

KEMP LITTLE LLP

OPEN SOURCE SOFTWARE: FREEDOMS, RESPONSIBILITIES AND GOVERNANCE

TABLE OF CONTENTS

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

36.	Informal interventions by the FSF	24
37.	UK: FSF Europe and BT’s Home Hub	25
G.	GOVERNANCE AND COMPLIANCE: OSS INSIDE THE ORGANISATION	26
38.	Introduction	26
	The contractual matrix	26
39.	Inbound transactions	26
40.	Outbound transactions	26
	Governance and compliance	27
41.	Governance: general	27
	Fundamentals of OSS governance	28
42.	Objectives	28
43.	Key principles	28
44.	OSS governance is particular to each organisation	28
45.	The range of organisations for which OSS governance is relevant	28
46.	Contexts of OSS governance – building blocks, threads and integration	29
47.	OSS achievements to date	29
	Thread 1: the people context	29
48.	Identifying/involving all stakeholder groups	29
	Thread 2: the strategy context	30
49.	Aligning OSS with other strategy statements	30
	Thread 3: the policy context	31
50.	The policy statement – the heart of OSS governance	31
	Thread 4: the process context	34
51.	Processes – dependencies, pre-, during and post-implementation	34
H.	CONCLUSION	35
52.	Conclusion	35

TABLES

1.	The Open Source Definition	5
2.	Top 20 Open Source Licenses (July 2009)	8
3.	OSS Licence Types	9
4.	GPL, Version 3: summary of key conditions	10
5.	Stakeholders, their OSS objective and how they are achieved	29
6.	OSS strategy statement for [Organisation]	30
7.	OSS policy statement for [Organisation]	31
8.	Checklist for OSS process statement for [Organisation]	34

OPEN SOURCE SOFTWARE: FREEDOMS, RESPONSIBILITIES AND GOVERNANCE

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 1970s¹, OSS emerged into the mainstream in the 1990s², continuing to become increasingly widely used in the 2000s so that today its use may fairly be said to be approaching ubiquity. Looking ahead, its scope and appeal is only likely to 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 computing, all of which are gaining traction in the late 'noughties';
- **the current generational shift in the software industry:** 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 is now firmly established. This shift is another spur for OSS; and
- **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 software³ 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'.

2. **Recent surveys/reports.** A number of recently published surveys and reports illustrate just how widely used and significant OSS has become:

- **Gartner November 2008 survey:** when IT consultancy Gartner released its survey in November 2008 of OSS use by 274 end user organisations around the world, it came up with two key findings:
 - At that time, 85% of the companies surveyed used OSS, and the remaining 15% were expecting to do so within the next 12 months.
 - 69% of the companies surveyed had no formal policy for evaluating or cataloguing use of OSS within their organisation.

The Gartner survey revealed a disconnect between uptake and effective governance: on Gartner's figures, OSS governance remained more widely honoured in the breach than the

¹ 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>).

² 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.

³ Although on a TCO (total cost of ownership) comparison basis the balance may not be so clear cut in favour of OSS.

observance and in the press release that accompanied the survey, Gartner’s research director Laurie Worster said:

“Just because something is free doesn’t mean it has no cost. Companies must have a policy for procuring OSS, deciding which applications will be supported by OSS and identifying the intellectual property risk or supportability risk associated with using OSS. Once a policy is in place, then there must be a governance process to enforce it”⁴.

- **Black Duck spring 2009 survey:** Gartner’s findings were corroborated by a survey published in March 2009 by Black Duck Software, the leading provider of OSS management services⁵, which (although of a smaller survey sample) found that only 40% of larger companies (those which employed 500 developers or more) had written governance policies and, of the sample as a whole including SMEs, only one in five had written governance in place.
- **Forrester spring 2009 report:** in its Spring 2009 report, ‘OSS Goes Mainstream’⁶, Research consultancy Forrester found Open Source Software (‘OSS’) to top the issues list for software decision makers inside the organisation: what they are looking for most is what will help them **“go faster, cheaper, better”** – improve integration, reduce cost and stay innovative – and OSS **“hit on all these points”**.
- **Black Duck end 2009 survey:** in a survey⁷ of 175 OSS users across a wide range of application segments concluded at the end of 2009, Black Duck found that OSS accounted for 20% of the code base of the average product or application surveyed and estimated the cost of developing that OSS code at around \$25m per product/application. Black Duck’s CEO commented:

“Driven by a new pragmatism with its roots in creating software more efficiently and effectively, development organizations and companies are using open source to gain significant competitive advantage in a multi-source development process”.

These reports and surveys point up the importance, ubiquity and rationale of OSS use today: it speeds up times to market, frees up expensive development resource for higher value work, tends towards standardisation in a converging world and hastens the trend towards cloud and service based computing. But these achievements have been made at a time when the legal uncertainties around the central OSS ‘copyleft’ inheritance principle has yet to be fully worked through in the courts.

3. **Purpose and scope.** Accordingly, the purpose of this White Paper is two fold: to serve as an introduction from the legal perspective to current OSS issues; and to provide guidance on the

⁴ <http://www.gartner.com/it/page.jsp?id=801412> - Gartner press release of 17 November 2008: “Gartner Says as Number of Business Processes Using Open-Source Software Increases, Companies Must Adopt and Enforce an OSS Policy”.

⁵ <http://www.blackducksoftware.com/news/releases/2009-03-11> - Black Duck press release of 11 March 2009: “Black Duck Survey Reveals Open Source Development Trends”.

⁶ http://www.forrester.com/rb/Research/open_source_software_goes_mainstream/q/id/54205/t/2

⁷ <http://www.blackducksoftware.com/news/releases/2009-11-10>

practical steps organisations may consider for effective OSS governance to balance the freedoms OSS confers and the responsibilities OSS licences impose. It combines and updates the previous (second, July 2009) version of our ‘Introduction to OSS’ and the first (December 2009) version of our ‘OSS Governance in the Organisation’ White Papers.

OSS is an increasingly broad subject. Much has been written about it⁸ 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.

⁸ 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

4. **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.

5. **The FSF and the four freedoms.** In 1985 Richard Stallman⁹, an ex-MIT academic, established the Free Software Foundation¹⁰ ('**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 licensee¹¹:

- 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.

6. **GNU, the GPL and Linux.** In 1983 Mr Stallman had announced the GNU¹² 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 Linux¹³ to create a complete operating system. This combination, known as GNU/Linux, was licensed under the GPL.

