

KEMP LITTLE LLP

OPEN SOURCE SOFTWARE: AN INTRODUCTION

TABLE OF CONTENTS

A. BACKGROUND	3
1. Introduction.....	3
2. Purpose and scope.....	4
B. OSS: THE HISTORICAL CONTEXT	5
3. Origins of the OSS movement	5
4. The FSF and the four freedoms.....	5
5. GNU, the GPL and Linux	5
6. The OSI and the OSD	5
7. OSS today	6
C. OSS: THE BUSINESS CONTEXT	7
8. Reasons for the rapid uptake of OSS	7
9. Reactions to OSS of established technology developers	7
10. Emergence of an OSS industry?	8
D. THE OSS LICENCES	9
11. Plethora of OSS licences.....	9
12. Feature range of OSS licences	10
13. The GNU General Public License (GPL)	10
14. The GNU Lesser GPL (LGPL)	12
15. Other common OSS licences	12
E. CURRENT OSS LEGAL ISSUES	13
Key GPL legal issues (1): copyleft, ‘contains’ and GPL v2	13
16. Introduction: GPLv2 Section 2(b).....	13
17. The operating system (‘OS’) and the software stack.....	14
18. Combining programs: compilation	14
19. Combining programs: linking, plug-ins and calls.....	15
20. Linux	15
21. Scripting: Java and JavaScript	16
22. Copyright and the GPL	17
23. GPL Section 2(b)	18
24. Other GPL points	19
25. Key GPL legal issues (2): ‘Tivoisation’	20
26. Key GPL legal issues (3): patents	20
27. Key GPL legal issues (4): software as a service	21
F. THE CASES: OSS IN THE COURTS.....	22
28. USA: the SCO-Linux cases.....	22
29. USA: Jacobsen v Katzer and Kamind Associates, Inc.....	22

30. USA: FSF v Cisco Systems, Inc.	23
31. Other US litigation	23
32. Germany: gpl-violations.org: Sitecom, Fortinet, D-Link and Skype.....	23
33. Informal interventions by the FSF	24
34. UK: FSF Europe and BT's Home Hub	24
G. GOVERNANCE: OSS INSIDE THE ORGANISATION.....	25
35. Introduction	25
The contractual matrix.....	25
29. Inbound transactions	25
37. Outbound transactions.....	25
The governance matrix	25
38. Governance: general	25
39. Governance: the policy context.....	26
40. Governance: the people context.....	27
41. Governance: the technical context	27
42. Governance: awareness and training.....	27

PANELS, CHARTS AND TABLES

1. The Open Source Definition.....	6
2. Top 20 Open Source Licenses (July 2009)	9
3. OSS Licence Types	10
4. GPL, Version 3: summary of key conditions.....	11

OPEN SOURCE SOFTWARE: AN INTRODUCTION¹

A. BACKGROUND

1. **Introduction.** A feature of the software world over the last ten years has been the rise and rise of Open Source Software ('OSS'). From its origins in US academia in the early 1970s², OSS emerged into the mainstream in the 1990s³ and has continued to become increasingly widely used in the 2000s. Looking ahead, its scope and appeal will almost certainly increase, due to a fairly unique combination of circumstances:

- **the Internet:** OSS modules are readily downloadable from sites like sourceforge.com. To that extent OSS is similar to other software delivery techniques that the Internet powers, like virtualisation, software oriented architecture, ('SOA') software as a service ('SaaS') and cloud computing⁴, all of which are gaining traction in the late 'noughties';
- **the current generational shift in the software industry:** the announcement in July 2009 that Google is developing its Chrome browser into a full operating system⁵ most recently demonstrates the generational shift from the traditional 'software as a licence' – on the PC at home or in the server room at the office – towards remote, service based computing embracing these Internet enabled delivery techniques. This shift is another spur for OSS;
- **the onset of adverse economic circumstances:** The appeal of OSS has been further heightened with the onset of adverse economic conditions as it increases competitive pressures to reduce costs and innovate. This in turn has sharpened the perceptions that OSS benefits from lower costs than traditional proprietary software⁶ and from a distinct 'coolness' factor in the development world - the idea that the 'eco community' (or 'bazaar') approach to software development is at least as innovative as more structured 'cathedrals';
- **'tipping point' acceptance:** finally, OSS has reached a tipping point. In the private sector, IT research consultant Gartner in its November 2008 survey of 300 OSS users found that 85% of them were using OSS and that the remaining 15% all had plans to do so within the year⁷; and in the public sector, HMG has committed to accelerate use of OSS⁸.

¹ By Richard Kemp, Partner, Kemp Little LLP, London.

² A good article on how OSS became mainstream is 'Open Source Software Monetized: Out of the Bazaar and into Big Business' (Mahony and Naughton, The Computer & Internet Lawyer, vol. 21. no. 10, October 2004). A good description of the OSS movement's history is 'A Primer on Open Source Licensing Legal Issues: Copyright, Copyleft and Copyfuture' (Dennis Kennedy, <http://www.denniskennedy.com/opensourcedmk.pdf>).

³ E.g. leading Linux developer Red Hat Software released Red Hat Linux in 1995, IPO'd in August 1999, was added to the NASDAQ 100 in 2005 and moved to the New York Stock Exchange in late 2006.

⁴ http://www.kemplittle.com/PDFs/HotTopicArticle_CloudComputing_Feb2009.pdf

⁵ Chromium is the OSS project behind Google Chrome, with the parts of it written by Google released under the BSD licence and other parts subject to a number of different permissive OSS licences (source: http://en.wikipedia.org/wiki/Google_Chrome).

⁶ Although on a TCO (total cost of ownership) comparison basis the balance may not be so clear cut in favour of OSS.

⁷ Out of that sample of 300 users, Gartner estimated that about 25% of all the software they were using was OSS. But of those 300 companies, about 70% did not at the time of the survey have a corporate policy framework in place to support their use of OSS (source: <http://www.gartner.com/it/page.jsp?id=801412>).

⁸ See e.g. HMG's 10 point action plan published on 25 February 2009 (source: <http://www.ft.com/cms/s/0/8379a24c-0377-11de-b405-000077b07658.html>).

2. **Purpose and scope.** This purpose of this practice note is to serve as an introduction from the legal perspective to current OSS issues. OSS is an increasingly broad subject. Much has been written about it⁹ and much of that is available on the internet. A number of links have been provided in tables and footnotes to some of the publicly available material.

Sections B and **C** briefly consider OSS in its historical and business contexts.

In essence, OSS is software provided under licence granting certain freedoms to a licensee and should properly be seen as a range of associated licensing techniques. There are many different types of OSS licences differing widely in clarity, length and legal effect, and **Sections D, E** and **F** overview key licences, a number of the topical knottier OSS legal issues, and such case law as there is: a striking feature of OSS law is its lack of case law, adding complexity and uncertainty to interpretation and analysis.

Section G takes a practical look at OSS inside the organisation and governance from the perspectives of inbound and outbound contracts and the growing importance of compliance and governance.

⁹ See e.g.: 'Open Source Licensing, Software Freedom and Intellectual Property Law' (Prentice Hall, 2004) by Lawrence Rosen, a past General Counsel of the Open Software Initiative; 'Understanding Open Source and Free Software Licensing (O'Reilly, 2004); Open Source for the Enterprise (O'Reilly, 2004); 'The Open Source Alternative: Understanding Risks and Leveraging Opportunities (Wiley & Sons, 2008) by Heather J Meeker. The website of the Software Freedom Law Centre (SFLC) has a publications section containing legal analyses of a number of OSS issues (<http://www.softwarefreedom.org/resources/>).

B. OSS: THE HISTORICAL CONTEXT

3. **Origins of the OSS movement.** The origins of the OSS movement can be traced back to a cultural attitude prevalent in US academic circles of opposition to the restrictive nature of exclusive rights under intellectual property laws, itself an echo of the counter culture of the 1960s. It was particularly antithetical to the commercial exploitation in the 1970s and 80s of the operating system UNIX developed by AT&T employees at Bell Laboratories.

