LEGAL ISSUES FOR THE USE OF FREE AND OPEN SOURCE SOFTWARE IN GOVERNMENT*

BRIAN FITZGERALD† AND NIC SUZOR‡

[This article explains the notion of free and open source software and the reasons why governments throughout the world are giving it close consideration. In particular, it highlights key legal issues facing the adoption and development of free and open source software by governments. From the aspect of government procurement, the article examines the models used by governments to create a level playing field for the supply of free and open source software, intellectual property warranties and indemnities and the operation of the Trade Practices Act 1974 *(Cth). In terms of government development of free and open source software, the article considers the licensing mechanisms that will be implemented in the development and distribution of such software. In the final section, the article assesses the threat software patents and the current SCO litigation provide for free and open source software. The article concludes by emphasising that governments need to be fully aware of this landscape to assess what is the most effective technology available.]*

CONTENTS

* We owe thanks to Graham Bassett (Barrister-at-Law, Lismore Chambers, New South Wales) for assisting us with the development of this paper. In particular, Graham helped us with the early sections of the paper and Table 1. This is a significantly revised version of a paper presented at the Linux Conference Australia in Adelaide, January 2004; Asian Law Institute Conference in Singapore, May 2004; and the Red Hat–Phillips Fox Free and Open Source Software Seminars in Canberra, Melbourne and Brisbane, November 2004. We would also like to acknowledge the efforts of the following people: Steve Alford and David Mackey of the Australian Government Information Management Office, for generating discussion of these issues in Australia; Mark Webbink, Vice-President of Red Hat Inc, for travelling to Australia twice in recent years to support discussion of these issues; Daniel Ravicher, senior counsel of the Free Software Foundation, for highlighting some finer points in this legal landscape during his recent visit; and Professor Eben Moglen, for his work as chief legal counsel for the Free Software Foundation. We thank all of them for the assistance they have provided to us. We were also fortunate to hear Richard Stallman talk in Brisbane in September 2004 and have benefited from hearing his latest thinking directly from the source.

† BA (Griff), LLB (Hons) (QUT), BCL (Oxon), LLM (Harv), PhD (Griff); Head of the School of Law, Queensland University of Technology; bf.fitzgerald@qut.edu.au.

‡ BInfTech, LLB (QUT); Research Assistant, Queensland University of Technology; nic@suzor.com.

I INTRODUCTION

A grassroots movement started by free software guru Richard Stallman in the 1980s has revolutionised the way we think about the development and distribution of computer software. Stallman was frustrated by the fact that he could not access the source code[1] of software that was controlling a Xerox printer in his lab at Massachusetts Institute of Technology ('MIT'). His quest to open up access to source code in software has led to the creation of a powerful form of collaboration known as the free software movement.[2]

Free software is distributed with the source code disclosed, or open, at the point of distribution. Non-free or proprietary software is distributed with no source code disclosed, meaning that anyone who wishes to discover that source code must engage in a difficult and time-consuming process of reverse engineering.[3] Many developers fear that openly distributing program source code will promote free-riding on community-based developments because it allows recipients to use software to their advantage and profit without giving back to the community.

In order to remedy the most extreme examples of this, Stallman ensured that the source code he distributed was covered by a legally-binding obligation: the

[1] 'Source code' refers to the human-readable instructions that comprise a computer program. Source code files must be processed by a utility known as a 'compiler', which translates the instructions into binary machine code capable of execution by a computer. In general, modifications to a program's functionality cannot be made without access to its source code: Wikipedia contributors, *Source Code* (7 July 2005) Wikipedia: The Free Encyclopedia <http://en.wikipedia.org/wiki/Source_code>.

[2] See generally Brian Fitzgerald and Graham Bassett (eds), *Legal Issues relating to Free and Open Source Software* (2004) <http://www.law.qut.edu.au/files/open_source_book.pdf>; Brian Fitzgerald and Graham Bassett, 'Legal Issues relating to Free and Open Source Software' (2001) 12 *Journal of Law and Information Science* 159.

[3] Reverse engineering involves 'decompiling' a program's machine code into partially-readable source code. However, the process is laborious and does not enable discovery of all aspects of the program's workings. See generally Mike Perry and Nasko Oskov, *Introduction to Reverse Engineering Software* (2004) Association for Computing Machinery <http://www.acm.uiuc.edu/sigmil/RevEng>.

GNU[4] General Public License ('GPL').[5] The GPL obliges those who modify free software code to disclose their modifications to any recipient of the altered software, which in essence means the whole community. In this way, the GPL attaches itself to the copyright in software code owned by a licensor, so as to oblige recipients to share their improvements for the benefit of all users.

This was Stallman's powerful insight: copyright in software code can be used not only to restrict access and exploit its benefits for monetary reward, but also to maintain open access for downstream users and developers. Thus, if software is released with free access to its source code, any improvements made by its users must be similarly disclosed.[6]

Today, many governments are expressing interest in the free software model, and the private sector is not far behind. Some governments have already begun the task of migrating to the use of free software in the public sector. The open source GNU/Linux operating system now rivals Microsoft Windows, at least at an institutional level.[7] The Australian Government Information Management Office ('AGIMO') recognises that the use of free and open source software is 'particularly widespread in areas such as network infrastructure, single-purpose computer servers, security, internet and intranet applications, and network communications' in both the private and public sectors.[8] The adoption of GNU/Linux and applications like OpenOffice and Mozilla Firefox for desktop

[4] 'GNU' is a recursive acronym for 'GNU's Not Unix':

> A recursive acronym is an acronym which refers to itself in the expression for which it stands, similar to a recursive abbreviation. The earliest example is perhaps the credit card VISA, which was named in 1976 as a recursive acronym for VISA International Service Association. In computing, it soon became a hackish (and especially MIT) tradition to choose acronyms and abbreviations which referred humorously to themselves or to other abbreviations. Perhaps the earliest example in this context, from about 1977 or 1978, is TINT ('TINT Is Not Teco'), an editor for MagicSix.