7. **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 Perens¹⁴ and Eric Raymond¹⁵, leading members in the OSS movement, established the Open Source Initiative¹⁶ ('**OSI**') to promote OSS on pragmatic rather than ethical or philosophical grounds¹⁷. The OSI is the steward of the Open Source Definition ('**OSD**'¹⁸) and part of its function

⁹ Personal home page at <http://www.stallman.org/>

¹⁰ www.fsf.org

¹¹ The second and fourth freedoms imply access to the source code.

¹² A recursive acronym for 'GNU's Not Unix'.

¹³ 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.

¹⁴ Personal home page at www.perens.com

¹⁵ Personal home page at <http://www.catb.org/~esr/>

¹⁶ <http://www.opensource.org/>

¹⁷ 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 Table 1).

TABLE 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.**

8. **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**'¹⁹), 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.org²⁰, 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**'²¹) and
- the Freedom Task Force ('**FTF**'²²).

¹⁸ The full definition is at <http://opensource.org/docs/osd>. The OSD was written by Mr Prerens, 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.

¹⁹ <http://www.softwarefreedom.org/>

²⁰ <http://gpl-violations.org/>

²¹ <http://www.fsfe.org/index.en.html>

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

C. OSS: THE BUSINESS CONTEXT

9. **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 sector²³.

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.

10. **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'²⁴ 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 Novell²⁵. In mid 2007 it

²³ 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>)

²⁴ See Mahony and Naughton cited at footnote 2 above.

²⁵ <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, Xandros²⁶ and Linspire²⁷.

In March 1999, IBM²⁸ 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 Microsystems, now owned by Oracle Corporation²⁹, has constantly been a significant proponent of OSS and arguably the largest corporate contributor to the Open Source movement. The OpenOffice.org³⁰ 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 Network³¹ (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.

11. 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 \$748m in its fiscal year to 28 February 2010³² and with over 3,000 employees ranks under Novell (approx 4,000 employees) and Oracle/Sun 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.

²⁶ <http://www.microsoft.com/presspass/press/2007/jun07/06-04XandrosPR.msp>

²⁷ <http://www.microsoft.com/presspass/press/2007/jun07/06-13LinspirePR.msp>

²⁸ <http://www.ibm.com/developerworks/opensource/>

²⁹ <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/>. The deal completed in January 2010 after the EU's concerns about Sun's MySQL OSS database business were resolved.

³⁰ <http://www.openoffice.org/>

³¹ <http://www.openinventionnetwork.com/>

³² <http://investors.redhat.com/releasedetail.cfm?ReleaseID=454482>

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

D. THE OSS LICENCES

12. **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 provider provides a regularly updated table of the top 20 OSS licences in use and their estimated popularity. Table 2 shows this table as at May 2010³⁴.

TABLE 2 – TOP 20 OPEN SOURCE LICENSES (MAY 2010)		
Rank	Licence (click on licence name to link to the licence text)	% use
1.	GNU General Public License (GPL) 2.0	48.51
2.	GNU Lesser General Public License (LGPL) 2.1	9.30
3.	Artistic License (Perl)	9.15
4.	Berkeley Software Distribution (BSD 2.0) License	6.24
5.	GNU General Public License (GPL) 3.0	5.63
6.	Apache License 2.0	4.10
7.	Massachusetts Institute of Technology (MIT) License	4.10
8.	Code Project Open 1.02 License	3.07
9.	Microsoft Public License (Ms-PL)	1.58
10.	Mozilla Public License (MPL) 1.1	1.21
11.	Common Public License (CPL)	0.55
12.	Eclipse Public License (EPL)	0.47
13.	GNU Lesser General Public License (LGPL) 3.0	0.45
14.	zlib/libpng License	0.43
15.	Academic Free License	0.39
16.	Common Development and Distribution License (CDDL)	0.33
17.	Open Software License (OSL)	0.31
18.	Mozilla Public License (MPL) 1.0	0.26
19.	PHP License 3.0	0.24
20.	Ruby License	0.24

Three things stand out particularly from the table:

- 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%;

³⁴ Black Duck Software – see <http://www.blackducksoftware.com/oss/licenses#top20>.

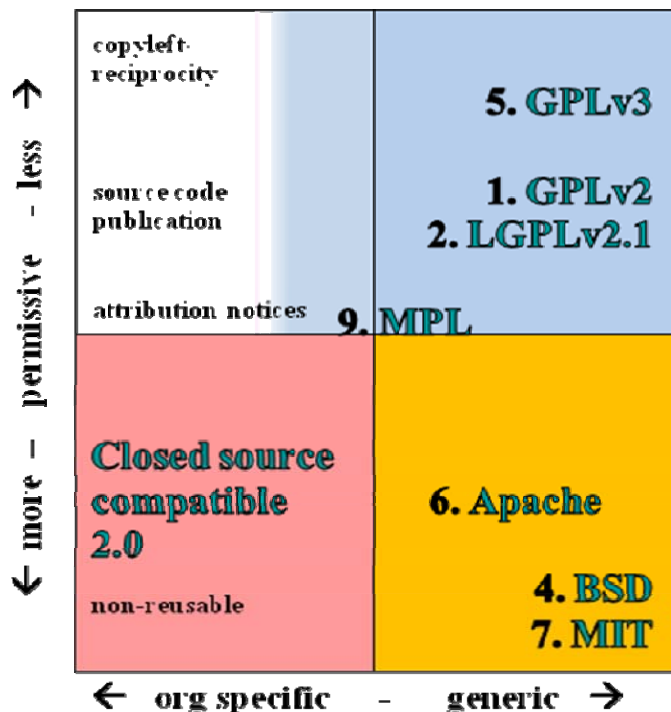
and the GPL version 3 ('GPLv3') and LGPLv3 (both published in Jun2 2007), 5% and 0.5% respectively;

- 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 around 75% of the total, and the top 10 licences over 90%.

13. **Feature range of OSS licences.** The chart at Table 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.

TABLE 3 – OSS LICENCE TYPES



14. **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 17 to 30** 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 and is summarised at Table 4 below.

TABLE 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).

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).

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 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.

15. **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.

16. **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>. As at April 2010, 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.