4. **The FSF and the four freedoms.** In 1985 Richard Stallman¹⁰, an ex-MIT academic, established the Free Software Foundation¹¹ ('FSF') as a non-profit body dedicated to the development of Free Software. The FSF would oversee the GNU Project, hold the copyright in the software created for it and enforce the licences. To be considered Free Software, the FSF identified four essential freedoms for any software licensee¹²:

- to run the software for any purpose;
- to study how the software works and adapt it;
- to redistribute copies of the software; and
- to improve the software, and release those improvements.

5. **GNU, the GPL and Linux.** In 1983 Mr Stallman had announced the GNU¹³ Project, a mass collaboration to create free software whose first goal was to create a full operating system as a UNIX replacement. The licence adopted for the GNU Project's software was the GPL – the GNU General Public License ('GPL'). By 1992 all necessary components had been completed except the operating system kernel, then known as GNU Hurd. In 1992, this gap was filled when the GNU software was combined with a new kernel called Linux¹⁴ to create a complete operating system. This combination, known as GNU/Linux, was licensed under the GPL.

6. **The OSI and the OSD.** By the late 1990s, some parts of the OSS community considered that the anti-IP sentiments of Mr Stallman and others were inhibiting the widespread take up of OSS. In 1998 Bruce Perens¹⁵ and Eric Raymond¹⁶, leading members in the OSS movement, established the Open Source Initiative¹⁷ ('OSI') to promote OSS on pragmatic rather than ethical or philosophical grounds¹⁸. The OSI is the steward of the Open Source Definition ('OSD'¹⁹) and part of its function

¹⁰ Personal home page at <http://www.stallman.org/>

¹¹ www.fsf.org

¹² The second and fourth freedoms imply access to the source code.

¹³ A recursive acronym for 'GNU's Not Unix'.

¹⁴ Linux was developed by Linus Torvalds, a computer sciences specialist at Helsinki University. Incorporating elements from MINIX ('Minimal UNIX'), a UNIX related precursor OSS operating system, Mr Torvalds and his collaborators coded a completely new operating system kernel known as Linux ('Linus's 'MINIX'). Linux was first publicly released in September 1991.

¹⁵ Personal home page at www.perens.com

¹⁶ Personal home page at <http://www.catb.org/~esr/>

¹⁷ <http://www.opensource.org/>

¹⁸ The OSI's views were memorably advocated in Mr Raymond's essay 'the Cathedral and the Bazaar' where the cathedral represents the structured world of proprietary software development and the bazaar the looser communitarian approach (<http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/>).

is to review and approve licences conforming to the OSD. Many different licences satisfy the OSD, but the types of obligations they impose can vary quite widely. The OSD has developed ten criteria to determine whether a licence for software is open source (see figure 1).

Figure 1 - The Open Source Definition

1. **Free redistribution:** software to be available for redistribution without payment.
2. **Source code:** software to be distributed with the source code or well-publicised access to it.
3. **Derived works:** license to allow modification of the software and distribution of resulting derived works.
4. **Integrity of author's source code:** distribution of "patch files" used to recreate the derived work (rather than full source code) to be permitted.
5. **No discrimination against persons or groups.**
6. **No discrimination against fields of endeavour:** for example, limiting to non-commercial purposes is not permitted.
7. **Distribution of license:** must be no need to execute extra licences for redistributed software.
8. **License must not be product specific:** licence rights not to depend on the software being distributed with other specified software.
9. **License must not restrict other software:** The license must not place restrictions on software distributed together with the licensed software.
10. **License must be technology neutral.**

7. **OSS today.** A number of increasingly important groups have recently developed with the aim of evangelising compliant OSS use. These include:

- the Software Freedom Law Center ('**SFLC**'²⁰), which provides legal representation and other law-related services to protect and advance OSS. It is chaired by Eben Moglen, former General Counsel of the FSF;
- gpl-violations.org²¹, which seeks to raise public awareness about past and present infringing use of GPL software; and
- regional groupings of the FSF, including the Free Software Foundation Europe ('**FSFE**'²²) and
- the Freedom Task Force ('**FTF**'²³).

¹⁹ The full definition is at <http://opensource.org/docs/osd>. The OSD was written by Mr Perrens, OSI co-founder, and is based on his earlier work for the Debian Project – a project to create and maintain a free operating system based on GNU/Linux.

²⁰ <http://www.softwarefreedom.org/>

²¹ <http://gpl-violations.org/>

²² <http://www.fsfe.org/index.en.html>

²³ <http://www.fsfe.org/projects/ftf/ftf.en.html>

C. OSS: THE BUSINESS CONTEXT

8. **Reasons for the rapid uptake of OSS.** There are many reasons for the OSS model's popularity and, as mentioned at the start, it currently benefits from a unique combination of circumstances. For many projects, a sort of OSS eco-system develops around a community of programmers who seek constantly to improve the code they work with and to provide bug fixes and additional functionality free of charge. Major OSS projects usually employ rigorous peer review mechanisms to marshal all contributions into a coherent, consistent and stable product. As a result, code adopted for an OSS product can be of at least as high a standard as software produced commercially. With source code readily available, OSS can be adapted to work on new hardware as and when available. Developers and users alike know that the OSS product will not become obsolete simply through the obsolescence of the original hardware platform (as might happen to proprietary software if maintaining a version for the obsolete platform was no longer commercially attractive).

Competitive pressures increasingly take time out of the product cycle – the period between the time when design starts and first customer availability. Using pre-made OSS components (particularly for routine, lower level tasks) shortens the development phase and frees up internal resources to develop higher level software that confers competitive advantage. This trend applies across most industry sectors where technology transforms business and is becoming particularly noticeable in the consumer electronics sector²⁴.

Finally, in developing countries, the introduction of OSS can be used to establish a local software industry and as a result OSS is becoming widely adopted by governments in countries such as Brazil, Russia, India, China and South Africa. Customisation, integration and support services for software packages can be provided locally if source code is accessible.

9. **Reactions to OSS of established technology developers.** Faced with the rise of OSS, the responses of established technology vendors have differed. Software-only developers initially showed more hostility than vendors who could leverage hardware sales out of their OSS investment. In recent years, increasing OSS mainstream use and 'monetisation'²⁵ of OSS has resulted in edgy detente if not rapprochement, although tensions remain under the surface, especially relating to patents.

For example, Microsoft is encouraging OSS vendors to enter into patent co-operation agreements and in November 2006 it entered into an agreement with Novell under which it promised not to assert its patent rights against customers purchasing SuSe Linux from Novell²⁶. In mid 2007 it

²⁴ E.g. mobile handsets: 'The Open Source Cellphone: Lead Users and Early Adopters in the Mobile Phone Industry', Anmol Madan, Media Lab, MIT, 2006 (<http://web.media.mit.edu/~anmol/351-madan-1.pdf>)

²⁵ See Mahony and Naughton cited at footnote 2 above.

²⁶ <http://www.microsoft.com/presspass/exec/steve/2006/11-02NovellInterop.msp>. Microsoft would also pay Novell for 350,000 "subscription coupons" for SuSe Linux which it could sell to its customers. Coupons could be redeemed from Novell for single- or multi-year subscriptions, upgrades and technical support.

entered into patent co-operation agreements with two more Linux distributors, Xandros²⁷ and Linspire²⁸.

In March 1999, IBM²⁹ established a Linux Technology Centre to work on developing OSS which now employs several hundreds of Linux kernel developers. IBM has one of the largest technology patent portfolios in the world and in 2004 pledged not to use its patents against Linux.

Sun Microsystems³⁰ is a significant proponent of OSS and arguably the largest corporate contributor to the Open Source movement. The OpenOffice.org³¹ project was created after Sun acquired the German software company StarDivision in 1999 and StarOffice, StarDivision's office productivity suite, was released as OSS in 2000. In 2005 Sun open sourced its flagship operating system with the OpenSolaris project and it continues to employ hundreds of software engineers to work on OpenSolaris. Since November 2006, Sun has been moving towards licensing the components of its Java platform under the GPL

In addition, a consortium of companies formed the Open Invention Network³² (OIN) to defend the OSS movement from patent infringement actions. OIN members include major technology companies such as IBM, Google, NEC, Novell, Philips, Red Hat, and Sony. Each agrees not to use its Linux-related patents against each other. The OIN has also begun purchasing relevant patents and provides free licences to all its members.