Wikipedia contributors, *Recursive Acronym* (2005) Wikipedia: The Free Encyclopedia <http://en.wikipedia.org/wiki/Recursive_acronym>.

[5] Free Software Foundation, *The GNU General Public License: Version 2.0* (June 1991) The GNU Project <http://www.gnu.org/licenses/gpl.html>.

[6] For a detailed overview of the processes of and motivations for peer- and user-led production, of which free software is the prime example, see Yochai Benkler, 'Coase's Penguin, or, Linux and *The Nature of the Firm*' (2002) 112 *Yale Law Journal* 369; Josh Lerner and Jean Tirole, 'Some Simple Economics of Open Source' (2002) 50 *Journal of Industrial Economics* 197; Eric von Hippel, 'Innovation by User Communities: Learning from Open Source Software' (2001) 42 *Sloan Management Review* 82. Benkler's key point is that, in certain circumstances, peer production can most efficiently bring together the best intellects for the job. This is possible because it has the capacity to utilise a vast distributed network of knowledge, which in certain circumstances is far superior to any other formally- or traditionally-organised mode of knowledge production. Benkler explains that motivation to participate in sharing of knowledge through peer production is, on current evidence, reasonably achievable due to 'indirect appropriation' — including money, design of the end product, and pleasure or social profile gained through involvement in peer production: at 424–5.

[7] See generally Gartner Group, *A Final Report for the Australian Taxation Office: Open Source Study* (2003) <http://www.ato.gov.au/content/downloads/OpenSourceStudyFinal.pdf> ('Gartner Report').

[8] AGIMO, *A Guide to Open Source Software for Australian Government Agencies* (2005) 10 <http://www.sourceit.gov.au/data/assets/pdf_file/42065/A_Guide_to_Open_Source_Software.pdf>. Netcraft, a respected internet research and analysis organisation, suggests in its most recent survey that over 69 per cent of all active websites use the free Apache web server: Netcraft, *June 2005 Web Server Survey* (1 June 2005) <http://news.netcraft.com/archives/2005/06/01/june_2005_web_server_survey.html>.

computers has not been as rapid, but there is growing interest evident amongst large-scale government, business and end users.[9]

This article highlights the key issues facing governments in adopting and developing free software. Part III considers several benefits associated with government usage of free software. Part IV explores how issues of procurement, intellectual property infringement and non-excludable warranties interact with the development and supply of free software in the public sector. Part V deals with licensing issues in public sector development. In Part VI, we consider threats to the free software movement, including software patents and recent litigation against its adopters. We conclude that there are significant policy arguments in favour of governments adopting free software in appropriate cases, and a solid, informed analysis of the benefits and risks involved should be undertaken when evaluating proposed software solutions in the public sector. In order to fully appreciate these issues, we need to start with an understanding of the concept of free software and the most important free software licences.

II THE FREE SOFTWARE MODEL

A *What Is 'Free Software'?*

Richard Stallman, founder of the Free Software Foundation ('FSF'), describes four values embodied in the phrase 'free software':

- the freedom to run the program, for any purpose ('freedom 0');
- the freedom to study how the program works, and adapt it to your needs ('freedom 1'). Access to the source code is a precondition for this;
- the freedom to redistribute copies so you can help your neighbour ('freedom 2'); and
- the freedom to improve the program and release your improvements to the public, so that the whole community benefits ('freedom 3'). Access to the source code is a precondition for this.[10]

Free software is not free because it has no price; it is free because it embodies values that enhance liberty for users and programmers. As Stallman points out, 'when I speak of free software, I'm referring to freedom, not price. So think of free speech, not free beer.'[11]

Alongside the support of the open source developer community, it is through the legal mechanism of the licence that the free software model is implemented

[9] Elizabeth Millard, *Firefox Continues to Gain Browser Share* (18 July 2005) CIO Today <http://www.cio-today.com/news/Firefox-Continues-To-Gain-Share/story.xhtml?story_id=102003F7F7R0>.

[10] FSF, *The Free Software Definition* (27 October 2001) <http://www.fsf.org/philosophy/freesw.html>.

[11] Richard Stallman, 'Free Software: Freedom and Cooperation' (Speech delivered at New York University, New York, 29 May 2001) <http://www.gnu.org/events/rms-nyu-2001-transcript.txt>. On the power of free software models to enhance digital diversity, see Brian Fitzgerald, 'Intellectual Property Rights in Digital Architecture (Including Software): The Question of Digital Diversity' (2001) 23 *European Intellectual Property Review* 121; Brian F Fitzgerald, 'Software as Discourse: The Power of Intellectual Property in Digital Architecture' (2000) 18 *Cardozo Arts and Entertainment Law Journal* 337.

and maintained. There are two main types of free and open source software licences. Simpler licences, such as the revised Berkeley Software Distribution ('BSD')[12] and MIT/X Window System ('MIT/X11') licences, allow redistribution of the licensed program and its use in both source and binary forms,[13] with or without modifications, on the conditions that the copyright notice is retained and any applicable warranties are disclaimed. There is no requirement that derivatives of the free software must themselves be free. On the other hand, 'copyleft' licences (such as the GPL) attempt to create a contributory commons by requiring that any redistribution of the software or its derivatives also occurs under the free licence.[14]

B *GPL and Copyleft Licences*

A licensing system that promoted sharing and innovation was integral to the development of GNU/Linux.[15] Stallman realised that without a legal mechanism to protect free software, commercial parties could incorporate free code in their developments without any obligation to make their improved or derivative source code available for access. To remedy this, Stallman created the GPL. The GPL covers the initial program and 'any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language'.[16]

Stallman places the GPL in a direct commercial and political context called 'copyleft':

> To copyleft a program, we first state that it is copyrighted; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code *or any program derived from it* but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable.[17]

[12] The original BSD licence had what came to be known as an 'obnoxious advertising clause', which required attribution to be displayed on all advertising materials. This caused a problem when there were many contributors to a project, because the attribution material quickly became large and unwieldy. Current versions of the licence do not include this clause, but there are still many examples of software products released under the original licence or modified versions thereof: FSF, *The BSD License Problem* (2005) The GNU Project <http://www.gnu.org/philosophy/bsd.html>.