17. **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 18 and 19**);
- as practical examples relevant to Section 2(b), how Linux and scripting (Java and JavaScript) work (**paras 21 to 24**);
- relevant generally applicable copyright law principles behind Section 2(b) (**para 25**);
- how Section 2(b) may then be interpreted to apply, especially for Linux and Java (**26**); and

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)

- a number of additional GPL points relevant to the analysis (27).

18. **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.

19. **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

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.

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**.

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).

20. **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.

21. **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.

⁴⁵ See also paragraph 5 above.

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 19 and 20 and the Linux kernel can combine with:

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

22. **Scripting.** A “script” is a list of commands that can be passed to an application to make it perform certain tasks. Scripts generally are in source code form and require an interpreter to operate; they are distinct from “programs”, because programs can execute via a compiler on an operating system without necessarily interacting with any other application. At the same time, scripts are distinct from the core code of the applications that run them, which is usually written in a different language⁴⁶.

Client-side scripting are run by the viewing web browser. Client-side scripting is often used to change interface behaviours within a specific web page, in response to mouse or keyboard actions or at specified timing events. The dynamic behaviour occurs instantly within the browser.

Server-side scripts, on the other hand, 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. Server responses may also be determined by parameters in the URL, the type of browser being used or even the passage of time.

The result of either technique – client- or server-side scripting is described as a “dynamic web page”, and both may be used simultaneously.

23. **Java.** 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).

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**’⁴⁷).

⁴⁶ 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.

⁴⁷ The JVM can be hosted by (reside on) any OS or system architecture, and that is how platform independence is achieved.

Java applets are examples of client-side scripts, which are run by and within the web browser on the client computer. They 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 classes⁴⁸ and these files can raise GPLv2 copyleft issues if published under GPLv2.

24. **JavaScript.** JavaScript code (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 – may also raise GPLv2 copyleft issues when the code published under GPLv2.

The way in which GPLv2 JavaScript may be used is illustrated by means of the steps in the following example - where special offers from the retailer whose site the user is browsing are sent to the user:

- (i) The user types in a URL retailer address into his or her browser (eg www.retailer.com);
- (ii) The browser generates a request to the specified address for the web page;
- (iii) The request is routed via the Internet to the web application (the server);
- (iv) The server sends the web page to the browser which, in this illustration includes some JavaScript on the page which shows a map that the user can click on to get special offers from the retailer's outlet in/nearest to the location of the user's click;
- (v) The browser receives the web page and displays the page on screen for the user to see;
- (vi) The user clicks on the map, and the JavaScript program records where the user clicked, and converts this to coordinates;
- (vii) The JavaScript program sends the coordinates to the server (at this point the page on the user's screen does not change);
- (viii) The server receives the request, converts the coordinates to the nearest outlet of the retailer, and sends a list of special offer prices to the browser; and
- (ix) The JavaScript program receives the special offers, and displays them in the browser window for the user to select from.

This illustration raises two main issues where the JavaScript is licensed under GPLv2. First, the initial distribution at step (iv) is 'distribution' for GPLv2 purposes. This is generally not an issue as

⁴⁸ 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.

regards source code distribution as JavaScript is itself in source code form – being interpreted, rather than compiled at run-time⁴⁹.

Secondly, whether the server-side program that the GPLv2 JavaScript code communicates with could be considered to ‘contain’ or be ‘derived from’ the GPL v2 code within Article 2(b) of GPL v2 and so be considered to be a ‘work based on’ the GPL code within Article 2, GPL v2. This raises the questions overviewed at paragraph 20 above around the way in which programs are combined and which are addressed in outline at paragraph 25 and 26 below.

25. 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 purposes⁵⁰. 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 USA⁵¹) include to reproduce the copyrighted work and prepare derivative works based on it⁵², and the language of the GPLv2 draws on US copyright law in speaking of works ‘based on’ or ‘derived from’ the Program⁵³.

Classically, in order to succeed in a claim for infringement of copyright in computer programs, a copyright holder has to show⁵⁴:

- (i) that copyright is capable of subsisting and in fact subsists in the work at issue;
- (ii) that he/she is the owner of the copyright;
- (iii) that acts have been carried out within the exclusive rights of the right holder; and
- (iv) 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 potentially Java applets, servlets at paragraph 23 and JavaScript paragraph 24) 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);

⁴⁹ But see <http://www.gnu.org/philosophy/javascript-trap.html>

⁵⁰ 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.

⁵¹ 17 U.S.C. § 106

⁵² In 17 U.S.C §101, “derivative works”⁵² 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.

⁵³ 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).

⁵⁴ 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.

- 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**;
- **A** (your application software) makes system calls on **B** (Linux) through an API;
- **A** (JavaScript in your web browser application) communicates with and causes to execute **B** (a server application);

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 use⁵⁵;
 - to hold that infringement had taken place would amount to derogation from grant⁵⁶; or
 - the right holder has implicitly or expressly licensed⁵⁷ 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.

26. 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

⁵⁵ 17 U.S.C. § 106

⁵⁶ colloquially, ‘giving with one hand and taking away with the other’.

⁵⁷ In copyright terms, a licence is simply permission to do something that the right holder could otherwise stop you doing by enforcing the copyright.

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 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 answer⁵⁸. The view of the FSF⁵⁹ 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 in general terms less debate and a greater consensus that Section 2(b) does not apply.

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.

⁵⁸ 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.

⁵⁹ <http://www.gnu.org/licenses/gpl-faq.html#MereAggregation>

27. **Key GPL legal issues (2): bare or contractual licence?** A common question about the GPL is whether it constitutes a bare licence or a contractual licence. The distinction is important for three reasons:

- (i) a contractual licensee can enforce licence terms against the licensor but a bare licensee cannot bring a claim against the licensor;
- (ii) non-express terms may be implied by the courts in the ordinary way into a contractual, but not a bare licence; and
- (iii) 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, supported by the US Court of Appeals decision in the *Jacobsen* case before settlement in February 2010 (see **paragraph 32** below) 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 as a unilateral contract (an offer made to the world by the authors to use their 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.

28. **Key GPL legal issues (3): ‘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

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

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.

29. **Key GPL legal issues (4): 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 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.

30. **Key GPL legal issues (5): 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.

⁶³ See footnote 27 above.

F. THE CASES: OSS IN THE COURTS

31. **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. The IBM case was administratively closed on September 20, 2007 as a consequence but the Novell case lives on as at spring 2010.

32. **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 August 13, 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

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.

back to the District Court “for further proceedings consistent with this opinion” so that it could make a finding on the licensor’s likelihood of success on the merits⁷².

