain the copyright notices and the copying file.[105]

It is heartening that the CAFC has understood the concepts behind open-source licensing. In various passages, it clearly appreciated the basis of the movement and the underlying rights. The CAFC has delivered the highest instance recognition to open licences—an encouraging sign for FOSS development. The appeal has also pleased many in the FOSS community. For example, the ruling closely follows the reasoning presented an *amicus curiae* arguing against the District Court ruling. In it, the OSI, the Linux Foundation and others argued that 'it would be enormously beneficial to public licensing for this Court to state clearly a rule regarding the importance of interpreting public licenses in a manner consistent with their unique nature and federal copyright policy'.[106]

Despite the final decision in *Katzer*, this author still considers that it is preferable to classify FOSS licences as contracts. Why is the FSF adamant that its licences, in particular the GPL, are not contracts? There are practical reasons why some FOSS proponents insist on licences not having contractual strength. Moglen is on record as stating that the GPL primarily rests on copyright and the international protection awarded by the Berne Convention. He is uneasy with the global variability of contract law, and concerned that a judge in one jurisdiction may impose local contract law interpretations which may affect the project globally.[107] By using copyright instead of contract it is in the licensee's best interest to make sure that the licence is valid as he would otherwise be infringing copyright. Moglen says:

> So all that I do is bring an infringement action. It is the defendant's responsibility to prove license and the only credible license for the defendant to plead is my license, because code is not otherwise available except under that license.[108]

This seems a rather negative view of copyright licensing—as if all use of the licensed work should be considered a priori infringement until proven otherwise. A similar example might be if one invites a guest home, and the moment they enter the premises, trespass is claimed. The invitation can be seen as a unilateral licence allowing the guest to perform an action they

105 add ref?

106 AT Falzone and CK Ridder, Brief of Amici Curiae Creative Commons Corporation, the Linux Foundation, the Open Source Initiative, Software Freedom Law Center, Yet Another Society, the Perl Foundation, and Wikimedia Foundation, Inc. in Support of Plaintiff-Appellant and Urging Reversal (2008), http://jmri.sourceforge.net/k/docket/cafc-pi-1/ccc_brf.pdf.

107 M Hardin, 'Interview Eben Moglen, Legal Counsel, Free Software Foundation', Auskadi Blog (2004), http://auskadi.civiblog.org/blog/_archives/2005/6/25/972325.html.

108 Ibid.

would not otherwise have a right to do. However, as discussed above in *Jacobsen v Katzer*, there is a real possibility that the user will not have a claim in copyright and so the obligations contained in the licence could only be enforced through contract.

The above analysis is important point if one considers the practicalities in FOSS development. Code is passed between and modified by people all over the world. It is possible to imagine a situation where code has reached a developer in such a modified state that the original owner will no longer be able to claim copyright over it. English courts have considered the minimal amount of code that would be infringing by following the general qualitative test in cases of copying from another work. In both *Richardson Computers v Flanders*[109] and *Ibcos v Barclays*,[110] the courts found that if there had been any copying from a protected original work, there had to be an analysis of whether such copying had been substantial to determine infringement. Even quantitatively minimal copying might be qualitatively substantial. This is evident in the case of *Cantor v Tradition*,[111] where copying of original source code took place from former employees of a financial services company. In this case, expert witnesses found that only 2 per cent of the original source code had been copied, accounting for only 2,952 lines of code out of 77,000. The lines of code were deemed to be of importance for some modules in the resulting software, but the copying was not considered sufficient to find infringement. The copier nonetheless agreed to take financial responsibility for the infringed code and offered to pay for it. One could thus imagine a situation where enough changes have occurred to create a new work—one where the original code would no longer be subject to copyright.