10. Emergence of an OSS industry? The software world is currently seeing the rise of a significant number of increasingly successful companies whose revenues are based on OSS. Red Hat, the leading Linux developer, announced sales of \$175m in its (at the time of writing) most recent quarter (to 31 May 2009) and with around 3,000 employees ranks under Novell (approx 4,000 employees) and Sun Microsystems (approx 30,000, pre-Oracle merger) as the largest OSS companies; and other Linux companies like Canonical, eRacks, Mandriva, Plat'Home and Xandros are becoming increasingly successful in the current climate. Other OSS companies we are likely to hear more from in the coming months include Alfresco, Digium, Fonality, Jaspersoft, MuleSource, Opengear, xTupe and Zimbra³³.

²⁷ <http://www.microsoft.com/presspass/press/2007/jun07/06-04XandrosPR.msp>

²⁸ <http://www.microsoft.com/presspass/press/2007/jun07/06-13LinspirePR.msp>

²⁹ <http://www.ibm.com/developerworks/opensource/>

³⁰ <http://www.sun.com/software/opensource/>. On 20 April 2009, Sun and Oracle Corporation announced that Oracle would acquire Sun in a transaction valued at approximately \$7.4bn see <http://www.sun.com/third-party/global/oracle/>.

³¹ <http://www.openoffice.org/>

³² <http://www.openinventionnetwork.com/>

³³ See e.g. the 'Open Source 50' at <http://www.thevarguy.com/the-open-source-50/>.

D. THE OSS LICENCES

11. **Plethora of OSS licences.** Today, there are many hundreds of OSS licences in use, varying widely in length, clarity, intent and legal effect, and ranging from the intrusive, ‘copyleft’ GPL through to short licences containing virtually no express terms. As a practical matter, a good start point is to identify the OSS concerned and the licence terms under which it is made available and then to assess whether the licence attaches any particular terms. A leading OSS service provider³⁴ provides a regularly updated table of the top 20 OSS licences in use and their estimated popularity. Figure two shows this table as at July 2009.

Figure 2 – Top 20 Open Source Licenses (July 2009)		
Rank	Licence (click on licence name to link to the licence text)	% use
1.	GNU General Public License (GPL) 2.0	50.01
2.	GNU Lesser General Public License (LGPL) 2.1	9.62
3.	Artistic License (Perl)	8.69
4.	Berkeley Software Distribution (BSD) License	6.32
5.	GNU General Public License (GPL) 3.0	5.14
6.	Apache License 2.0	3.92
7.	Massachusetts Institute of Technology (MIT) License	3.83
8.	Code Project Open 1.02 License	3.34
9.	Mozilla Public License (MPL) 1.1	1.25
10.	Microsoft Public License (Ms-PL)	1.02
11.	Common Public License (CPL)	0.59
12.	zlib/libpng License	0.46
13.	Eclipse Public License (EPL)	0.45
14.	Academic Free License	0.42
15.	GNU Lesser General Public License (LGPL) 3.0	0.41
16.	Open Software License (OSL)	0.34
17.	Mozilla Public License (MPL) 1.0	0.28
18.	Common Development and Distribution License (CDDL)	0.26
19.	PHP License 3.0	0.26
20.	Ruby License	0.24

Three things stand out particularly from the table:

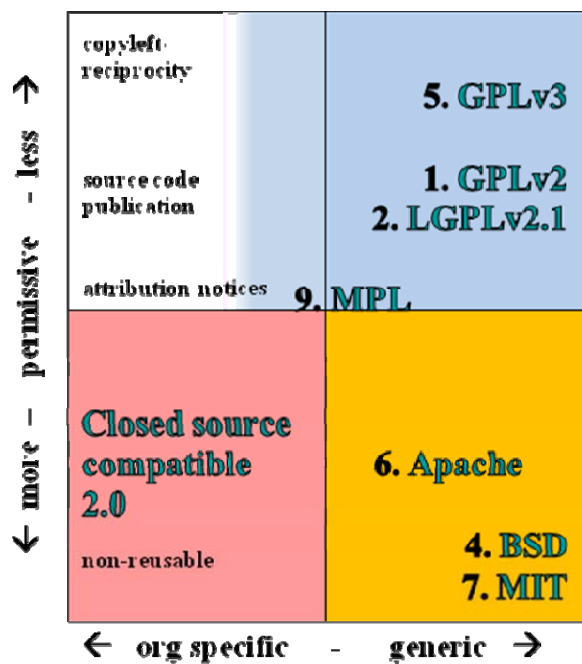
³⁴ Black Duck Software – see <http://www.blackducksoftware.com/oss>.

- first, the prevalence of the GPL which accounts for about half the OSS world. These are the FSF licences with the more zealous ‘copyleft’ terms. The four GPL licences together account for 65% of the total OSS licence world, with GPL version 2 (‘**GPLv2**’) alone accounting for 50%; its cousin - aimed at software libraries – the Lesser GPL version 2 (‘**LGPLv2**’), 10%; and the GPL version 3 (‘**GPLv3**’) and **LGPLv3** (both published in Jun2 2007), 5% and 0.5% respectively;
- secondly, and next in popularity, are the academic licences accounting for around 10%. These include the more liberal and permissive BSD and MIT licences, each also in the top 10; and
- thirdly, the top 4 licences account for 80% of the total, and the top 10 licences over 90%.

12. **Feature range of OSS licences.** The chart at figure 3 borrows from the table at figure 2 to plot restrictiveness/permissiveness of licence (on the y axis) against specificity of licence (on the x axis) to give an ‘at a glance’ summary of the range of OSS licence features:

- in the top right hand corner (in blue) are the GPL and LGPL – the least permissive, most generic licences;
- in the bottom left hand corner (in pink) are the so called ‘closed source’ compatible licences (licences that are issued by a developer and specific to it for its proprietary software) – more permissive, but organisation specific and non-generic;
- in the bottom right corner (orange) are the Apache, BSD and MIT licences - permissive and generic; and
- in the middle sits the MIT licence.

Figure 3 – OSS Licence Types



13. **The GNU General Public License (GPL).** The GPL was written by Richard Stallman and the FSF in 1989. Since then its underlying philosophy and the skilful way in which it turns

intellectual property law against itself have made it popular. As the licence accounting for around half the OSS in use, it is the licence for the GNU Project, GNU/Linux, much software on the Java platform and for programmers subscribing to the FSF's Free Software ethic.

The GPL protects the FSF's four essential freedoms identified at paragraph 4 above. The legally radical mechanism to achieve this and the thing for which the GPL is best known was called 'copyleft' by the FSF. Copyleft is the idea that the freedoms guaranteed by the GPL would also apply to new works derived from the original GPL-licensed software. Copyleft would serve to expand the rights of the public, in contrast to the traditional role of copyright which grants exclusive rights to the author of new software. The scope and extent of copyleft and GPLv2 Section 2(b), its transmission mechanism, are overviewed in paragraphs 16 to 24 of Section E below.

Version 1³⁵ of the GPL was issued in 1989. Version 2³⁶ (GPLv2), the licence Linus Torvalds chose to apply to the Linux kernel, was issued in 1991. Version 3³⁷ (GPLv3) was published in its final form and adopted on 29 June 2007.

Figure 4 – GPL version 3: summary of key conditions

Section 0 – definitions: GPL v3 uses the terms “propagate” and “convey” instead of the term “distribute” used in GPLv2. “Distribute” has different specific meanings in different jurisdictions.

Section 1 - source code: establishes the “corresponding source” for a program in object code form as including source code but also scripts for controlling compilation and installation of object code.

Section 2 - basic permissions: clarifies that the rights under the licence are perpetual and irrevocable whilst the GPL conditions are met. The program may be made, run and propagated without regard to the GPL conditions, as long as the work is not conveyed. “Conveying” means enabling other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

Section 3 - protecting users' legal rights from anti-circumvention law: prohibits using any program released under the GPL as a “technological protection measure” as defined by the US Digital Millennium Copyright Act or similar laws.

Section 4 - conveying verbatim copies: clarifies that a party may charge money for each copy it conveys and may offer support or warranty protection for a fee.

Section 5 - conveying modified source versions: conveying "a work based on the program" or "the modifications to produce it from the program, in the form of source code" requires that certain conditions to be met, including licensing the entire work under the GPL.