[13] As noted above, a 'binary' is a program which has been compiled (automatically transposed) from human-readable source code into a machine-executable binary format. Usage in binary form normally entails running the program, while use of the source code will typically involve viewing, modifying or compiling it: see generally Wikipedia contributors, *Executable* (27 June 2005) Wikipedia: The Free Encyclopedia <http://en.wikipedia.org/wiki/Executable>.

[14] See Lawrence Rosen, *Open Source Licensing: Software Freedom and Intellectual Property Law* (2004) 105.

[15] For a detailed overview of the history of GNU/Linux, see Glyn Moody, *Rebel Code: Linux and the Open Source Revolution* (2001). See also Lawrence Lessig, *The Future of Ideas: The Fate of the Commons in a Connected World* (2001) 50–5; Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software* (2002) ch 11.

[16] FSF, *The GNU General Public License: Version 2.0*, above n 5. For a detailed discussion of the notion of 'derivative works' and the GPL, see Fitzgerald and Bassett, *Legal Issues relating to Free and Open Source Software*, above n 2, 29–32.

[17] FSF, *What is Copyleft?* (5 May 2005) The GNU Project <http://www.gnu.org/copyleft/copyleft.html> (emphasis in original).

It is a rudimentary yet powerful licence. By releasing code under the GPL, the licensor creates an obligation to make accessible the source code of any software deriving from the licensed source code. Consequently, a commercial developer who takes free code under a GPL licence and incorporates it into the code of their product is (upon distribution) obliged to make the source code of the entire product available to its recipients. As Stallman explains, the GPL 'has the strength to say no to people who would be parasites on our community.'[18] It has this strength because Stallman and the FSF foresaw that they could exploit their copyright in software to control the way the software code was treated once it left their hands.[19]

Copyleft software licences (sometimes called 'restrictive free licences') retain software freedom for downstream users by preventing proprietary 'code forking'. Code forking occurs when a version of the code is taken by a particular person or group of people in order to continue development of the software in a different direction from the original code — one code base branches out into several projects. Code forking is instrumental in preventing any single organisation from dictating the way in which software must develop. Problems arise when an entity takes free software and closes the code, creating a new proprietary product, which may be developed and commercialised without returning its improvements. The GPL prevents this by requiring a licensee, upon distribution of a derivative work, to release the source code of any changes or modifications to the community under the same terms as the licensor.

Non-restrictive free software licences, on the other hand, do not include a similar restriction and will allow proprietary derivative code to be created and distributed. For example, the original Apache[20] licence allowed a derivative work to be released with or without modifications in source or binary form. The licensee could make changes without a requirement to share them, provided the name of the derivative work was changed. The revised BSD licence does not oblige modifiers of licensed software to disclose the source code to their modifications when distributing a derivative work.

The GNU Lesser General Public License ('LGPL') provides what is known as a 'weak copyleft'. It does this by requiring any changes to the software itself to be licensed under the same terms, but — unlike the GPL — allows linking of the software to non-free programs. The LGPL is useful for free software libraries that, for compatibility reasons, would benefit from incorporation into non-free software. We discuss the issue of linking in greater detail below. The differences between copyleft and non-copyleft free software licences are highlighted in the following table, which provides examples of key clauses.[21]

[18] Stallman, 'Free Software: Freedom and Cooperation', above n 11.

[19] Anne Fitzgerald and Brian Fitzgerald, *Intellectual Property in Principle* (2004) 414–20.

[20] The Apache HTTP Server Project is an open source web server application. It is now more widely adopted than Microsoft's Internet Information Services platform and is used to display more than 68 per cent of webpages: Apache Foundation, *Welcome* Apache HTTP Server Project <http://httpd.apache.org>.

[21] See generally Joe Barr, *Live and Let Licence* (23 May 2001) ITworld.com <http://www.itworld.com/AppDev/350/LWD010523vcontrol4>; Larry Rosen, *Which Open Source Licence*

Table 1: Examples of Free Software Licences[22]

	GPL	LGPL	Revised BSD	MIT/ X11
Allows copying and distribution of verbatim copies	✓	✓	✓	✓
Allows charging of fees for copies	✓	✓	✓	✓
Allows charging of fees for warranty protection	✓	✓	✓	✓
Allows publication in binary form without accompanying source code			✓	✓
Allows distribution of modified versions of the software under the same licence	✓	✓[23]	✓	✓
Allows distribution of the software or derivatives under other licences		✓[24]	✓	✓
Allows linking with software released under other licences		✓	✓	✓
Requires changes to the software to be documented	✓	✓		
Requires republication of the original copyright notice	✓	✓	✓	✓
Requires publication of a disclaimer of warranty for redistributions	✓	✓	✓	✓
Prohibits use of upstream authors' names to promote or endorse derivatives			✓	✓

All of these licences are free software licences. The GPL and LGPL are copyleft licences, and attempt to ensure that any changes to the software are released to the public. The BSD- and MIT/X11-style licences are simpler and allow downstream developers to use the code in nearly any way they see fit. Which licence a developer chooses is often dependent on his or her goals. The GPL helps foster and protect a free software community and codebase, while BSD-style licences are more useful for authors coding for the benefit of all potential downstream uses, whether in free software or not. Finally, the LGPL is

Should I Use for My Software? (2001) Rosenlaw and Einschlag Technology Law Offices <http://www.rosenlaw.com/html/GL5.pdf>.