B. Enforcement

Considering there are such large numbers of participants and projects, a surprising feature of FOSS is the very small number of cases that have been brought to court. There are perhaps two reasons. Firstly, FOSS projects are, for the large part, small and medium-sized endeavours, where large numbers of participants only make one-off contributions.[112] The small size of the projects, coupled with the fact that most of the developers are not motivated by profit,[113] means that litigation will not be a priority for developers. As there is a strong sense of community in the open-source

[109] *John Richardson Computers Ltd v Flanders and Chemtec Ltd* [1993] FSR 497.
[110] *Ibcos ComputersLtd v Barclays Mercantile Highland Finance* [1994] FSR 275.
[111] *Cantor Fitzgerald International v Tradition (UK) Ltd* [1999] Masons CLR 157.
[112] J Lerner and J Tirole, 'Economic Perspectives on Open Source', in J Feller et al (eds), *Perspectives on Free and Open Source Software* (Cambridge, MA, MIT Press, 2005) 55–57.
[113] R Ghosh, 'Understanding Free Software Developers: Findings from the FLOSS Study', in Feller et al, above n 112, 34–42.

environment,[114] conflict is usually resolved internally. Second, the FSF has a very effective enforcement body for policing implementation of the GPL called GPL-violations.org.[115] It is a non-profit branch of the FSF which monitors GPL usage, informs the public about infringement, names and shames perpetrators, and, if necessary, issues cease-and-desist letters to the offenders.[116] In an industry that is famously risk-averse, and where profits can be tight, it pays to comply with low-impact enforcement measures such as cease-and desist letters. Similarly, companies that use open-source software rely heavily on the community for updates, bug fixes and for providing interoperability checks. GPL-violations.org serves as an effective mechanism for keeping the community on one's side.

Nonetheless, there has been some litigation involving open source and, particularly, the GPL. The first case involved open-source developer MySQL, the makers of a widely used database software released under the GPL. MySQL brought an action against NuSphere—a software company that it believed was using its source code to produce proprietary software—something which contravenes the terms of the GPL.[117] This suit was filed in response to a claim by NuSphere against MySQL claiming 'breach of contract, tortious interference with third party contracts and relationships and unfair competition'.[118] MySQL's counter-claim was for 'trademark infringement, breach of the interim agreement, breach of the GPL license, and unfair and deceptive trade practices'.[119] This case was settled out of court[120] in an agreement that seems to have suited MySQL as NuSphere agreed to comply with the terms and conditions of the GPL.

The GPL has been the subject of three separate injunctions in German courts. Harald Welte is an open-source developer working for the netfilter/iptables team, which is software used in the Linux kernel. Welte is also one of the main supporters of GPL-violations.org. Part of his strategy has been to file complaints in German civil courts to help with enforcement strategies. The first injunction was filed in 2004 against network equipment manufacturer Sitecom.[121] Welte claimed that Sitecom offered a wireless network router which operated with firmware containing netfilter software

114 C Kelty, 'Trust among the Algorithms: Ownership, Identity, and the Collaborative Stewardship of Information', in R Ghosh (ed), *Code: Collaborative Ownership and the Digital Economy* (Cambridge, MA, MIT Press, 2005) 127–31.

115 http://gpl-violations.org/.

116 S O'Mahony, 'Guarding the Commons: How Community Managed Software Projects Protect their Work' (2003) 32(7) *Research Policy* 1179.

117 K Nikulainen, 'Open Source Software: Why Is it Here and Will it Stick Around?' (2004) 1(1) *SCRIPT-ed* 149, http://www.law.ed.ac.uk/ahrc/script-ed/docs/opensource.asp.

118 This, of course, lends credence to the previous discussion about contracts. A FAQ about the case can be found at http://www.mysql.com/news-and-events/news/article_75.html.

119 Ibid.

120 http://www.mysql.com/news-and-events/press-release/release_2002_14.html.