The litigation finally concluded with a settlement agreement on February 19, 2010, nearly four years after the case was launched with the defendants agreeing to pay Mr Jacobsen \$100,000 and accepting a permanent injunction against copying or modifying the software material at issue⁷³.

33. **USA: *FSF v Cisco Systems, Inc.*** On May 20, 2009, the FSF announced the settlement of its long running dispute with Cisco⁷⁴. 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 2008⁷⁵. 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.

34. **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 number of further lawsuits since then and most have settled out of court⁷⁶.

35. **Germany: gpl-violations.org: Sitecom, Fortinet, D-Link and Skype.** Gpl-violations.org was founded by Harald Welte⁷⁷ 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:

- **Sitecom:** In 2004 the German lower regional court of Munich⁷⁸ confirmed a temporary injunction enjoining the distribution of OSS in breach of the GPL’s requirements. The

⁷² See also case notes in the International Free and Open Source Software Law Review (‘IFOSSLR’, Volume 1, Issue 1, July 2009 - <http://www.ifosslr.org/ifosslr/article/view/5/9>) 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).

⁷³ See http://en.wikipedia.org/wiki/Jacobsen_v._Katzer#cite_note-Order_of_Dismissal.2C_Federal_District_Court_Doc..23404-0

⁷⁴ <http://www.fsf.org/news/2009-05-cisco-settlement.html>

⁷⁵ <http://www.fsf.org/licensing/complaint-2008-12-11.pdf>

⁷⁶ See <http://www.softwarefreedom.org/news/>

⁷⁷ 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.

⁷⁸ LG München, dated May 19 2004, Az. 21 O 6123/2004

defendant, Sitecom, had used netfilter/iptables without providing access to the source code. The court ruled⁷⁹ 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;

- **Fortinet:** 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 GPL⁸⁰;
- **D-Link:** 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 ruling⁸¹ treated Section 4 of the GPL as a condition subsequent which, when broken, revoked D-Link's licence to use the software;
- **Skype:** In July 2007 it was reported⁸² 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 2008⁸³ 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.

36. 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

⁷⁹ An unofficial translation can be found at http://www.jbb.de/judgment_dc_munich_gpl.pdf

⁸⁰ <http://gpl-violations.org/news/20050414-fortinet-injunction.html>

⁸¹ 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

⁸² <http://yro.slashdot.org/article.pl?sid=07/07/24/174240&from=rss>

⁸³ <http://laforge.gnumonks.org/weblog/2008/05/08/>

before the courts. As at spring 2010, we are not aware of any court actions that FSF Europe has brought in the UK.

37. **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 Hub⁸⁴, 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 firmware⁸⁵ 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.

⁸⁴ http://www.theregister.co.uk/2007/01/29/bt_says_enough_gpl/

⁸⁵ See http://www.btyahoo.com/broadband/adhoc_pages/gplcode.html

G. GOVERNANCE AND COMPLIANCE: OSS INSIDE THE ORGANISATION

38. **Introduction.** As OSS use has reached ubiquity in the organisation at a time when the legal uncertainties around the central OSS ‘copyleft’ inheritance principle has yet to be fully worked through in the courts, so increasing emphasis is currently being placed on a more formal approach to OSS governance and compliance as the practical way to balance and reconcile the freedoms and benefits that OSS confers with the responsibilities that OSS licences impose.

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 (**paragraphs 39 and 40**) and on the governance matrix (**paragraphs 41 to 51**).

The contractual matrix

39. **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 Software⁸⁶ and Palamida⁸⁷ 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 FOSSology⁸⁸ project also provides similar services.

40. **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.

⁸⁶ <http://www.blackducksoftware.com/>

⁸⁷ <http://www.palamida.com/>

⁸⁸ <http://www.fossology.org/home>

Governance and compliance

41. **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. OSS governance is now, somewhat belatedly, rising up the corporate agenda⁸⁹ and the purpose of the following paragraphs is to suggest a practical approach to creating documentation which will implement sensible, proportionate OSS governance⁹⁰. This approach is based on the premise that most organisations wish for reputational and competitive reasons to be seen to be good corporate citizens in their use of OSS.

Effectively, we propose a three-tier approach which ensures that internal governance discussions result in staff throughout the organisation buying into its statements of strategy, policy and process, and integrating them fully with how the organisation works. There is no magic to this approach, which focuses clearly on the high level issues, the policy that the organisation will define for its stakeholders and the day-to-day processes around implementation: collectively the OSS governance toolkit.

Different organisations' circumstances will differ widely so it is not practical to offer template 'one-size-fits-all' documents. However, the following paragraphs offers pointers as to what stakeholders should consider when developing OSS governance for their organisation and the areas that should be covered by strategy, policy and process statements.

⁸⁹ See e.g. the abstract from IT research company Forrester's paper "Best Practices: Improve Development Effectiveness Through Strategic Adoption Of Open Source" of 2 February 2009 – so just a couple of months before Forrester's 'OSS Goes Mainstream' report in spring 2009: "[OSS] is getting renewed attention from application development professionals who are looking for cost-saving alternatives amid the economic recession. But many aren't asking the right question: Instead of "should we adopt [OSS]?" they should be asking, "how will we adopt [OSS]?" [OSS] is already seeping into development shops through a variety of channels, whether managers know it or not. Unchecked tactical adoption of [OSS] creates unmanaged risk and unrealized returns, and application development professionals should not tolerate it. **Regardless of whether you view adoption of [OSS] as desirable or inevitable, the first step in moving from a tactical mess to a strategic plan is to specify the conditions under which [OSS] is permissible in your development shop. By creating a concise [OSS] policy, re-engineering the software acquisition process, and adding control points to [lifecycle management] processes and tools, application development professionals can shift from tactical responses to conscious integration based on realistic expectations and articulated economic benefits**" (emphasis added) (<http://www.forrester.com/Research/Document/Excerpt/0,7211,46361,00.html>).

⁹⁰ With the historically low take up of more formal OSS governance there has until recently been relatively little publicly available online material about OSS governance. OSS software/support developers Black Duck, Palamida and HP's FOSS Bazaar provide resources at:

- <http://www.blackducksoftware.com/resources/whitepapers#managingos> (Black Duck);
- http://www.palamida.com/themes/resources/Palamida_WhitePaper_PCIComplianceAtRisk.pdf (Palamida);
- <https://fossbazaar.org/openSourceGovernanceFundamentals> (White paper on FOSS Governance Fundamentals) and <https://fossbazaar.org/content/best-practices-open-source-governance> (Best Practices in Open Source Governance).