22 FSF, *The GNU General Public License: Version 2.0*, above n 5.

23 The LGPL only allows modified versions to be incorporated into other LGPL-licensed libraries, not other programs.

24 Any software released under the LGPL may be relicensed under the GPL. This is useful when incorporating LGPL code into software that is not a library.

useful for promoting the adoption of open standards, by allowing integration of common library implementations into all software, but still requiring changes to the library to be openly licensed.

A further aspect that warrants clarification is terminology; namely, what is the difference between free software[25] and open source software?

C *The Open Source Initiative*

The Open Source Initiative ('OSI') is a non-profit organisation. Its leading proponent, Eric Raymond, has conceptualised business models enabling commercial exploitation of open source programs.[26] Programs distributed with the Open Source Certified trademark[27] are published on an approved list of licences[28] that conform to the open source definition.[29] The main elements of such licences are:

- free redistribution so that a party may not require a fee or royalty for the downstream distribution of the program;
- the program must include source code and allow distribution in source as well as compiled form. If a program is not distributed with its source code, there must be a well-publicised means of obtaining the source code for no more than a reasonable reproduction cost — preferably by downloading from the internet without charge;
- derived works and modifications must be allowed and be capable of distribution under the same terms as the original licence;
- the licence may preserve the integrity of the original author's code by requiring that any redistributions include the source code in its unmodified format along with separate patches (modification chunks) to incorporate subsequent alterations. In this way, 'unofficial' changes can be made available but readily distinguished from the base source code;
- the licence must not discriminate against any person, group of persons, fields of endeavour, technology or software package;
- the right to use the program must not be contingent upon entry to some other form of licence or agreement such as a non-disclosure agreement;
- the right to use the program must not be contingent upon the program being part of a particular software distribution; and

[25] Free software should be distinguished from the term 'freeware', which means free in price to download: *Trumpet Software Pty Ltd v OzEmail Pty Ltd* (1996) 34 IPR 481, 485 (Heerey J). 'Freeware' has nothing to do with the notion of making source code available for access and should not be used to describe the free software model.

[26] These business models include: loss leader; widget frosting; give away recipe/open restaurant; accessorising; free the future, sell the present; free the software, sell the brand; and free the software, sell the content: Eric S Raymond, *The Cathedral and the Bazaar* (2005) Eric S Raymond's Home Page <http://www.catb.org/~esr/writings/cathedral-bazaar>; Shawn W Potter, 'Opening Up to Open Source' (2000) 6 *Richmond Journal of Law and the Public Interest* 24; Martin Fink, *The Business and Economics of Linux and Open Source* (2002).

[27] OSI, *OSI Certification Mark and Program* (2005) <http://www.opensource.org/docs/certification_mark.html>.

[28] OSI, *The Approved Licenses* (2005) <http://www.opensource.org/licenses/index.html>.

[29] OSI, *The Open Source Definition: Version 1.9* (1997) <http://www.opensource.org/docs/definition.html>.

- the licence must not place restrictions on other software that is distributed along with the licensed software. For example, the licence must not insist that all other programs distributed on the same medium must be open source software.[30]

D *Tension between 'Open Source' and 'Free' Software*

The difference between free software and open source software is mainly a philosophical one. Because the definition of 'open source' is somewhat broader than that of 'free' software, it is clear that all free software is open source, but not all open source software is free. In practice, however, most licences that satisfy the OSI definition will also be considered 'free'.[31]

The OSI was initially formed by a small group of computer scientists, including Bruce Perens and Eric Raymond, in order to promote the commercial uptake of free software and respond to concerns that the term 'free' would discourage commercial adoption. While the definition of 'open source' was drawn from accepted free software guidelines,[32] the emphasis of the OSI was not on freedom but on the benefits of using an open source methodology for software development. After a short period, Bruce Perens resigned from the board of OSI, regretting that 'Open Source has de-emphasized the importance of the freedoms involved in Free Software.'[33]

The FSF has noted that the changed focus of open source software encourages commercial developers to

> gain the favourable cachet of 'open source' for their proprietary software products — even though those are not 'open source software' — because they have

[30] Ibid.

[31] For an example of an OSI-approved licence that is not considered 'free' by the FSF, see Technical Pursuit Inc, *Reciprocal Public License: Version 1.1* (2002) Open Source <http://www.opensource.org/licenses/rpl.php>. The FSF considers that this licence is non-free because '1. It puts limits on prices charged for an initial copy. 2. It requires notification of the original developer for publication of a modified version. 3. It requires publication of any modified version that an organization uses, even privately': FSF, *Various Licences and Comments about Them*, The GNU Project <http://www.gnu.org/licenses/license-list.html#NonFreeSoftware License>. There are also licences that are considered free by the FSF and the OSI but not by some prominent free software groups (like Debian). See, eg, The Mozilla Foundation, *Mozilla Public Licence Version 1.1* (8 November 2004) Mozilla <http://www.mozilla.org/MPL/MPL-1.1.html>, which Debian does not understand to be clearly free and therefore does not distribute in its main GNU/Linux distribution: Branden Robinson, *Re: Bug#251983: libcwd: QPL License Is Non-Free; Package Should Not Be in Main* (8 June 2004) Debian-Legal Mailing List <http://lists.debian.org/debianlegal/2004/06/msg00131.html>; Barak Pearlmutter, *OCAML QPL Issue* (8 March 2003) Debian-Legal Mailing List <http://lists.debian.org/debian-legal/2003/03/msg00459.html>; Martin Krafft, *Moving libcwd to Debian Non-Free* (2 October 2004) Debian-Legal Mailing List <http://lists.debian.org/debian-legal/2004/10/msg00009.html>.