121 Landgericht Muenchen No 21 0 6123/04. An English version of the injunction can be found at http://www.jbb.de/judgment_dc_munich_gpl.pdf.

released under the GPL. Although Sitecom did not modify the software, it did not keep the copyright notices, and it 'closed' netfilter/iptables in that it ceased to offer it under the GPL. The claimants asked the court to order that the defendant ceased

> distributing and/or copying and/or making available to the public the software 'netfilter/iptables' without at the same time . . . making reference to the licensing under the GPL and attaching the license text of the GPL as well as making available the source code of the software 'netfilter/iptables' free of any license fee.[122]

The Munich District Court agreed with the claim and issued the injunction. This case is of tremendous importance for the validity of FOSS licences. Firstly, the Munich District Court recognised that the GPL is valid contract in accordance to German law. Moreover, it upheld the contractual validity of the main clauses, including the copyleft clause.[123]

The second injunction was issued in 2005 also by the Munich District Court in a case also brought by Welte.[124] The complaint was made against Fortinet, a manufacturer of firewall software. Welte and GPL-violations.org claimed that Fortinet was using Linux in its own code without releasing the modifications under the GPL. Interestingly, Welte was able to bring an action against Fortinet when another developer assigned code—called initrid—that was being used by the defendants. The injunction mirrors that issued in Sitecom. Eventually Fortinet settled out of court and announced its compliance with the terms and conditions of the GPL stating that:

> The source for the Linux Operating System Kernel and other GPL licensed components, including Fortinet's modifications, is available upon written request at the cost of CD copying and distribution. Additionally, Fortinet and its partners are providing written copies of the GPL license terms with all Fortinet product shipments.[125]

To conclude Welte's hat-trick, in 2006 he obtained another ruling against network hardware manufacturer D-Link for similar violations, this time in Frankfurt.[126]

While these three victories for the GPL have been widely publicised in FOSS circles, the litigation that made technology news headlines was that of *SCO v IBM*.[127] In March 2003 the SCO Group—a well-known software developer of Unix-related products—filed a lawsuit against IBM alleging that the company was infringing its intellectual property over the Unix kernel. SCO claimed that back in 1985, AT&T and IBM signed a contract to produce a version of Unix called AIX. In 1995, SCO purchased all of the

122 As cited by Höppner, above n 76.
123 For a more detailed analysis of the case, see Höppner, above n 122.
124 Landgericht Muenchen No 21 0 7240/05.
125 http://www.fortinet.com/news/pr/2005/pr042505_gpl.html.
126 Landgereicht Frankfurt No 2-6 0 224/06.
127 *Caldera Sys, Inc v Int'l Bus Machs Corp* (D Utah 2003) (No 03-CV-0294).

intellectual property related to Unix from AT&T, hence the infringement claims. SCO argued that they own part of the code for AIX used in the Linux kernel code included with all Linux distributions. As a result of this action, IBM countersued SCO, claiming that the company has been infringing its (IBM's) copyright and patents, and alleging that SCO was in violation of the GPL because they used and modified the Linux kernel licensed with the GPL.[128]

The case is still ongoing[129] despite the fact that most commentators seem to believe that SCO's claims are baseless. At the heart of the problem is that SCO has been unable to prove ownership of vital components of code used in Linux. Moreover, when disclosing code, it has been shown that the elements to which they laid claim had already been released under other FOSS licences, including the BSD.[130] At the time of writing SCO had filed for bankruptcy,[131] and experts agree that FOSS users 'have little to fear from this litigation because SCO will struggle in proving IBM did not have the right to contribute its derivative and independent code to Linux'.[132]

In addition to the litigation over contracts and copyright, a recent FOSS concerned competition law. In *Wallace v IBM*,[133] the Seventh Circuit Court of Appeals found that the GPL did not contravene US antitrust law. Software developer Daniel Wallace claimed that he wanted to compete against the Linux operating system by selling derivatives or writing an operating system from scratch, but that this was not possible because Linux is offered for free. According to Mr Wallace, the GPL is part of a conspiracy because it makes software free forever, and it is impossible to compete against free products. Mr Wallace clearly missed the point of the definition of free software outlined above. Free is not free as in beer, but free as in freedom. Mr Wallace lost the case in the first instance because he could not prove that he had suffered an antitrust injury. Judge Easterbrook stated:

> Many more people use Microsoft Windows, Apple OS X, or Sun Solaris than use Linux. IBM, which includes Linux with servers, sells mainframes and supercomputers that run proprietary operating systems. The number of proprietary operating systems is growing, not shrinking, so competition in this market continues quite apart from the fact that the GPL ensures the future availability of Linux and other Unix offshoots.[134]

128 KD Goettsch, 'SCO Group v IBM: The Future of Open-Source Software' [2003] *University of Illinois Journal of Law, Technology & Policy* 581.

129 The most recent developments in this case can be followed http://www.groklaw.net.

130 G Lehey, *SCO's Evidence of Copying between Linux and UnixWare* (2004), http://www.lemis.com/grog/SCO/code-comparison.html.

131 US Bankruptcy Court for the District of Delaware, 07-11337-KG The SCO Group, Inc.

132 A LaFontaine, 'Adventures in Software Licensing: SCO v IBM and the Future of the Open Source Model' (2005) 4(2) *Journal on Telecommunications & High Technology Law* 449.

133 *Daniel Wallace v IBM, Red Hat and Novell*, 467 F 3d 1104.

134 Ibid, 6.

All of the above litigation and enforcement strategies point to the conclusion that FOSS licences are legally valid, and that the movement rests on firm legal foundations.

C. FOSS in Practice

The various technical and legal considerations explored above are important, but beyond that, what is the relevance of open-source software in everyday life? After all, the intricacies of the various licences and underlying philosophies seem to be most relevant to specialist lawyers and 'Internet geekdom'. However, the significance of FOSS to the wider public makes it an important subject of legal study.

FOSS is playing an increasingly important role in mainstream software development. Even so, it is common to read about FOSS in terms of its opposition to proprietary and commercial software from many commentators including prominent advocates of free and open-source software.[135] This rather combative approach seems to be diminishing in relevance as FOSS develops and follows a philosophy of peer production and open distribution of code.[136] The key question, then, is as to whether it forms a viable system of creating computer programs.

It is difficult to measure the adoption of FOSS in the wider community. While Linux has not managed to take a part of the operating system market away from Microsoft Windows, FOSS has proven to be an excellent system for servers, with some projects, such as Firefox, having made their way into mainstream use.[137] The FOSS community seems vibrant and active with SourceForge hosting 257,594 separate projects on its site. A survey of open-source communities amongst more than 3,000 projects found 127006 identifiable individuals programming code.[138]

Software relies entirely on source code. One could say that computer code is the currency of the information age, as the writing of instructions takes expertise and time. Code, then, is a valuable commodity within the digital environment. Without code, computers are useless. FOSS is mostly a system for producing code in a cheap manner by harnessing the power of crowds of programmers who are willing to write and share it with the wider community through the use of open licences. If one accepts the idea

135 R Stallman, 'Why Software Should Not Have Owners', in J Gay (ed), *Free Software, Free Society: Selected Essays of Richard M Stallman* (Boston, GNU Press, 2002) 47–52.

136 For more about the concept of peer production, see Y Benkler, *The Wealth of Networks: How Social Production Transforms Markets and Freedom* (London, Yale University Press, 2006) 59–90.