See also the OLEX (OpenLogic Exchange) Wazi at <http://olex.openlogic.com/wazi/2009/create-open-source-policy/> (Best Practices for Creating an Open Source Policy) and <http://olex.openlogic.com/wazi/2009/create-an-open-source-governance-process/> (From Policy to Process: Best Practices for Creating an Open Source Governance Process); and <http://www.softwarefreedom.org/resources/2008/foss-primer.pdf> (a Legal Issues Primer for Open Source and Free Software Projects). In the published books, see in particular Meeke, *The Open Source Alternative*, Wiley, 2008, Chapters 10 (Developing a Corporate Open Source Policy) and 10A (Open Source Corporate Policy) and Woods/Guliani, *Open Source for the Enterprise*, O'Reilly, 2005, Chapter 7 (Designing an Open Source Strategy).

Fundamentals of OSS governance

42. **Objectives.** Embarking on the journey towards effective OSS governance can be a challenging process for any organisation. Starting out, it is critical to know the direction of travel: what are the organisation's objectives for OSS and governance? As with other intellectual-property-based policies and governance, these can generally be succinctly stated in terms of reducing/managing risk and maximising reward by:

- (i) avoiding disputes and managing regulatory risks;
- (ii) achieving good management/housekeeping for a financial event – for example, an investment round, IPO or trade sale;
- (iii) ensuring customer satisfaction; and
- (iv) being a good corporate citizen.

43. **Key principles.** Supporting these objectives, the key principles of OSS governance may similarly be concisely articulated as:

- (i) *acquisition*: know what OSS your organisation is using;
- (ii) *source reliability*: know where that OSS is coming from;
- (iii) *tracking*: know what that OSS does and where it is being used and re-used;
- (iv) *roles and responsibilities*: know who is responsible for what in relation to OSS; and
- (v) *licence compliance*: know that your organisation is complying with its OSS licence obligations.

44. **OSS governance is particular to each organisation.** Effective OSS governance does not operate in a vacuum. It is relatively simple to state the basic key OSS governance objectives and principles, but applying them to a specific organisation requires that they be tailored, both in terms of high-level strategy and tactics, and day-to-day operational procedure.

45. **The range of organisations for which OSS governance is relevant.** If your organisation only uses OSS for internal purposes – that is, there is no re-distribution outside the organisation – the issues (and therefore the governance) will differ from those of an organisation that uses OSS in the products or services that it markets. Equally, in the 'internal use only' case, the position of a public sector organisation – say a Government Department or Local Authority - will be different from the private sector as public sector organisations, in their drive to use public money wisely, may be encouraged or mandated to use OSS over proprietary solutions and may have more formal, even statutorily prescribed, procurement procedures which OSS governance will need to be consistent with.

If your organisation develops software using OSS and then distributes software with OSS components (whether as a service or as a licence), the issues arising will probably be different and more complex issues than those involved where the use is only internal. There will also be a difference in emphasis depending on whether you are a business-to-consumer ('B2C') organisation supplying OSS components contained in the software you sell, or a business-to-business ('B2B') organisation whose end-user customers are other organisations who then sell on to the consumer. Other factors affecting the emphasis that OSS governance will take in any particular case include:

- The geographical spread of the business(es) – a company with a number of development centres around the world will look at things differently from a company with all its developers under one roof.
- Product spread – to take an example from the communications industry, a manufacturer of devices with embedded software applications like mobile phones will be in a different position from a fixed or mobile operator who principally supplies telecoms services rather than products (even if, as in the case of BT, the service may be delivered using a router containing embedded OSS applications as part of the service).

46. **Contexts of OSS governance – building blocks, threads and integration.** It is helpful to think of the components of successful OSS governance as building blocks, linked or threaded together by context. These connecting start with ‘achievements to date’ and previous experience of OSS governance implementation; they then integrate the people context (Section C, Table 1 below); the strategic context (Section D, Table 2); the policy context (Section E, Table 3); and the process context (Section F, Table 4). Each thread and the individual building blocks in it, needs then to be integrated across the organisation to set the context.

47. **OSS achievements to date.** Each organisation at the stage where it is considering formalising OSS governance will almost certainly have arrived at a start point which likely has some notable OSS achievements to date – it might have shaped the core OSS issues it faces in its business and may already have done ad hoc work identifying the top OSS licences it uses.

Thread 1: the people context

48. **Identifying/involving all stakeholder groups.** OSS use in the organisation on anything other than a purely ad hoc basis will involve a number of stakeholder groups both inside and outside the organisation, and effective OSS governance will depend on them integrating and cooperating in a way that is supportive and positive. There may well be many interested stakeholders whose interests will need intermediation in order to arrive at an agreed approach to governance. In Table 5 below, we have shown by way of illustration a description of potential stakeholders in an organisation, their possible objectives in relation to OSS and how those objectives can be achieved.

TABLE 5 - STAKEHOLDERS, THEIR OSS OBJECTIVES AND HOW THEY ARE ACHIEVED		
STAKEHOLDER/GROUP	PRIME OSS OBJECTIVE	HOW OBJECTIVE IS ACHIEVED
1 CEO/Leadership Team	To manage and ensure effective use of OSS aligned with corporate strategy	Shaping and delivering best practice to achieve OSS governance
2 CFO/Finance Team	To identify, quantify and manage the organisation’s OSS benefits and risks	Identifying and recording OSS components and licences and other commitments (such as other software assets)
3 CIO/Technical Team	To deliver OSS components and developments on time and on budget; to manage technical aspects of OSS governance programme	Implementing technical side of OSS governance (e.g. code indicator tool)
4 Customers	To gain business advantage through use of the organisation’s technology/services in knowledge that OSS risk is being managed	Performing contractual commitments contained in contracts with the organisation
5 Developers	To know that OSS use is encouraged and to understand how he/she is able to use OSS in daily work	Following OSS governance and feeding back on possibilities for improvement
6 Directors/Supervisory	To ensure that the organisation adopts	Formulating effective OSS governance policies and