[32] The initial OSI definition of 'open source' was identical to that adopted by the following guidelines: The Debian Project, *Debian Social Contract* (26 April 2004) Debian <http://www.debian.org/social_contract.html#guidelines>. See OSI, *History of the OSI* (2005) <http://www.opensource.org/docs/history.php>.

[33] Bruce Perens, *It's Time to Talk about Free Software Again* (17 February 1999) Debian-Devel Mailing List <http://lists.debian.org/debian-devel/1999/02/msg01641.html>.

> some relationship to free software or because the same company also maintains some free software.[34]

By doing this, developers reap the benefits of the open source development methodology without returning their own improvements to the users of free software.

In an effort to be all-encompassing when discussing this area of activity, while still respecting the nuances of these ideological differences, it has become fashionable to use the term 'free and open source software' ('FOSS').

Why, then, have governments become interested in this grassroots and ideologically-charged model of software development and distribution?

III Benefits of FOSS for Governments

In recent years, governments throughout the world have come to recognise the benefits of FOSS.[35] A study by The MITRE Corporation on behalf of the United States Department of Defence was cautiously optimistic, concluding that the open source model

> encourages significant software development and code reuse, can provide important economic benefits, and has the potential for especially large direct and indirect cost savings for military systems that require large deployments of costly software products.[36]

Taiwan has an open source project supported by the National Science Council and Ministry of Education, which examines the use of open source products to reduce royalty payments for office software in government agencies and schools.[37]

Due to the high regard for privacy in Europe, the German government is supporting an open source personal encryption utility, GNUPG, to reduce reliance on proprietary privacy-enhancing code such as PGP.[38] The Linux community has also entered a cooperative project with the Software Research Institute of the Chinese Academy of Sciences and NewMargin Venture Capital, a venture arm of the Chinese government, called Red Flag.[39] Initially, it developed a localised operating system for servers, but now incorporates developments for PCs, PDAs

[34] FSF, *Why 'Free Software' is Better than 'Open Source'* (5 May 2005) The GNU Project <http://www.gnu.org/philosophy/free-software-for-freedom.html>.

[35] See generally Robert W Hahn (ed), *Government Policy toward Open Source Software* (2002) <http://www.aei-brookings.org/publications/abstract.php?pid=296>. For an excellent overview of government policy and legislative activity regarding FOSS, see Centre for Strategic and International Studies, *Government Open Source Policies* (2004) <http://www.csis.org/tech/OpenSource/0408_ospolicies.pdf>.

[36] Carolyn Kenwood, *A Business Case Study of Open Source Software* (2001) xxv <http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf>; see also The MITRE Corporation, *Use of Free and Open-Source Software (FOSS) in the US Department of Defence* (2003) <http://www.egovos.org/rawmedia_repository/588347ad_c97c_48b9_a63d_821cb0e8422d?/document.pdf>.

[37] Tiffany Kary, *Taiwan Opens Door to Open Source* (4 June 2002) ZDNet News <http://news.zdnet.com/2100-3513_22-931885.html>.

[38] FSF, *The GNU Privacy Guard* (1 March 2004) GNUPG <http://www.gnupg.org>.

[39] Red Flag Software, *About Us*, Red Flag Linux <http://www.redflag-linux.com/egyhq.html>.

and China's computerised lottery system.[40] The Peruvian Parliament has a Bill before it to mandate the use of open source products in government offices.[41] David Nuñez, a Peruvian Congressman, circulated a letter to Microsoft on the internet that sparked much debate on the relative merits of free and open code as opposed to proprietary development.[42] There are many more examples of governments moving towards open source solutions, including South Africa, Brazil, Spain, Finland and India.[43]

The Gartner Report identifies five key factors that have influenced and heightened public interest in FOSS:

1 public sector organisations must reconcile budget reductions with annual software price increases of up to 30 per cent. Alternative licensing arrangements reduce up-front costs, making open source software an attractive option;
2 supporters of open source software have been increasingly vocal amidst a more technologically-literate community, while proprietary licensing schemes have received negative publicity;
3 antitrust litigation against Microsoft and other large software vendors has engendered negative sentiments toward what is perceived as a monopoly over United States-based commercial software;
4 the World Trade Organization has introduced penalties for member countries that fail to prosecute piracy. Governments unable to enforce software copyright internally due to cultural factors may use open source software as a compliance strategy; and
5 finally, the range of open source software solutions has expanded dramatically, and now encompasses a growing selection of tools with substantial organisational support.[44]

More broadly, the four key factors most commonly cited as motivating the adoption of FOSS in government are cost, open standards, security, and benefit to the community.

[40] Ibid.

[41] Software Libre en la Administración Pública (Proyecto de Ley N° 1609) 2002 (Peru).

[42] Thomas Greene, *MS in Peruvian Open-Source Nightmare* (19 May 2002) The Register <http://www.theregister.co.uk/2002/05/19/ms_in_peruvian_opensource_nightmare>. See also Dee-Ann LeBlanc and Stacey Tipton, *Ending Microsoft FUD: An Interview with Peruvian Congressman Villanueva* (21 May 2002) Linux Today <http://linuxtoday.com/news_story.php3?ltsn=2002-05-20-006-26-IN-LF-PB>.