Section 6 - conveying non-source forms: specifies how source code should be provided.

Section 7 - additional terms: deals with additional terms that may be used to supplement the GPL terms (to permit wider compatibility with other OSS licences).

³⁵ <http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt>

³⁶ <http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>

³⁷ <http://www.fsf.org/licensing/licenses/gpl.html>

Section 8 – termination: confirms there is no right to use the program except by the licence, and that failure to comply with its conditions will result in termination of the licensee’s rights. Termination under this section will not affect the licences of parties who have received program copies under it. It also introduces a process for reinstating the licence on cessation of violations.

Section 9 - acceptance not required for having/running copies: but copying, distributing or modifying the program indicates acceptance of the licence (without the rights granted by the GPL these actions would constitute copyright infringement).

Section 10 - automatic licensing of downstream recipients: clarifies that downstream recipients of the program are automatically licensed by the original licensors. No right to sublicense.

Section 11 – patents: (see section E, paragraph 26 below).

Section 12 - no surrender of others' freedom: states that conditions imposed by a third party contradicting the GPL do not excuse the licensee from meeting its conditions.

Section 13 - use with the GNU Affero General Public License (‘AGPL’): permits the linking or combination of code licensed under the GPL with code licensed under the AGPL, which relates to networked use of software.

Section 14 - revised versions of this licence: clarifies that some versions of a program may be covered by one or more versions of the GPL.

Sections 15 and 16 – broad disclaimer of warranty and limitation of liability: these provisions also recognise that a warranty or liability may be accepted in return for a fee.

14. **The GNU Lesser GPL (LGPL).** The LGPL³⁸ is the second of the FSF’s two main licences and, as the name suggests, is less intrusive than the GPL. It is frequently used for software libraries that will be utilised by proprietary programs.

15. **Other common OSS licences.** The OSI publishes the licences that it has approved as conforming with the OSD on its website at <http://www.opensource.org/licenses/alphabetical>. At July 2009, 67 licences were listed. In contrast to the lengthy and technically complex GPL, there are licenses which are relatively short, very permissive and which impose few, if any, conditions. Two of the most commonly used are the academic licences, MIT and BSD. The MIT License³⁹ is little more than a copyright notice with a broad permission to deal (and sub-license) the software and a broad warranty and liability disclaimer. The BSD License⁴⁰ also consists of a copyright notice, a broad permission to redistribute and use the software (with or without modification) and a warranty and liability disclaimer and also adds three conditions: first, that redistributions of source code retain the licence terms; secondly, that redistributions of object code include the licence terms in accompanying documentation or other materials; and thirdly, that the name of the licensor should not be used to endorse or promote products derived from the software without permission.

³⁸ <http://www.gnu.org/licenses/lgpl.html>

³⁹ <http://www.opensource.org/licenses/mit-license.php>

⁴⁰ <http://www.opensource.org/licenses/bsd-license.php>

E. CURRENT OSS LEGAL ISSUES

Key GPL legal issues (1): the limits of copyleft – derivative works, ‘contains’ and GPLv2.

16. **Introduction: GPLv2 Section 2(b).** Section 2(b) of GPLv2 remains the subject of debate and a source of confusion. It sits at the heart of the copyleft mechanism and reads as follows:

“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 ...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*⁴¹ 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.”

Essentially the ‘viral’ mechanism by which the four freedoms set out at paragraph 4 above are propagated, Section 2(b) provides that any code which is your ‘work’ (let’s say, ‘**A**’) that ‘contains’ or is ‘derived from’ ‘the Program’ – i.e the GPL code (‘**B**’) - must be licensed under the GPL: colloquially, ‘your work’ (**A**) is ‘GPL’d’. Where your ‘work’ (**A**) is software you’d rather keep secret – for security reasons or as conferring competitive advantage, for example - the tension between that objective and the GPL terms in circumstances where **A** could be said to ‘contain’ or be ‘derived from’ **B** is evident because GPL compliance means publishing and disclosing the source code to **A**, your ‘work’.

This underlying tension is amplified in three ways, one technical, two legal. The first legal tension arises because in order to ensure compliance with ‘copyleft’ the guardians of the OSS world take a fairly aggressive approach to interpreting both what is a ‘derivative work’ under US copyright law and what ‘contains’ means under Section 2(b). The technical tension arises from the complexity, as a matter of computer science, of the nature of the very granular processes by which (in the example) **A** interoperates (works together) with, and so may be argued to ‘contain’ or be ‘derived from’, **B**. The third area of tension is lack of legal certainty: GPLv2 was published almost twenty years ago and there is as yet no case law as to the meaning of Section 2(b).

The technical and legal debates on the limits of GPLv2 Section 2(b) copyleft are extensive and unresolved. This note does not go into in detail⁴². Briefly, the following paragraphs outline:

- by way of technical background the operating system and software stack and how software is compiled and interoperates with other software (**paragraphs 17 to 19**);
- as practical examples relevant to Section 2(b), how Linux and Java work (**20 and 21**);
- relevant generally applicable copyright law principles behind Section 2(b) (**22**);
- how Section 2(b) may then be interpreted to apply, especially for Linux and Java (**23**); and
- a number of additional GPL points relevant to the analysis (**24**).

⁴¹ Emphasis added

⁴² For two good discussions, see Cap. 14 (‘the Border Dispute of GPL2’) of ‘The Open Source Alternative’, Meeker, cited at footnote 9 above; and ‘Dangerous Liaisons – Software Combinations as Derivative Works? Distribution, Installation and Execution of Linked Programs under Copyright Law, Commercial Licenses and the GPL’ by Prof. Dr. Lothar Detterman (Berkeley Technology Law Journal 2006, 1421)

17. **The operating system ('OS') and the software stack.** In order to put the Section 2(b) debate into its proper technical context, it will help briefly and generally to describe relevant basics of how software operates and is compiled and works together.

The software⁴³ at the heart of a computer is its OS. The foundation of the OS is the **kernel**. The kernel performs the roles of 'authorised person' with direct hardware access and 'traffic cop' directing operations in order to allow the different software components to work together and exchange information to perform tasks. Visualising the software as a stack and working upwards:

- the **kernel** is at the bottom of the stack;
- above the OS kernel sits the rest of the OS:
 - **utilities**: software tools designed to help the kernel manage by performing specific tasks;
 - **drivers** that work the hardware devices (screen, printer, graphics, etc);
 - **interfaces**: colloquially known as 'hooks and handles', interfaces provide specifications (sets of standard rules or protocols) that enable other software to work with the kernel⁴⁴;
- **libraries**: large blocks of pre-written, reusable code to perform common tasks that other, independent programs can link to and share. Libraries simplify the programmer's job and avoid reinvention of the wheel. They sit between the OS and applications; and
- **applications** – software that performs the user's instructions – is at the top of the stack.

As 'authorised person', the kernel has direct, privileged access to and control over the computer's hardware resources. These resources include the computer's memory space (a precious resource), processor, storage and hardware devices. Memory is partitioned into **kernel space** for the kernel and some utilities and drivers, and **user space**, generally for applications and libraries. Interfaces generally sit between user space and kernel space. Software in user space cannot access the hardware resources directly. It has to access them indirectly through kernel space by **system calls**, requests to the kernel to perform particular services picked from a menu.

As 'traffic cop', the kernel directs how the computer's memory and processor resources are most efficiently accessed, allocated and used and how the output from the processes carried out is displayed to the user. The kernel will determine how much and the order in which processing time is allocated to each program and manage memory space by sharing it among system components and other programs that use the kernel's services.

18. **Combining programs: compilation.** Computer software is initially written in **source code**, a set of instructions, and declarations written as a text file in a human readable computer programming language. A Computer does not read source code. In order to run (or execute) on the computer, source code must be converted into **executable code**, the 0s and 1s that the computer can read (also known as binary, machine or object code). The conversion process in which high level source code is turned into lower level executable code is known as **compilation**.

⁴³ 'Software', 'programs' and 'code' are used here interchangeably.

⁴⁴ Application programming interfaces (APIs) are interfaces in the source code that an operating system (or library or other application) provides in order to be able to respond to requests for service from other applications. Interface specifications also permit other OS software like drivers to work with the kernel.