	Board	appropriate OSS governance aligned to organisation's strategy	ensuring they are properly implemented
7	OSS Compliance Officer ('OSSCO')	To develop and implement OSS governance and ensure ongoing compliance with it	Articulating, agreeing and implementing OSS strategy, policy and process statements
8	OSS Working Party ('OSSWP')	To provide a focal point for interests of organisation's stakeholders and a crucible for OSS governance	Managing OSSCO; communicating back to other stakeholders
9	HR Team	To understand the HR and legal status to be given to OSS governance and policy statements	Ensuring that OSS policy statement forms part of the organisation's employee/contractor handbook
10	Legal Team	To minimise legal risks and maximise benefits to the organisation in its contractual commitments and OSS governance	Helping other stakeholders to manage OSS governance, with particular emphasis on documents (statements, contracts and so on)
11	Sales & Marketing Team	To generate revenue generation & reduce cost while ensuring customer satisfaction	Preventing unauthorised OSS use
12	Shareholders	To maximise share value	Using OSS in an efficient, compliant way to achieve cost reduction, increase in profit, increased competitiveness, increased efficiencies and reduced IP leakage
13	Suppliers	To perform contractual commitments in contracts with the organisation	Compliance with the organisation's inbound transactions/procurement policies for OSS

Thread 2: the strategy context

49. **Aligning OSS with other strategy statements.** OSS governance does not operate in a vacuum. At the highest level, it should align with other statements of organisational strategy – including corporate, risk management and IP strategy generally. The OSS strategy statement is also the mechanism by which stakeholders can establish and express an internal consensus. It can then provide a key point of reference for communication and education and for the development of the OSS policy statement. The organisation's leadership must be able to intermeditate between the different groups and arrive at an agreed, short, clear, high-level statement about where and why it will and will not use OSS. Each particular organisation adopting OSS Governance will need to consider its approach, but a suggested illustrative OSS strategy statement is set out at Table 6 below.

TABLE 6 - OSS STRATEGY STATEMENT FOR [ORGANISATION]

1. **[Organisation]'s OSS objectives.** [Organisation] will continue to use OSS in order to increase [Organisation]'s:
 - o ability to attract the best talent by building a development community at the forefront of OSS skills;
 - o competitiveness by increasing development and operational efficiency and effectiveness, enabling faster time to market and reducing costs; and
 - o value to stakeholders.
2. **OSS compliance.** [Organisation] fully recognises and respects the rights of, and its agreements with, others just as it expects others to respect [Organisation]'s rights and perform their agreements with us. Accordingly, [Organisation] respects the need to ensure compliance with its legal obligations in licence agreements for OSS that it uses.
3. **OSS governance within [Organisation]: achieving the right balance.** [Organisation] is committed to implementing best-practice OSS governance. The purpose of [Organisation]'s best-practice OSS governance is effectively, appropriately, proportionately and transparently to balance the objectives set out at paragraph 1 and the compliance expectation set out paragraph 2. This balance will be achieved:

within [Organisation]:

 - o by supporting [Organisation]'s development community in its work - as governance for developers by developers;
 - o by effective communication, including educating, training and raising/maintaining awareness of OSS issues among all stakeholders;
 - o by taking into account the interests of all stakeholders; and
 - o through the active and timely support of all stakeholders;

with [Organisation]’s partners:

- by ensuring that [Organisation]’s supplier and customer partners are aware of and comply with their OSS obligations, through [Organisation]’s contracts and appropriate relationship management.
- 4. **The mixed software environment.** [Organisation]’s use of OSS will continue to be in a ‘mixed’ software environment:
 - using OSS and proprietary software owned by [Organisation] and third parties;
 - constantly evaluating where OSS is best used within [Organisation]; and
 - re-using OSS components where appropriate thereby leveraging [Organisation]’s knowledge and technical resources.
- 5. **Further details.** This strategy statement forms part of [Organisation]’s OSS governance along with our policy statement and process statement. It is subject to review and change. For further details please contact [Organisation]’s OSS Compliance Officer at [email] and our OSS online resource kit at [intranet URL].

Thread 3: the policy context

50. **The policy statement – the heart of OSS governance.** The heart of OSS governance and compliance is the OSS policy statement. A well-crafted policy should:

- be **clear and brief**, otherwise people won’t read and understand it;
- be **event-driven**, setting out roles and responsibilities: to whom should OSS queries be addressed, and who does what in particular scenarios;
- set out **criteria and decision points** for OSS use. Apply Occam’s razor – the simplest answer is usually the best – and try to calibrate the policy so it will settle 80% of decisions, while also providing for effective management of exceptions; and
- set out the **information** to be collected and tracked.

Although, again, each organisations will need to consider what is appropriate for its own circumstances, we have also suggested illustrative wording for an OSS policy in Table 7 below around three main headings: scope and rationale; roles, responsibilities, training and awareness; and transaction type (inbound, in-house and outbound). The wording gives a starting point which will need to be adapted to fit the circumstances of the particular organisation involved.

TABLE 7 - OSS POLICY STATEMENT FOR [ORGANISATION]

A. SCOPE AND RATIONALE

1. Scope

- **Purpose:** This policy statement is designed to supplement our existing policy and processes relating to [Organisation]’s products and services. It deals specifically with those development and licensing considerations which must be fully understood and complied with when using and otherwise dealing with OSS within products and services that [Organisation] [markets][uses].
- **To whom does this policy statement apply?** This policy statement is mandatory and applies to everybody in [Organisation] who is responsible for [product design, launch and support] across all [Organisation]’s solutions, whether as an employee or contractor. The intention is to ensure that [Organisation] fully understands and complies with the obligations and duties as contained in the relevant OSS licence terms, and is also seen to do so.
- **What is the legal status of this policy statement?**⁹¹ This policy statement [forms part of [Organisation]’s HR handbook (for employees) and part of the Contractor handbook (for corporate and individual contractors)] and accordingly has the

⁹¹ A number of related issues arise here.