137 S Weber, *The Success of Open Source* (Cambridge, MA, Harvard University Press,2004) 190–92.

138 R Ghosh and VV Prakash, 'The Orbiten Free Software Survey' (2000) 5(7) *First Monday*, http://firstmonday.org/issues/issue5_7/ghosh/index.htm.

of code as a commodity, FOSS is a runaway success. The latest version of the Debian operating system has 213 million SLOC, while the Linux kernel has 5.2 million SLOC. To put things into perspective with proprietary software, Windows XP has 29 million SLOC, Windows Vista has 50 million SLOC, and Mac OSX (based largely on open-source code) has 86 million SLOC.[139]

The commercial successes enjoyed by FOSS developers are a good indicator of its viability, a phenomenon reinforced by the enthusiasm with which public administration bodies around the world have embraced open source. The EU seems to be at the forefront of FOSS adoption. In 2002, a report for the European Commission recommended that, wherever possible, public administrations in the EU should fund software projects and purchase software that fulfilled certain characteristics compatible with open licences.[140]

As a result of this and other research projects,[141] the Commission adopted the European Public Licence (EUPL),[142] the latest addition to the expanding open-source licence portfolio. While the licence is initially intended to be used as the official release method for projects funded under the IDABC framework, it is also the first open-source licence with an officially sanctioned translation in the 23 official languages of the EU, making it particularly useful for public-sector administrations within Member States.

FLOSS has been adopted in the European public sphere as it moves towards e-governance and e-democracy, allowing cost-effective, stable and secure access to information technologies. The German federal government has signed an agreement with IBM to purchase computers for use in its offices that will have Linux installed, thus greatly reducing their costs and increasing security.[143] The Spanish region of Extremadura decided to move into open-source operating systems in public institutions and had a total of 100,000 computers running Linux by the end of 2003.[144]

Another area in which FOSS has proved important is in developing countries. A simple cost comparison shows the clear advantages of this type of technology for countries that do not have the resources to purchase expensive proprietary commercial software licences. For example, German

139 Figures from H Dahdah, 'Tanenbaum Outlines his Vision for a Grandma-proof OS' [2007] *Computer World*, http://www.computerworld.com.au/index.php/id;1942598204;pp;1; and G Robles, *Debian Counting* (2004), http://libresoft.dat.escet.urjc.es/debian-counting/.

140 R Ghosh et al, 'Free/Libre and Open Source Software: Survey and Study', CORDISS Report D18, (2002), part II B, 23–26.

141 Eg R Ghosh et al, *Study on the Economic impact of open source softwareon innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU*, Contract ENTR/04/112, http://ec.europa.eu/enterprise/ict/policy/doc/2006-11-20-flossimpact.pdf.

142 Full text and accompanying documents can be found at http://www.osor.eu/eupl.

143 'IBM signs Linux deal with Germany', BBC News, 3 June 2002, http://news.bbc.co.uk/1/hi/business/2023127.stm.

144 AE Cha, 'Europe's Microsoft Alternative', *Washington Post* 3 November 2002, A01.

Linux distributor SuSE calculates that the cost of proprietary licences for operating system and applications generally used for constructing a Windows-based web server would cost almost €6,000; this generally includes the licences for one system.[145] In contrast, a SuSE Linux distribution that contains all of those applications and can be installed in an unlimited number of systems would only cost €90, and many Linux distributions can even be downloaded directly from the Internet without cost.

One drawback is that free or low-cost Linux distributions come with no support. Support must be purchased at extra cost. In addition, some distributions such as Red Hat Enterprise Linux are offered at considerably higher prices than the free download ones. The additional costs often represent the payment for full support—and the package usually covers unlimited licences. A study by Forrester Research of 140 large firms in North America found that, even taking into consideration some of the more expensive Linux distributions, the cost of every server machine running Linux was 60 per cent less than a comparable server running Windows.[146] Others, however, have pointed out that the migration from an environment running proprietary software and operating systems into a FLOSS operating system is considerably more expensive than expected. For example, the local government in Munich commissioned a study to calculate the cost of migration from Windows to Linux. The study found that there was no noticeable difference in cost between migrating to Linux and migrating to a later proprietary Windows version. On the contrary, the study estimated that the migration might cost as much as €3000 per client in hardware, software and training.[147] Yet others have pointed to examples of cheap migrations.[148] For developing countries, and in particular the least-developed countries, migration would not generally be a problem as there are often no operating systems to migrate from.