[43] See Kenneth Wong, *Free/Open Source Software and Governments: A Survey of FOSS Initiatives in Governments* (2003) International Open Source Network <http://opensource.mimos.my/fosscon2003cd/paper/full_paper/kenneth_wong.pdf>; Open Software Working Group (South Africa), *Free/Libre and Open Source Software and Open Standards in South Africa* (2004) National Advisory Council on Innovation <http://www.naci.org.za/pdfs/floss_v2_6_9.pdf>; Government Information Technology Officers' Council, *Using Open Source Software in the South African Government: A Proposed Strategy* (2003) Open Source Software in Government <http://www.oss.gov.za/docs/OSS_Strategy_v3.pdf>.

[44] Gartner Group, above n 7, 12. The Report's major recommendation was that the Australian Taxation Office should develop a policy on free and open source software: at 7.

A *Cost*

The first benefit associated with FOSS is that it may reduce the total cost of software ownership. Free software does not, for example, require a product key for every computer, user or site. Thus, as Eben Moglen points out, the provider of a 'fully redistributable system containing only free software … can reduce the unit cost of software to zero',[45] leaving the customer to pay only for installation and support.[46]

The development model of FOSS is also more efficient than that of proprietary software. A free software developer can reuse code that was written for any other project, by any other developer, rather than having to start from scratch each time a particular solution is required.

A key benefit of using and contributing to FOSS is that in many cases the required infrastructure already exists. Taking an existing project and having publicly-funded programmers make required changes carries no expense other than the developers' wages; the distribution channels, project management software and developer base are already in place.[47] When creating new projects, such tools need to be established, but their practical ubiquity and standard interfaces make deployment simple and keep training costs low.

In either case, interested developers from around the world can have the opportunity to aid the government in developing software that will benefit all involved, at no cost to the government. However, cost should not be the only factor governments use to evaluate software solutions. The emphasis of the free software movement is on freedom, not price.

[45] Eben Moglen, *Free Software Matters: Free Government, II* (30 October 2002) <http://emoglen.law.columbia.edu/publications/lu-24.html>.

[46] There is considerable debate as to whether the total cost of ownership ('TCO') of GNU/Linux systems is less than that of Microsoft systems, particularly due to training and ongoing support costs; but see Robert Frances Group, *Total Cost of Ownership for Linux in the Enterprise* (2002) <http://www-1.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf>, which found that TCO for Microsoft web servers is 2.3 times greater than GNU/Linux servers over a three year period. See also CyberSource Pty Ltd, *Linux vs Windows: Total Cost of Ownership Comparison* (4 January 2004) <http://members.iinet.net.au/~cybersrc/about/linux_vs_windows_tco_comparison.pdf>, which found that there was a 36 per cent saving over three years for organisations migrating existing hardware to GNU/Linux, and a 26 per cent saving where new hardware is purchased. *Contra* Julia Giera and Adam Brown, Forrester Research, *The Costs and Risks of Open Source: Debunking the Myths* (2004) <http://download.microsoft.com/download/7/d/0/7d059de9-1557-415c-8332-920db6f89e44/FRSTRossCosts0404.pdf>, which demonstrated Linux planning costs to be between 5 and 20 per cent higher, and training costs 15 per cent higher, than the Microsoft equivalents; Laura DiDio, The Yankee Group, *Linux, UNIX and Windows TCO Comparison, Part 1* (2004) 1 <http://download.microsoft.com/download/6/b/7/6b7c5fa1-fcc9-434e-b1e6-5025b7f97786/YankeePart1.pdf>, which claims that for large enterprises, 'a significant Linux deployment or total switch from Windows to Linux would be three to four times more expensive and take three times as long to deploy as an upgrade from one version of Windows to newer Windows releases'. Expected costs of a technological rollout will be particular to any given organisation, and must be carefully evaluated in each case.

[47] Free project management software includes version tracking tools like Concurrent Versions System and bug tracking software like Mozilla's Bugzilla. See FSF, *CVS — Open Source Version Control* The GNU Project <http://www.nongnu.org/cvs>; The Mozilla Organisation, *Home — Bugzilla* Bugzilla.org <http://www.bugzilla.org>.

B *Open Standards*

Many advocates argue that open standards are crucial in any government acquisition of software.[48] Open standards are file formats and communication protocols which are agreed upon by community consensus and not controlled by proprietary companies. The adoption of open standards guarantees future access to current data, even if the hardware and systems used to create that data become obsolete. It also entails that governments neither mandate the use of a particular vendor's systems (by communicating through a proprietary format) nor lock future generations into using the same proprietary systems. The standard is always published, so users are free to comment, criticise, and modify it while knowing precisely what information is being stored.

While open standards do not equate exactly to open source, open source software is more likely to use open standards because of the public consultation inherent in the development process.[49] Pursuant to the Australian government's support for increased interoperability between software standards, the government aims to adopt open standards so as to 'ensure that Australian Government ICT systems interoperate in a trusted way with partners from industry and other governments'.[50] Integral to such interoperability is the use of software with publicly-accessible source code and documentation.

C *Security*

Another benefit of free software relates to the positive effect that source code availability has on software security. Bruce Shneier argues that security is better served by full disclosure of vulnerabilities and fast release of patches[51] — characteristics shared by open source software. In proprietary software, vulnerabilities are harder to identify because there is no public access to the source code. Further, once a vulnerability is discovered, only the licensor has the power to patch the hole; users must wait for an official update. Con-

[48] See, eg, Sebastian Rahtz, *Developing an Open Source Policy* (2005) The University of Manchester: Manchester Computing <http://www.mc.manchester.ac.uk/eunis2005/medialibrary/papers/paper_134.pdf>.