First, the HR aspects of the Policy are particularly important in considering how the organisation will ensure that OSS governance is effective. If it already has IT (email acceptable use for example) or intellectual property policies that are incorporated expressly or by reference into the HR handbook or even the contract of employment, it will be relatively straightforward to treat OSS governance similarly. If there is nothing comparable already in place, a number of questions need to be addressed, including particularly

same legal status as equivalent policy statements – [that is to say, it [forms part of your engagement terms with the Organisation]]];

- o **Design process:** All products and services that [Organisation] markets and that contain OSS must be [design-approved by the [Organisation] [review body OR OTHER AUTHORISED BODY OR PROCESS], taking into account architectural, security, legal, commercial and all other relevant considerations. In particular, as part of that design approval OSS licence terms must be understood and processes put in place to ensure [Organisation] compliance once the product/service is launched.
- o **Code indicator tool⁹²:** All source code in products and services that [Organisation] markets are to be scanned before launch using an OSS indicator tool. This will enable OSS code to be identified and all associated OSS licences to be checked for compliance with the licence’s terms. Information from the scan must be acted on so as to ensure [Organisation]’s compliance with the obligations in the relevant OSS licences.
- o **Further details:** This policy statement forms part of [Organisation]’s OSS governance along with our strategy statement and process statement. It is subject to review and change. For further details please contact [Organisation]’s OSS Compliance Officer at [email] and see our OSS online resource kit at [intranet URL].

2. Rationale

- o The rationale behind this part of the policy statement is to provide an introduction to OSS models and ensure OSS licences are given the attention and respect they require as a legal document.

The licensing of OSS code follows a different style of business model to the type [Organisation] has historically been used to. Most proprietary software is licensed under what can be called a **proprietary model**, where the copyright owner reserves all the rights the law grants, except for certain specific rights which are granted for a licence fee (for example, for £10 I license you (grant you permission) to use, but not to copy, modify or publish etc the software). OSS code on the other hand is in the main licensed either under:

- an **Academic Model** - such as the BSD, MIT, AFL or Apache licenses. Academic OSS Licences are typically light-touch agreements that basically seek “Freedom” for the software code. The main positive obligation on the Licensee is the duty to identify the origins of the OSS code – “attribution” ; or
 - a **Reciprocal Model** - such as the GPL, MPL, CPL and EPL. Reciprocal OSS Licences are generally more assertive in putting positive obligations on the Licensee with the objective of ensuring that all the copyright owner’s rights (to use, copy, modify, publish and so on) are passed down to other users.
- o [Organisation] will continue to operate in a ‘mixed’ software environment, using proprietary software under the **proprietary model** and (for OSS) the **Academic model** and the **reciprocal model**.
 - o Regardless of the underlying model, every software licence that attaches to software code (whether proprietary or OSS) constitutes a legal agreement between the licensor and the licensee. [Organisation] will comply fully with its legal obligations as set out in any licence agreement attaching to software code that is used within [Organisation], including that used within [Organisation] products or services.

B. ROLES, RESPONSIBILITIES, TRAINING AND AWARENESS

consequences of non-compliance where a developer uses OSS otherwise than in accordance with the OSS Policy or contributes to a OSS project otherwise than as permitted.

Secondly, how ‘binding’ does the Organisation want the policy statement to be on staff and/or contractors? If it is not of any binding effect, the policy may lack teeth. If it is binding, the usual way is to follow the language at the end of this bullet, and to say it forms part of the engagement arrangements. Either way, it should be consistent with policy statements that may be regarded as equivalent, like email acceptable use or IP policies for example.

Thirdly, HR difficulties can be compounded by the tension that generally arises between copyright law (where copyright in software developed by an employee in the course of his or her employment generally vests in the employer by operation of law) and code contributions to OSS projects (which generally provide that copyright in code contributed to the project is owned by the project). Again, corporate policy needs to be thought through and articulated in advance here.

Fourthly, it is worth remembering that under English law for example software developed by a contractor – whether an individual or a corporation – needs to be expressly assigned in order to belong to the organisation engaging the contractor.

⁹² The products of specialist OSS service providers like Black Duck, Palamida and Fossology and the code indicator tools and other technology platforms they supply can automate and take significant cost out of manual processes. An OSS indicator tool in particular serves a number of purposes. First, the code base of the organisation can be run through the tool in order to assess what OSS is currently in use internally; output in some cases can be aligned with the organisation’s source code management system. Secondly, on an inbound transaction, the organisation can use the tool to assess what OSS is in use, e.g. in an acquisition target (and check that the responses to due diligence for example are complete and accurate), or where the company is licensing in software from a third party. Thirdly, a number of the commercially available indicator tools can be ‘parameterised’/programmed in advance to set up agreed ‘do’s’ and ‘don’ts’ – rule of the road – for in-house OSS use. See also Table 4, Part B below (Processes).

3. Roles and responsibilities

- **OSS Compliance Officer**⁹³: In order to help [Organisation] achieve its OSS objectives, [Organisation] has created the position of OSS Compliance Officer ('OSSCO'). OSSCO will be the first line of support for the development community within [Organisation] on questions you may have about OSS.
 - **OSS working party**⁹⁴: OSSCO will report to the OSS working party ('OSSWP'). The OSSWP has members drawn from [Organisation]'s stakeholders. The role of the OSSWP is to give guidance to the OSSCO and, reporting to [], to ensure that [Organisation]'s use of OSS is aligned with [Organisation]'s strategy and the OSS strategy statement.
4. **Training and awareness**⁹⁵. OSSCO and the OSSWP will organise and carry out regular and frequent OSS training and awareness to ensure that the principles of [Organisation]'s OSS strategy and policy are understood and met throughout [Organisation].

C. OSS POLICY FOR INBOUND TRANSACTIONS, IN-HOUSE DEVELOPMENT AND OUTBOUND TRANSACTIONS⁹⁶

5. OSS policy for inbound transactions

- **OSS in [Organisation]'s procurement policies**
 - Pre-contractual documents (RFIs, RFPs, and so on) and contracts are to provide that software deliverables to [Organisation] will not contain OSS unless OSS components have been individually identified before contract signature and expressly agreed by [Organisation].
 - [Organisation]'s procurement contracts will reserve the right for [Organisation] to apply code indicator tool to carry out assessment in any case.
 - [Organisation]'s procurement contracts will include warranty/indemnity protection for non-identified/agreed OSS and (in addition to normal remedies) provide for rewriting as remediation on case-by-case basis.
- **OSS in inbound development agreements**: as per procurement policies outlined above.
- **OSS in M&A**:
 - Technical and legal due diligence to be configured to enable all OSS in target company's code base to be identified early on.
 - Consider using code indicator tool provider on escrow basis to carry out independent assessment.
 - Allow sufficient time between signature of contracts and closing/completion for remediation by rewriting.
- OSSCO and [legal representative of [Organisation]] will be available to discuss particular issues arising on inbound transactions.