Along with the cost benefits, FOSS has other characteristics that makes it valuable for developing countries. As has been stressed, source code is made available in all FOSS licences, which means that programmers can make changes to how the software works. This would give those in developing countries flexibility in adapting the software to their needs—particularly to make language ports.[149] Good examples of this are the many non-English Linux distributions, often in languages that would not warrant

[145] Figures taken from http://www.suse.com/en/private/products/suse_linux/prof/winprice.html.

[146] J Giera, 'The Costs and Risks of Open Source', Forrester Research Report (2004).

[147] Unilog Management, *Client Study for the State Capital Munich: Executive Summary of the LHM 2002* (2003), http://www.forget-me.net/Linux/free-software-study-munich.pdf.

[148] See eg R Benner, 'Migration from Windows to Linux Saves Thousands' [2004] *IT Manager's Journal*, http://www.itmanagersjournal.com/software/04/01/09/2231250.shtml.

[149] P Krakowski, 'ICT and Free Open Source Software in Developing Countries' (2006) 223 *IFIP International Federation for Information Processing* 319.

the interest of commercial developers.[150] Proprietary software, by contrast, is offered to the user as a block of sealed bits that cannot be changed. Even attempting to reverse engineer and decompile proprietary software could be considered unlawful in many jurisdictions.[151]

A further advantage for developing countries is the reliability and security of non-proprietary software when compared to proprietary software. Faulty, vulnerable or buggy software can cost considerable sums of money. For example, a survey of IT specialists by *CIO Magazine* found that companies spend 7–8 per cent of their computer-related budgets on security. Another report from 2001 calculates that faulty software costs companies in the US a staggering US$78 billion a year.[152]

However, the ultimate advantage of FLOSS for developing countries is that it offers a powerful tool to encourage the development of native technologies, enabling the move from imitation to innovation. True, there will be an initial need to copy and share source code originating from developed countries, but once this has been achieved, then indigenous innovation could take over. In the words of Wayne Marshall, a Unix programmer in Guinea: 'Open-source advocates can be sure that Africans get community; Africans get bazaar.'[153]

With these advantages, one should not be surprised that public institutions in developing nations are looking at non-proprietary software in a favourable manner. It has the potential to help these nations bridge the digital divide through enhancing their technological capability.

A good example of a country in which FOSS has been widely adopted is China, a country heavily involved in the development of indigenous tools for e-governance. The flexibility of the non-proprietary model can be seen in the development of a Chinese distribution of Linux called Red Flag Linux aimed at Chinese consumers.[154] Another version of Linux called Yangfan Linux (which means 'raise the sails') supported by the Chinese government is set to replace Windows and Unix on all computers and servers in the Chinese government.[155] A survey of Chinese software developers conducted by Evans Data Corporation found that about two-thirds of those developers are planning to write OSS-related applications in the next year, a figure that shows the extent to which this model is growing in

150 Eg Ubuntu Linux is distributed with support for 100 languages (they claim 'from Afrikaans to Zulu'). Other noteworthy distributions are Dreamlinux in Brazilian Portuguese; gnuLinEx in Spanish; and Asianux in Chinese, Korean and Japanese.

151 For more about the law of decompilation, see P Samuelson and S Scotchmer, 'The Law and Economics of Reverse Engineering' (2002) 111 *Yale Law Journal* 1575.

152 M Levinson, 'Let's Stop Wasting $78 Billion a Year', *CIO Magazine* 15 October 2001, http://www.cio.com/archive/101501/wasting.html.

153 W Marshall, 'Algorithms in Africa', *Linux Journal* 1 June 2001, http://www.linuxjournal.com/article.php?sid=4657.

154 http://www.redflag-linux.com/.

155 M Berger, 'inuxWorld Expo: Chinese Government Raises Linux Sail', *Infoworld* 13 August 2002, http://archive.infoworld.com/articles/hn/xml/02/08/13/020813hnchina.xml.