[49] Christoph Engemann points out that if, as Lawrence Lessig argues, code is law and the technological choices we make have a binding effect on our behaviour, then that code should be readable and accessible, as law should be. The accessibility of law is one of the basic foundations of its legitimacy: see Christoph Engemann, 'Electronic Government und die Free Software Bewegung: Der Hacker als Avantgarde Citoyen' in Daniel Gethmann and Markus Stauff (eds), *Politiken der Medien: Medien als Kriegs- und Regierungstechnologien* (2004); Lawrence Lessig, *Code and Other Laws of Cyberspace* (2000) <http://www.code-is-law.org>.

[50] AGIMO, *Open Source Software* (2004) <http://www.agimo.gov.au/__data/assets/pdf_file/35495/Open_Source_Software_Statement.pdf> ('AGIMO OSS Position Paper'). For a further explanation of the interplay between interoperability and open standards, see also at 2:

> The Australian Government is among the leaders internationally in the field of interoperability based on the use of open standards, an area connected with open source software. The Australian Government already has in place the '*Government Interoperability Technical Framework*', developed in consultation with an expert group of information technology architects drawn from key agencies. The Framework specifies technical standards that will enable Australian Government ICT systems to communicate and exchange information regardless of the type of technology used.

[51] Bruce Schneier, *Full Disclosure* (15 November 2001) Crypto-Gram Newsletter <http://www.schneier.com/crypto-gram-0111.html>.

versely, free software benefits by having greater public scrutiny of the source code, faster release times, and, if necessary, the problem can even be fixed by its users.

A recent study of the use of FOSS in the United States Department of Defense identified that free software was vital to information security in three ways:

1 the free software community has 'produced infrastructure software … with low rates of software failure combined with early and rapid closure of security holes, which makes such systems useful as the security linchpins in broader security strategies';[52]
2 the communities have had a 'long-term fascination with developing more and more sophisticated applications for identifying and analyzing security holes in networks and computers, resulting in [free and open source] products … that are invaluable to in-depth analyses of security risks';[53] and
3 free software 'contributes to security by making it possible to change and fix security holes quickly in the face of new modes of cyberattack. This ability, which allows rapid response to new or innovative forms of cyberattack, is intrinsic to the FOSS approach and generally impractical in closed source products'.[54]

William Caelli argues that since software cannot be trusted to be secure, users — and particularly governments — must be able to examine the workings of software systems to be satisfied of their security and be able to implement tougher security measures where the system is found to be vulnerable.[55] Caelli further argues that 'open source licensing represents the ideal for the evaluation of the underlying security architecture in the operating system and the allied mechanisms that activate and support necessary hardware security features',[56] and that '[r]easonable prudence would thus suggest movement towards an open source solution.'[57]

D *Providing Information Resources to the Community*

Finally, free software provides a framework whereby the benefits of publicly-funded software development can be returned to the public. When a government develops publicly-funded software, there is a strong argument that, subject to issues of security and confidentiality, the software should be made available to the public. While not all internally-developed software may be

[52] The MITRE Corporation, above n 36, 20.
[53] Ibid.
[54] Ibid. See also a memorandum by John Stenbit, The Centre of Open Source and Government, *Open Source Software (OSS) in the Department of Defense (DOD)* (28 May 2003) <http://www.egovos.org/rawmedia_repository/822a91d2_fc51_4e6e_8120_1c2d4d88fa06?document.pdf>.
[55] William J Caelli, 'Security with Free and Open Source Software' in Brian Fitzgerald and Graham Bassett (eds), *Legal Issues relating to Free and Open Source Software* (2004) 89, 112.
[56] Ibid.
[57] Ibid 113.

suitable for public release, use of free software may provide a framework to release code to the commons without attracting liability or requiring further expenditure to support the software.

In this regard, Russell Pavlicek questions the windfall that private organisations reap from closed-code arrangements with governments:

> Which is more deplorable: that a few profit-making software companies won't be able to make as much profit from publicly funded software, or that the public who already paid for the software once with their tax dollars will have to pay for it again when the large software company puts it into their closed-source product?[58]

Where the line will be drawn between the use of GPL and proprietary software in government requires an assessment of the government's role in society. Some argue:

> The principal role of government and universities in the ecosystem is to undertake basic research and to dispense the findings both into the societal base of technical knowledge and to private enterprises and individuals capable of developing these innovations commercially.[59]

On the other hand, it could be said that the role of government is to maintain the public good. By exercising their discretion in acquiring or creating software solutions, governments can pass on benefits to the public. This has the effect of cheaply increasing access to technology and the intellectual commons.

There is evidence that the Australian government may be rethinking the strong controls it presently retains over copyright under Australian law,[60] as the Copyright Law Review Committee has been given a reference to inquire into the legitimacy of, and the continuing role for, Crown copyright.[61] It has been suggested that a sensible outcome would be for the Crown to retain copyright in material it develops, but take advantage of open licensing schemes where appropriate.[62] This would maximise both economic advantage and public access to information.

Lawrence Lessig argues that governments are so entranced by the minimalist role they supposedly ought to play in a free market economy that they have

[58] Russell C Pavlicek, *Don't Fear the GPL* (23 August 2002) InfoWorld <http://www.infoworld.com/article/02/08/23/020826opsource_1.html>.

[59] Microsoft Corporation, *What Is Microsoft's Concern with the GNU General Public License (GPL)?* (1 March 2005) Microsoft Shared Source Initiative <http://www.microsoft.com/resources/sharedsource/initiative/faq.mspx>.

[60] The Commonwealth and states of Australia are owners of copyright in original works made by, or under the direction and control of, the Commonwealth or the state: *Copyright Act 1968* (Cth) s 176. The Crown also retains copyright in material first published in Australia under its direction, and is bound by the express words of the copyright statute: ss 7, 177. However, provisions relating to Crown ownership may be modified by agreement: s 179.