6. OSS policy on in-house development

- **outline of authorisation mechanism**:
 - OSG will operate across the organisation on the basis of pre-approved OSS components/software and the OSS licences that attach to them.
 - Assessments through indicator tool: [Organisation] will:
 - assess what OSS it [and its contractors] are using in its operations; and
 - associate the relevant OSS licences with the OSS so assessed to be used;by:
 - assessing 'incoming' code using the code indicator tool;
 - pre-launch/release code assessments; and
 - carrying out periodical assessments of internally developed code to verify that the OSS being used within [Organisation] is what is expected to be used;
 - Remediation where necessary: Co will develop a process to review, assess and remediate instances of non-compliance with [Organisation]'s policy statement or otherwise in relation to a particular OSS licence;
- **OSS licence approval**
 - approval will be on the basis of the OSS licences determined to be most commonly used within [Organisation];
 - *Approval 'do's and don'ts'*: approval will be to enable use of the software concerned on the basis of clear, short,

⁹³ The OSSCO and the OSSWP are the lynchpins of the OSS governance process. The OSSCO is generally drawn from the development or technical rather than Legal team in practice, with Legal team representation on the working party.

⁹⁴ See previous footnote.

⁹⁵ An effective, continuing communication, training and awareness programme is of the essence of good OSS governance.

⁹⁶ The OSS policy should be event driven – i.e. it needs to think through and define in advance the sorts of issues that will arise. It should then aim to prescribe decision making which will deal with 80% of the issues that arise, with effective escalation to deal promptly with the other 20%. The events in this illustration are defined by reference to inbound, in-house and outbound transactions.

simple ‘do’s and don’ts’ addressing the needs of Co developers;

- *Scope of approval*: Unapproved open source software, software licensed on an unapproved licence, or use outside the ‘do’s and don’ts’ will be prohibited;
- *Post-implementation approval*: The post-implementation approval process will involve the OSSCO and will be designed to support the development community in giving timely positive assistance whilst respecting open source licence obligations;

7. **[Organisation]’s policy on contributions to OSS projects.** [set out here whether and if so to what OSS projects and on what terms [Organisation] developers may contribute code and other work⁹⁷.

8. **OSS policy on outbound transactions.**

- [Organisation]’s template [licence/services agreements] set out [Organisation]’s approach to OSS in its customer contracts;
- OSSCO and [legal representative of [Organisation]] will be available to discuss particular issues arising on outbound transactions.

Thread 4: the process context

51. **Processes – dependencies, pre-, during and post-implementation.** The OSS processes should take the strain of OSS governance. The process context is where the interrelationships with and *dependencies* (Section A of Table 8 below) on policies outside the OSS area and other building blocks and threads within it need to integrate. The *pre-implementation* steps that the organisation may need to take are set out at Section B of Table 8.

The project should be *implemented* like any other development in the organisation, with proper resource allocation, planning, mapping and timetabling (Section C of Table 8). Consider using a pilot in one part of the business to gain experience that can then be rolled out across the organisation as a whole. Consider also an amnesty to get the development community onside – winning hearts and minds. As a practical matter, the importance of technology platforms to minimise time and cost, increase efficiency, enhance collaboration, improve record-keeping and ensure validation can scarcely be over-emphasised. The OSS governance processes will need to be supple enough to cater for the full range of activities *post-implementation* (Section D of Table 8).

TABLE 8 - CHECKLIST FOR OSS PROCESS STATEMENT FOR [ORGANISATION]

A. DEPENDENCIES

1. Dependencies on/links with:

- OSS governance strategy and policy statements;
- [Organisation] patents and other IPR policies;
- Relevant stakeholder groups such as the group(s) within the Organisation responsible for software architecting and strategic direction;
- Source code management (including tools such as concurrent versions systems (CVS) and subversion);
- HR policies;
- Inbound/outbound contract groups;
- Exit strategy (if applicable).

B. PRE-IMPLEMENTATION

2. **Project planning, road mapping, timetabling.** Treat implementation of OSS governance at the process level like any other development project – with sufficient/appropriate resources, and detailed project planning, road mapping, dependency management and timetabling.

⁹⁷ See footnote 8.

3. **Indicator tool implementation**⁹⁸. Consider procurement of, and budget implications for, indicator tool well in advance of OSS governance implementation.
4. **Initial assessment**. Consider initial code assessment (NB: make sure you can continue to use the assessment results even after the contract with the indicator tool provider has terminated).
5. Consider **amnesty** for developers pre-implementation to encourage/bolster need for compliance use post-implementation.
6. Consider **pilot project** implementation initially before roll out across [Organisation].

C. IMPLEMENTATION

7. **Approval for OSS licences most commonly used.**
 - o Identify [Organisation]'s 'top [X]' OSS licences most commonly used within [Organisation], e.g.: [list];
 - o Refer to [intranet hyperlink] for methodology of how these OSS licences have been identified and analysis;
8. **Approval 'do's and don'ts'**
 - o Consider approval on the basis of short form, easily accessible/readable '*Do's and Don'ts*'.
 - o Consider maintaining intranet URL to show OSS [components] whose licences have been approved.
 - o Consider maintaining separate intranet URL to show OSS licences that are approved for use.
 - o Consider maintaining separate intranet URL of OSS components/licences (if any) whose use always requires prior specific approval from FOSSCO/legal.
9. **Pre-launch/release compliance** check using code indicator tool or otherwise.
10. Set out **service levels** for FOSSCO/FOSSWG responses to individual questions outside scope of policy/process guidance.

D. POST-IMPLEMENTATION

11. Arrangements for **code and other information repository**.
12. Periodical **code assessment**.
13. **Remediation** where necessary.
14. **Training and awareness**.

H. CONCLUSION

52. **Conclusion.** As OSS use in the organisation reaches ubiquity, OSS governance is rapidly becoming a 'must have' not just a 'nice to have' in order to manage risk and benefit effectively. Each organisation's needs will be different, and senior management will need to consider all aspects of this complex question carefully before embarking on OSS governance implementation, as they would any sophisticated software development project. At the end of the journey, management is looking to have in place integrated processes across all relevant business functions to manage effective use of OSS throughout the organisation. To get there, it should consider disassembling the various pieces into their building block components and threading them together by start point (achievements to date), people (stakeholders) and the strategic, policy and process aspects.

Kemp Little LLP (RHK), May 2010

⁹⁸ See also footnote 93 above.