Compilation is carried out by utility software known as the compiler. The compiler first reads in and analyses the source code and translates it into object code (at what is known as '**compile time**', which happens once). It then combines or links all the resulting compiled object code together efficiently to form a single '**executable**' capable of performing the user's instructions (known as '**link time**', which can happen once or many times, depending on how linking occurs). The time when the executable actually executes and performs the user's instructions is called '**run time**' (which happens many times).

19. **Combining programs: linking, plug-ins and calls.** Different programs can be combined to enable interoperation between them to take place in different ways, of which the most common include:

- **static linking:** where the compiler is instructed to bind code including code from a library permanently into the combined compiled code that makes up the executable. In static linking, all the relevant object code of all the software components is copied verbatim;
- **dynamic linking:** where, instead of copying all the object code verbatim, the code remains separate but the executable draws from the library when it is running by just copying certain references to it. As a result, dynamically linked executables are smaller than statically linked executables and so more efficient in terms of using precious memory;
- **plug-ins:** a plug-in is software conforming to an interface specification (like a printer driver) that a kernel as it is executing (i.e. in run time) loads when needed to perform a particular task (printing) and unloads when no longer needed. Plug-ins include loadable kernel modules ('**LKMs**') and allow for efficient use of precious memory;
- **system calls:** the mechanism by which an application requests the kernel to perform a service or task which the application does not have permission to perform directly.

20. **Linux**⁴⁵. Linux is the kernel of the GNU/Linux OS and is licensed under GPLv2. Arguably the most prominent OSS product, improving security, stability and functionality have seen its use increase significantly in recent years. Each particular version of Linux, which may run to several 10s of millions of lines of code, is known as a 'Linux distribution', a collaborative, remotely managed community project driven by developers and users. Popular Linux distributions are published by Canonical (as Ubuntu Linux), Red Hat (Fedora Linux and Red Hat Linux), Novell (SuSe Linux) and Mandriva (Mandriva Linux).

Linux was designed primarily for the desktop computer but is also available for a wide range of other systems, including servers (where it is estimated to hold around 15% of the server operating system market) and particularly embedded devices like smartphones (where its market share was estimated at 17% in 2006) and network firewalls and routers.

The way Linux operates has been the prime source of debate on the scope of GPLv2 Section 2(b). Linux works similarly to the ways described at paragraphs 18 and 19 and the Linux kernel can combine with:

- utilities and libraries by static linking at compile time and link time;

⁴⁵ See also paragraph 5 above.

- libraries by dynamic linking at run time;
- LKMs (the plug-ins like a printer driver) by loading/unloading as needed; and
- applications by system calls at run time.

Central to the Section 2(b) debate are the questions whether an executable resulting from one or more of these ways of combining code ‘*contains or is derived from*’ GPL code within Linux.

21. **Scripting: Java and JavaScript.** The Java platform is a portmanteau term for a range of software products and specifications developed by Sun Microsystems that are platform independent (able to run on different OS’s and system architectures independently). Most of Java is licensed under the GPLv2 and this area is the second source of debate on the scope of GPLv2 Section 2(b).

The key components of Java are the Java programming language (normally referred to as ‘**Java**’ on its own) which is used to write code that is compiled into **Java bytecode**, an intermediate language between the (high level) Java source code and (low level) object code. Java bytecode is compiled and executed as a **Java applet** on a Java Virtual Machine (‘**JVM**’,⁴⁶). Java applets dynamically link at run time to libraries within the JVM known as Java class libraries, or **.class** files. For ease of distribution, libraries as multiple class files are packaged together as Java archive (**.jar**) files. Java applets run on individual PCs. The Java platform also extends to servers in the form of **Java servlets** (which are to the server what Java applets are to the PC) and Java Server Pages (‘**JSPs**’). Java supports a process known as **inheritance**, in which different .jar files can inherit common functions or behaviour from other classes⁴⁷.

Java is important for **scripting** in creating dynamic web pages, where sets of requests or commands are passed to an application⁴⁸ to make it perform certain tasks. In networking (client server computing), scripting can be either client-side or server-side. Java applets are examples of client-side scripts, which are run by and within the web browser on the client computer.

JavaScript (confusingly, only distantly related to Java) is another example of a client-side scripting language that web designers use - for example, to specify that a credit card number entered in a form on the screen has the requisite numbers – that is also published under GPLv2.

Server-side scripts run requests or commands from the user as scripts directly on the server, normally a web server, to generate tailored Internet (HTML) pages. Server-side scripting is used in interactive websites operating databases where the parameters for queries (questions) to the database are extensive and the resulting web pages can be highly customised.

Section 2(b) issues in the area of scripting arise for Java and JavaScript in the following ways. Much of Java is now released under GPLv2, and the FSF world believes that Java applets and servlets running in the browser or on the web server ‘*contain or are derived from*’ GPL code (here,

⁴⁶ The JVM can be hosted by (reside on) any OS or system architecture, and that is how platform independence is achieved.

⁴⁷ An example of inheritance in Java (at <http://java.sun.com/docs/books/tutorial/java/concepts/inheritance.html>) is mountain bikes, road bikes and tandems as classes of bicycle where Java allows each class to have one direct superclass (bicycle) and each superclass to have an unlimited number of subclasses.

⁴⁸ Scripts are at one remove from applications since, unlike applications, they cannot make system calls directly on an OS: scripts pass commands to an application which then make system calls on the OS.

underlying elements of the Java platform, including use of .jar files with inheritance); and that Java Script programs similarly contain or are derived from the JavaScript language as a GPL work. Accordingly, in each case the source code for the applets, servlets and programs should be published.

22. **Copyright and the GPL.** Most countries' copyright laws have for a generation or more protected computer programs by subsuming them under the class of literary works for copyright purposes⁴⁹. As such, the holder of copyright in original software has the exclusive right to do certain acts that copyright restricts. The exclusive rights (in the USA⁵⁰) include to reproduce the copyrighted work and prepare derivative works based on it⁵¹, and the language of the GPLv2 draws on US copyright law in speaking of works 'based on' or 'derived from' the Program⁵².

Classically, in order to succeed in a claim for infringement of copyright in computer programs, a copyright holder has to show⁵³:

- that copyright is capable of subsisting and in fact subsists in the work at issue;
- that he/she is the owner of the copyright;
- that acts have been carried out within the exclusive rights of the right holder; and
- that those acts amount to infringement.

As mentioned above, the legal issues in pure copyright terms centre on whether the way in which GPL code (**B** in the example at paragraph 16, Linux at paragraph 20 and Java applets, servlets and JavaScript programs at paragraph 21) combines with your code (**A**, the would be GPL'd code) can potentially fall within and infringe the rightholder's exclusive rights.

For example, do any of the following cases where:

- in static linking, **A** (your code) at compile time copies into its files lines of code from **B** (Linux);
- in dynamic linking, **A** at run time copies into its files references to or a small number of lines of code from **B** (Linux or a GPL library or utility);
- **A** (your printer driver plug-in LKM) when loaded into the **B** (Linux) similarly copies references to or a small number of lines of code from **B**;

⁴⁹ See, e.g., in the EU, Council Directive 91/250 of 14 May 1991 on the legal protection of computer programs; in the UK, Section 3(1)(b) Copyright, Designs and Patents Act 1988 ('CDPA'); in the USA, e.g. *Apple Computer, Inc. v Franklin Computer Corp.* 714 F.2d 1249 (3d Cir. 1983) – but by 17 U.S.C. § 106 copyright is expressly stated not to “extend to any idea ... regardless of how it is described ...”; and generally, WIPO Copyright Treaty 1996, Article 4.

⁵⁰ 17 U.S.C. § 106

⁵¹ In 17 U.S.C §101, “derivative works”⁵¹ are defined separately from “compilations” and “collective works”. Essentially, a derivative work is a work based on one or more pre-existing works, a compilation is a work formed by collecting and assembling pre-existing works and a collective work is one where two or more independent works as contributions are assembled into a collective whole.

⁵² In the UK, these restricted acts include to copy the work and to make an adaptation of it; and, in relation to a computer program, adaptation means an arrangement, altered version or translation of it, translation being defined to include converting the program in or into a different language or code (in the UK under Sections 16 and 21 CDPA).