China.[156] India is another country where non-proprietary software is making strong advances. It was calculated that in January 2004, 10 per cent of all commercial computers sold in India contained Linux as their operating system.[157]

The eventual success of non-proprietary software in such populous countries as India and China could be the greatest encouragement for the use of this model in developing nations. The size of these markets alone would provide serious incentives for other countries to replicate the experiences in China and India—and if successful, it might even create a proprietary/non-proprietary divide.

VI. CONCLUSION

There can be little doubt that the software industry is still one of the powerhouses of the global economy despite the recent financial downturn. For example, in 2007 the global worldwide spending on software amounted to US$257 billion.[158] Given the economic importance of software, it is clear that any discussion about its legal protection is of the utmost interest to producers, consumers and regulators, who want a share of the growing demand for computer programs to be satisfied in their countries.

This chapter has described and analysed just a few of the many areas of relevance to FOSS development and distribution. It is common to see this topic treated in a shrill and partisan manner from both promoters and detractors of the movements. While it is clear that proprietary producers and commercial developers may see their share of the market reduced by FOSS, it would be a mistake to look at open source as a potential harm. There are many possibilities for the coexistence of the models despite the protestations of those at the fringes. As a personal anecdote, I am writing this on a Mac, a closed operating system that uses some open-source code, using both Microsoft Office and NeoOffice (an open source replacement for Microsoft Office), and have done some research on Firefox. Intransigence? The reality is that there is growing interaction between proprietary and non-proprietary systems.

The gradual acceptance of FOSS as a valid commercial strategy is shown by looking at how it has been adopted by the two largest software companies in the world, IBM and Microsoft, both of which are involved in a struggle for the profitable server market. IBM dominates the hardware

156 Evans Data Corporation. *Chinese Development Survey*, Vol 2 (2002), http://www.evansdata.com/n2/surveys/chinese_toc_02_2.shtml.

157 'Linux, Microsoft Face Off in India', Reuters, 11 August 2003, http://news.com.com/2100-1016_3-5062158.html.

158 Plunkett Research, *InfoTech Industry Overview* (2008), http://preview.tinyurl.com/5e2gwg.

market[159] while Microsoft is still ahead in software sales. IBM has thrown its considerable financial weight behind open source in order to dent Microsoft's software dominance. Back in 2000, IBM announced that it would make an unprecedented investment of US$1 billion in Linux.[160] Since then, IBM has become the biggest supporter of FOSS models as a valuable and profitable business model.[161] The fact that it still remains strong in the software market should serve as an indication that the strategy has been at least partially successful. Similarly, Microsoft has been shifting its attitude towards FOSS. While the software giant was initially opposed to open-source licensing and ideals,[162] it has been slowly moving towards a policy of peaceful coexistence, even releasing its own FOSS licences.[163] This has been quite a shift from the company that at one point had discussed in leaked documents the use of fear, uncertainty and doubt tactics in order to minimise the threat of Linux and FOSS to Microsoft's market share.[164] As with many other commercial strategies, it is not clear why Microsoft shifted its approach to FOSS. It may, however, have been motivated by IBM's success in harnessing the power of the FOSS community.

159 S Malik et al, 'IBM Servers, Worldwide, 2007', Gartner Research Report G00150814 (2007).

160 J Wilcox, 'IBM to spend $ 1 billion on Linux in 2001', *CNET News.com* 2000, http://news.cnet.com/news/0-1003-200-4111945.html.

161 J West, 'How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies' (2003) 32(7) *Research Policy* 1259.

162 See Guadamuz, above n 70.

163 Such as the OSI-approved Microsoft Public License (Ms-PL), see http://www.opensource.org/licenses/ms-pl.html.

164 These are known as the Halloween Documents. See http://edge-op.org/iowa/www.iowaconsumercase.org/011607/6000/PX06501.pdf.