⁵³ The right holder also has to show that the work is within the term of copyright but this is not an issue for computer programs yet.

- **A** (your application software) makes system calls on **B** (Linux) through an API;
- **A** (your web browser application) causes its OS on which the JVM resides to execute **B** (a Java Applet with inheritance or a JavaScript program);

involve the carrying out of restricted acts/acts exclusive to the rightholder like copying code or creating a derivative work or adaptation?

Such cases raise complex copyright law issues which await judicial determination including:

- if copying has taken place, is what has been copied (**B**) capable of benefiting from copyright protection in the first place – for example, is it the only way of doing something so that copying is absolutely necessary; or is it an idea rather than its expression which has been copied?
- if copying has taken place, and what has been copied is capable of benefiting from copyright protection, does it actually attract copyright protection – for example, copying a single individual command or a very small amount of code might well be insufficient to amount to copyright infringement;
- do the acts carried out amount to the creation of a derivative work (**A + B**) based on **B**?
- if acts carried out would otherwise amount to infringement, is there a defence available, on grounds that:
 - they amount to fair use⁵⁴;
 - to hold that infringement had taken place would amount to derogation from grant⁵⁵; or
 - the right holder has implicitly or expressly licensed⁵⁶ the acts concerned.

It is the last of these points – whether the acts carried out are licensed and if so on what terms – that have in the absence of case law been the focus of attention, and to which we now turn.

23. **GPLv2 Section 2(b)**. The first point to note about the GPL is that, by Section O, “the act of running the program is not restricted”. Essentially, within certain limitations, internal only use of GPL code does not trigger copyleft consequences. What triggers copyleft under Section 2(b) is distribution or publication.

Section 2(b) is densely worded and it will help to ‘unpack it’. Using the ‘/’ sign to parse it in this way, what it says unpacked is effectively that: ‘you must cause to be licensed as a whole at no charge to everyone under the terms of GPLv2 / every work that you distribute or publish / the whole or part of which / **contains or is derived from**’ / the whole or part of / the [GPL] Program / a modified version of the whole [GPL] Program / a modified version of part of the [GPL] Program’.

To the copyright law complexity of how the rules outlined at paragraph 22 on copying and derivative works operate at the technical level outlined at paragraphs 17 to 21, Section 2(b) adds yet

⁵⁴ 17 U.S.C. § 106

⁵⁵ Colloquially, ‘giving with one hand and taking away with the other’.

⁵⁶ In copyright terms, a licence is simply permission to do something that the right holder could otherwise stop you doing by enforcing the copyright.

further complexity. This is largely because of the use of the word “contains”, which is open to widely varying interpretations: if **A** (your software) is combined to **B** (the GPL program), when should they be regarded as together forming a new program that “contains” **B**? And when should the combination be regarded as of separate and independent works that do not “contain” **B**?

The FSF, Mr Torvalds and other professionals from the IT industry have all formed their own individual views of the answer⁵⁷. The view of the FSF⁵⁸ is that what constitutes combining two pieces of code into one program depends both on the mechanism of communication (how intimately the two are connected) and the semantics of the communication (what kinds of information are interchanged).

There is a strong argument that static linking creates a work that “contains or is derived from” the GPL-licensed work: where static linking takes place the compiler binds together the GPL-licensed code (**B**, normally the library) with **A**, your code, to form a new program as a single executable. It is not difficult to see why this new executable is considered by many to be a work which “contains or is derived from” the GPL-licensed work.

There is greater debate on dynamic linking, where there is no permanent combination and only a temporary copy of all or part (and it may be a very small part) of **B** is formed in the computer’s memory when the new program is run or where proprietary device drivers (**A**) are used with by the Linux kernel (**B**) as LKMs or plug-ins. The case for creation of a single work containing or deriving from the GPL-licensed work here is weaker but not unarguable, and the GPL guardians argue strongly that Section 2(b) still applies in many cases of dynamic linking and to LKMs that communicate intimately with the Linux kernel. In the case of device drivers, it should be noted that in practice many if not most hardware vendors, especially the developers of advanced graphics cards that permit their hardware to be used with Linux, decline to publish the source code for the drivers required for that use.

Where applications make system calls on the kernel, certainly through an API and generally otherwise, there is little debate and general acceptance that Section 2(b) does not apply.

Finally, the FSF consider⁵⁹ that using Java Applets or Servlets with inheritance or JavaScript with your web browser is likely in certain circumstances to trigger Section 2(b).

It is widely perceived that the difficulties in interpreting Section 2(b) are themselves inhibiting the take up of GPL software and it is understood that FSF Europe and the FTF are currently working on guidance to give the matter clarity which it is hoped will shortly be issued.

24. Other GPL points. A common question about the GPL is whether it constitutes a bare licence or a contractual licence. The distinction is important first, because the terms of a contractual licence may be enforced against the licensor whilst a bare licensee cannot bring a claim against the licensor. Secondly, contractual terms may also be implied or disallowed by the courts in certain

⁵⁷ In GPLv3 the reference to “contains” is omitted and this should reduce the scope for confusion but as seen from the figure 2 at paragraph 11, GPLv2 remains 10 times more widely used than GPLv3.

⁵⁸ <http://www.gnu.org/licenses/gpl-faq.html#MereAggregation>

⁵⁹ <http://www.gnu.org/philosophy/javascript-trap.html>

circumstances. Thirdly, if the GPL is a contract then it is possible that the remedy of specific performance might be granted by a court and this would potentially be a powerful remedy if used to compel a licensee to provide access to source code.

The publicly expressed views of the FSF⁶⁰, now supported by the US Court of Appeals decision in the *Jacobson* case (see paragraph 29) are that the GPL is a bare copyright licence with conditions attached and that breach of those conditions automatically results in the loss of the licence to distribute. However, some commentators characterise the GPL as a contract. Initially a unilateral contract – an offer made to the world by the author to use his/her software in compliance with the GPL conditions – where the normal requirement to communicate acceptance is waived by the licensor⁶¹ so that when the software is modified or distributed, the offer is accepted by conduct and a bilateral contract is created.

If the GPL is characterised as a copyright licence only (not a contract), which would seem likely given the FSF's position on the subject, then the relief available in a UK court would be damages, an account of profits (or a hybrid of the two) and an injunction to restrain further infringement.

Finally, it should be noted that GPLv2 does not have any express governing law or jurisdiction terms and that consequently for any GPLv2 case outside the USA complex conflicts of law issues may arise.

25. Key GPL legal issues (2): 'Tivoisation'. 'Tivoisation' is the term given to a technical method used by Tivo, Inc.⁶² for preventing modifications to its set top box. It is a controversial practice because the Tivo set top box is also built on the Linux kernel, for which the GPL guarantees (amongst other things) the freedom to make modifications. Although Tivo publishes the software for the set top box in a high level programming language, it has used digital rights management ('DRM') effectively to remove the ability to make modifications. The hardware uses DRM techniques to check whether its software has been modified and refuses to run it if a certain signature is not present. The algorithm for producing a signature is not published so, effectively, only Tivo can modify the STB software.

Section 6 of GPLv3 seeks to outlaw 'Tivoisation' by requiring a licensee distributing software with a consumer product to accompany the source code with "Installation Information", meaning any methods, procedures, authorization keys, or other information required to install and execute modified versions of the software in the consumer product. The installation information must be sufficient to ensure that the mere act of modification does not cause the consumer product to stop working.

26. Key GPL legal issues (3): patents. GPLv3 goes further than GPLv2 in taking steps to defend OSS from further threats of patent infringement claims and Section 11 aims to prohibit future deals similar to the one struck by Microsoft and Novell in November 2006⁶³. It holds that if

⁶⁰ <http://www.gnu.org/press/mysql-affidavit.html> at paragraphs 18 and 22.

⁶¹ Section 5 of the GPL version 2 states "...by modifying or distributing the Program (or any work based upon the Program), you indicate your acceptance of this License to do so..."

⁶² www.tivo.com

⁶³ See footnote 26 above.

a patentee distributes GPL-licensed software and grants a patent licence to some of the parties receiving that software, then the patent license will automatically extend to all software recipients and to any works based on it.

27. **Key GPL legal issues (4): software as a service.** Software as a Service (SaaS) describes where software is hosted by a company and made available to users indirectly via a web browser. There has been considerable controversy over whether the source code for OSS hosted by a SaaS provider must be made available to the users. Under the wording of current OSS licences (except the GNU Affero General Public License), the hosting of OSS software by a SaaS provider would not appear to fall within the definition of redistribution. Indeed, Section 0 of GPLv3 notes that mere interaction with a user through a computer network, with no transfer of a copy of a program, is not conveying and as a result, the obligations to publish source code may not be triggered.

F. THE CASES: OSS IN THE COURTS

28. **USA: the SCO-Linux cases.** This is a long running series of US cases⁶⁴ between SCO Group (SCO) and various GNU/Linux end users and distributors and Novell. In 1990 AT&T, which had created the UNIX operating system in 1969, sold all its rights in UNIX to Novell. In 1995 Novell sold certain parts of its UNIX business to the Santa Cruz Operation and in 2000 Santa Cruz Operation sold its entire UNIX business to Caldera Systems, which then changed its name to SCO Group. SCO began making statements that it owned the rights to UNIX and that it would seek royalties from GNU/Linux users and distributors. Novell then claimed that the copyright in UNIX had not been assigned in the 1995 sale and began registering the copyrights to certain UNIX products. SCO filed actions, claiming slander of title⁶⁵. In the meantime, SCO had begun actions against two other GNU/Linux distributors, IBM (alleging that IBM was in breach of a contract with SCO when IBM provided source code for the GNU/Linux code base⁶⁶) and Red Hat⁶⁷. SCO also sued some of its users (including AutoZone⁶⁸ and DaimlerChrysler⁶⁹) who had stopped using SCO's UNIX for GNU/Linux. On August 10, 2007, the federal district court judge decided that Novell had retained the copyright to UNIX and not assigned it to SCO⁷⁰. On September 14, 2007 SCO filed for Chapter 11 bankruptcy protection but the Novell and IBM cases live on as at July 2009.

29. **USA: *Jacobsen v Katzer and Kamind Associates, Inc.***⁷¹. This case was of considerable concern to the OSS movement, when, at an interim hearing, the US District Court for the Northern District of California found that breach by the licensees of their obligations under a non-exclusive OSS licence gave rise to a claim for breach of contract rather than a claim for copyright infringement. Leading members of the OSS community including the OSI, SFLC, Linux Foundation and Creative Commons Corporation submitted as amici curiae a brief supporting an appeal from the District Court's decision. On 13 August 2008, the US Court of Appeals for the Federal Circuit vacated the District Court's decision, finding that the licensees' obligations were conditions limiting the scope of the licence and not independent contractual covenants. By failing to comply with those conditions, the licensees had acted outside the scope of the licence and therefore an action for copyright infringement could be brought by the licensor. The injunction application was remanded back so that the District Court could make a finding on the licensor's likelihood of success on the merits⁷².

⁶⁴ A good, comprehensive resource for further information on the SCO litigation is <http://www.groklaw.net/>

⁶⁵ <http://www.groklaw.net/staticpages/index.php?page=20040319041857760>

⁶⁶ <http://www.groklaw.net/staticpages/index.php?page=20031016162215566>

⁶⁷ <http://www.groklaw.net/staticpages/index.php?page=20031017044328636>

⁶⁸ <http://www.groklaw.net/staticpages/index.php?page=AZ-Timeline>

⁶⁹ <http://www.groklaw.net/staticpages/index.php?page=DC-Timeline>

⁷⁰ <http://www.groklaw.net/pdf/Novell-377.pdf>

⁷¹ See also paragraph 24.

⁷² See also case notes in the International Free and Open Source Software Law Review ('IFOSSLR', Volume 1, Issue 1, July 2009 - <http://www.ifossilr.org/ifossilr/issue/1/showToc>) at page 27 ('Bad facts make good law: the Jacobsen case and Open Source' by Lawrence Rosen) and at page 41 ('Jacobsen v Katzer and Kamind Associates – an English legal perspective' by Mark Henley).

30. **USA: *FSF v Cisco Systems, Inc.*** On 20 May 2009, the FSF announced the settlement of its long running dispute with Cisco⁷³. In the early 2000s, Linksys Group, Inc. adopted as its chip set for the wireless broadband router that it made a chip from US ‘system on a chip’ maker Broadcom Corp. that included a Linux distribution customised for Broadcom by CyberTAN Technology, Inc., a Taiwanese company. In 2003, Linksys was bought by Cisco Systems, Inc. for US\$500m. Linksys declined to publish the relevant source code, and the FSF became involved in 2004/5, seeking source code publication from Cisco. In the absence of an acceptable settlement FSF sued Cisco in the New York District Court in December 2008⁷⁴. Under the settlement, Cisco agreed to appoint a Free Software Director for Linksys (OSS compliance officer) required to make ongoing compliance reports to the FSF; to notify Linksys customers of their rights under the GPL; to publish compliant licence notices, to make complete, corresponding and up to date source code available on its website; and to make a monetary contribution to the FSF.

31. **Other US litigation.** Considering the widespread popularity of OSS and the GPL in particular, and aside from the SCO, Jacobsen and Cisco litigation there remains little in the way of US case law on the enforcement of OSS licences against licensees. However, the dearth of reported cases may be misleading to the extent that the FSF and, more recently, the SFLC have taken active roles in the USA in ensuring compliance with the GPL outside the courts. In September 2007 the SFLC launched the first US copyright infringement lawsuit based on non-compliance with the GPL. It has filed a half dozen or so further lawsuits since then and most have settled out of court⁷⁵.

32. **Germany: *gpl-violations.org: Sitecom, Fortinet, D-Link and Skype.*** Gpl-violations.org was founded by Harald Welte⁷⁶ in Germany in January 2004. Most of Mr Welte’s successes have been achieved informally outside the courts, but he has brought a handful of cases in Germany. In 2004 the German lower regional court of Munich⁷⁷ confirmed a temporary injunction enjoining the distribution of OSS in breach of the GPL’s requirements. The defendant, Sitecom, had used netfilter/iptables without providing access to the source code. The court ruled⁷⁸ that Sitecom was not entitled to use the netfilter/ip-tables code for its proprietary products and prohibited it from distributing them. The court, upholding the decision made at the interim hearing, held that the GPL licence terms had been validly agreed between the parties by way of standard licence terms and conditions and that the defendant was in breach of the licence.

In April 2005, gpl-violations.org brought an action against Fortinet UK for using GPL software in firewall and anti-virus products and trying to conceal the fact using cryptographic techniques. The Munich district court granted a preliminary injunction against Fortinet Ltd., banning further distribution of those products until they complied with the GPL⁷⁹.

⁷³ <http://www.fsf.org/news/2009-05-cisco-settlement.html>

⁷⁴ <http://www.fsf.org/licensing/complaint-2008-12-11.pdf>

⁷⁵ See <http://www.softwarefreedom.org/news/>

⁷⁶ Welte wrote the netfilter/iptables software used to provide firewall functionality in the Linux kernel which gives him standing to bring actions for copyright infringement against Linux kernel users who disobey the GPL’s conditions.

⁷⁷ LG München, dated May 19 2004, Az. 21 O 6123/2004

⁷⁸ An unofficial translation can be found at http://www.jbb.de/judgment_dc_munich_gpl.pdf

⁷⁹ <http://gpl-violations.org/news/20050414-fortinet-injunction.html>

In 2006, gpl-violations.org brought a claim in Frankfurt against D-Link for selling a data storage device using the Linux kernel without enclosing the GPL text, disclaiming any warranty or disclosing the source code. The court in its July 2006 ruling⁸⁰ treated Section 4 of the GPL as a condition subsequent which, when broken, revoked D-Link's licence to use the software.

In July 2007 it was reported⁸¹ that Skype had been found in breach of the GPL by a Munich regional court. The breach was said to involve the way in which Skype had distributed a VoIP handset using an embedded Linux kernel, having failed to supply the source code with the handset. Welte reported in May 2008⁸² that Skype had accepted that judgment and withdrawn its appeal.

Gpl-violations.org maintains that its success in these cases has meant that it has not had to bring any new actions in Germany for the last year or so and publicises its approach to as focusing on achieving publication of the source code rather than damages.

33. Informal interventions by the FSF. Although the FSF, like gpl-violations.org, has made informal approaches to users of GPL-licensed software allegedly breaching the licence conditions, these approaches do not tend to attract publicity. The first step towards enforcement generally comes when OSS activists inspect a software or hardware product looking for signs that GPL software has been incorporated. If they believe that GPL software has been used in breach of the GPL conditions, they will generally post the details to their blogs or to the online forums of gpl-violations.org or the FSF. Other activists will then re-analyse the products to verify the initial findings. The negative publicity that this starts to create within the OSS community may itself be sufficient to encourage GPL compliance. If not, gpl-violations.org or the FSF may enter into a dialogue with the company allegedly in breach. There are very few examples of informal dialogue failing to resolve the issue, but in principle the final stage could involve the matter being brought before the courts. As at July 2009, we are not aware of any court actions that FSF Europe has brought in the UK.

34. UK: FSF Europe and BT's Home Hub. One case that is reported to have attracted some notice, however, was the intervention by FSF Europe over BT's Home Hub⁸³, a network device that utilises the Linux kernel. BT was challenged by FSF Europe for distributing the Home Hub without making available the firmware source code. BT admitted that the Home Hub used Linux kernel version 2.6.8.1 under GPL v2, but stated that it also used proprietary software which it claimed was not subject to the GPL. In January 2007 FSF Europe notified BT of its claim that it was breaching the GPL. Since then BT has published source code for certain parts of the firmware⁸⁴ but FSF Europe argued that some of the necessary code was still missing, saying that the GPL required publication of a top level Makefile (a file used to assist compilation), the scripts that would be used to generate a firmware image and also a script or file containing configuration information for certain library files. In spite of FSF Europe's claims, no further action appears to have been taken against BT.

⁸⁰ An unofficial English translation of the case (6.9.2006 no. 2-6 O 224/2006) is at http://www.jbb.de/judgment_dc_frankfurt_gpl.pdf

⁸¹ <http://yro.slashdot.org/article.pl?sid=07/07/24/174240&from=rss>

⁸² <http://laforge.gnumonks.org/weblog/2008/05/08/>

⁸³ http://www.theregister.co.uk/2007/01/29/bt_says_enough_gpl/

⁸⁴ See http://www.btyahoo.com/broadband/adhoc_pages/gplcode.html

G. GOVERNANCE: OSS INSIDE THE ORGANISATION

35. **Introduction.** As a practical matter, day to day legal aspects of OSS increasingly focus on use within the organisation – on the contractual matrix of inbound and outbound contracts and on the governance matrix.

The contractual matrix

36. **Inbound transactions.** Overlapping with the organisational policy, organisations should carefully consider treatment from the OSS perspective of inbound transactions (for example, company acquisitions or software purchases where copyright and other IP is assigned or license in) from the OSS perspective. This applies whether the organisation is acquiring or disposing of companies whose assets include software, and whether that software is owned by or licensed to the company: in reality this will be relevant for most company acquisitions of any size.

As well as corporate transactions, organisations should also consider their procurement operations to make sure they have effective ways of verifying that code they buy or license in does not contain unexpected OSS and other necessary contractual cover. Black Duck Software⁸⁵ and Palamida⁸⁶ are two companies that provide automated audits of OSS source code. They provide consulting services for specific development projects and are able to generate a report which identifies the origins of the code included in any particular project's code base. They also identify the applicable OSS licences for management or the legal department. Hewlett-Packard's FOSSology⁸⁷ project also provides similar services.

37. **Outbound transactions.** Correspondingly, where an organisation licenses or supplies to its customers products or services that contain or use OSS (and so therefore 'distributes' or 'publishes' in GPLv2 terms), it will need to consider the types of commitments it is prepared to make in its outbound licensing, sale or supply terms. Traditional copyright-type warranties or covenants as to ownership, quiet possession, freedom from encumbrances and further assurance may well not apply to at least some of types of OSS risks a customer using OSS in breach of applicable licence terms may be exposed to. Customers are increasingly asking their suppliers to address these risks by in specific express contract terms, and suppliers should give consideration as to the contractual cover, if any, they are prepared to provide.

The governance matrix

38. **Governance: general.** Inside the business, OSS has frequently been the CIO's boon and the GC's burden: whilst use of OSS by the technology group frees up scarce software development resources for other, higher value projects, it has very often remained for the legal department to assess the risks that may arise, involving analysis of the OSS code and how it is used and assessing the applicable licences and their impact. Although the Gartner survey in November 2008 mentioned at the start found that only 30% of the organisations they surveyed had a corporate OSS policy framework in place, this proportion is currently rising quickly. With more widespread

⁸⁵ <http://www.blackducksoftware.com/>

⁸⁶ <http://www.palamida.com/>

⁸⁷ <http://www.fossology.org/home>

adoption of effective internal OSS governance – normative policies, procedures and strategies for compliant OSS use within the organisation - this bifurcation of reward and risk is likely to come more into balance as the governance/compliance responsibility and the administration that underpins it begin to fall more within the bailiwick of the CIO and senior technical management, with second line support from the GC's team available to them.

39. **Governance: the policy context.** Implementing an organisational policy on OSS use is critical for ensuring that employees do not use OSS in a way which will bring adverse unintended consequences. Any use of OSS should have an express determination by management that it will not conflict with the company's business model. There is a common misconception amongst programmers that OSS is "free" in all senses of that word. It is sometimes confused with "public domain" software, where the author has disclaimed any intention to enforce his/her rights against users of the software. In fact, uncontrolled use of OSS could oblige the organisation to share proprietary source code or else face an injunction removing its products from sale. Another risk introduced by uncontrolled use of OSS is of third party IP infringement claims. Code of unknown or dubious origin can bring risks of third party claims and by default OSS licences expressly disclaim any warranties or indemnities.

An organisation should consider establishing a policy for the types of OSS code that will be accepted for its products. It should consider when and with what level of legal supervision the company will accept OSS. In particular, it should consider whether the contributors to a particular OSS project have been authorised to take part by their employers and the terms on which they have been allowed to make their contributions. It may be that the contributors purport to contribute software without the full authorisation of the owners of the relevant IP, i.e. their employer. There are a number of other issues relating to the interaction between employees and the OSS community. In particular, there is a risk that employees inadvertently disclose the company's confidential information or license the company's valuable intellectual property. There is also a risk that an employee will contaminate the company's IP with third party IP acquired from the OSS community. In the event that the company ever needed to enforce its IP rights against a third party, it would first have to establish a clear chain of title, unravelling the effects of any third party IP contamination in the process.

Increasingly, in order to address these points, OSS governance from the policy context is considered at three levels of documents:

- the OSS **strategy statement** is the short, high level statement of the organisation's OSS vision, which will need to be aligned to other high level statements of corporate strategy;
- the **policy statement** will amplify the strategy statement and state particular aspects of best practice that the organisation wishes to set and comply with. The policy statement will be operate at the corporate level and may also take the form of a policy applicable to individual developers and other stakeholders that is given contractual effect as part of the company's HR manual, for example; and
- there will normally be a **statement of processes** that the company is to follow implementing the policy and strategy. The process statement may say whether a code indicator tool is to be used, and if so how and how it will be used with the company's source code management

mechanism. It will likely contain guidance for the developer community in the form of ‘do’s and ‘don’ts’ for software covered by particular OSS licences. It will state authority levels and how to go about adding new OSS software or licences to the company’s OSS intranet registry.

40. **Governance: the people context.** Secondly, governance should be an inclusive process taking account of the interests of all organisation stakeholders affected by OSS use – including:

- the development community;
- product managers;
- customers;
- finance;
- operations;
- the legal group; and
- senior management.

41. **Governance: the technical context.** Once OSS governance has been implemented it may be appropriate to carry out periodical code audits and for any non-compliance use to be assessed and remediated. The commercially available code indicator tools are useful here. It may be worthwhile considering an ‘amnesty’ before implementation to enable non-compliance use to be reported and addressed.

42. **Governance: awareness and training.** Finally, generating OSS awareness and an ongoing programme of training and communication on OSS benefits and risks is important in order to ensure buy-on from all stakeholder groups. This can be achieved through documents, webinars and blogs on the corporate intranet.

Kemp Little LLP (RHK), July 2009