[61] Copyright Law Review Committee, *Terms of Reference — Crown Copyright* (5 December 2003) <http://www.clrc.gov.au/agd/www/Clrhome.nsf/Page/RWP3C2E5B1D1B98D6FACA256DE3000E9471>.

[62] See Brian Fitzgerald, *Submission to the Copyright Law Review Committee on Crown Copyright* (2004) Copyright Law Review Committee <http://www.clrc.gov.au/www/clrHome.nsf/Page/Present_Inquiries_Crown_copyright_Submissions_2004_Sub_No_17_-_Professor_Brian_Fitzgerald>.

become blind to the benefits of government intervention. He calls on governments to take a more active role:

> When government steps aside, it is not as though nothing takes its place. When governments disappear, it is not as if paradise prevails. It is not as if private interests have no interests, as if private interests do not have ends that they will pursue. To push the anti-government button is not to teleport us to Eden. When the interests of governments are gone, other interests take their place. Do we know what those interests are? And are we so certain they are better?[63]

Rod Dixon suggests that policy makers need to understand the way in which 'open source software development poses alternative explanations of human motivation for creative endeavors, which can be ignored or used to augment our public policy choices.'[64] Once the manner in which software flows through a socio-technical network is understood, policy makers should consider whether the traditional model for the deployment of software in government and associated intellectual property management is the best model to promote government policy in the 21st century.[65]

This is not to suggest that closed development or acquisition of 'off-the-shelf' software has no place in government. In each case, there should be an honest evaluation of which path will yield the best results. It is suggested, however, that where public funds are used to develop software, that software should ordinarily be returned to the public. There must be strong reasons to justify a decision not to release source code, such as confidentiality concerns or the presence of a large economic market for the closed product which should be exploited.

IV GOVERNMENT PROCUREMENT AND SUPPLY OF FREE SOFTWARE

Having determined that there are benefits in employing FOSS, how does government go about procuring it? There has been much debate over the past few years suggesting that standard government procurement practices are biased against FOSS, and that a level playing field needs to be established.[66] To overcome any suggested limitations in the procurement process, governments

63 Lawrence Lessig, 'Keynote Address: Commons and Code' (1999) 9 *Fordham Intellectual Property, Media and Entertainment Law Journal* 405, 418.

64 Rod Dixon, *Open Source Software Law* (2004) 122.

65 Ibid. Recently there has emerged an even stronger argument, free software as democratic principle, which suggests that core software infrastructure in a society — especially that relating to government, politics, law and/or criminal justice, including voting machine or electronic court software — should be open code. This argument is based on the assumption that software, like discourse, may contain bias and implement discrimination, and that to guard against this we need to be able to see what the software is doing. It has been argued that only open source software can guarantee the scrutiny and accountability of 'software as discourse' that a democracy requires: see generally Engemann, above n 49; Fitzgerald, 'Software as Discourse', above n 11; Brett M Frischmann, 'An Economic Theory of Infrastructure and Commons Management' (2005) 89 *Minnesota Law Review* 917.

66 See, eg, Kate Lundy, 'Open Source Software: Providing Greater Security and Innovation in the Delivery of E-Government Services' (Speech delivered at the Open Source Software Conference, Sydney, 11 September 2003) <http://www.katelundy.com.au/opensourceegovt.htm>; David A Wheeler, *Why Open Source Software/Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!* (9 May 2005) David A Wheeler's Personal Homepage <http://www.dwheeler.com/oss_fs_why.html#governments>.

throughout the world have responded by reassessing their procurement practices. Some have chosen to ensure the effectiveness and equity of their procurement processes by restating administrative guidelines,[67] while others have used legislation to place open source software on an equal footing.[68]

In Australia, the Commonwealth government has acknowledged the opportunities for innovation and other benefits that FOSS might provide.[69] While the Commonwealth government has no intention of enacting a legislative requirement that its agencies specifically consider open source software in procurement processes, it has taken steps to ensure that its administrative procedures equitably facilitate the procurement of all types of software.[70] As Senator Helen Coonan, Minister for Communications, Information Technology and the Arts, explained following the release of AGIMO's *Open Source Software* discussion paper in August 2004,[71] the government

> seeks to clear the air on the debate and provide some factual information on [its] approach to open source software and how [it is] acting to provide a level playing field for all suppliers of software solutions to government.
>
> As well as [the] position paper, the Government is preparing a range of tools to help government agencies evaluate emerging open source solutions against more familiar proprietary software on an informed basis and appropriately assessing value for money and fit for purpose.[72]

The message is that Australian government procurement policies 'allow agencies to use whatever software is available providing it meets agencies' needs and is cost effective as a business solution.'[73]

A *Government Procurement Practices*

By way of contrast, the Australian Democrats have lobbied for (and, in the Australian Capital Territory, obtained) legislative support for a more level

[67] See, eg, Stenbit, above n 54.

[68] See, eg, Financial Management and Accountability (Anti-Restrictive Software Practices) Amendment Bill 2003 (Cth); *Government Procurement (Principles) Guideline Amendment Act 2003* (ACT) ('*Government Procurement Act*'); State Supply (Procurement of Software) Amendment Bill 2003 (SA). For a comprehensive list of bills and legislation purporting to regulate government procurement procedures, see Grupo de Usuarios de Software Libre de Córdoba, *Legislation on the Use of Free Software in the Government* (29 November 2004) References for Official Actions regarding Free Software <http://www.aful.org/politique/perou/english/referencias.html>.

[69] AGIMO, *Better Services, Better Government* (2002) 20 <http://www.agimo.gov.au/publications/2002/11/bsbg>.

[70] See *Financial Management and Accountability Act 1997* (Cth); *Commonwealth Procurement Guidelines* (2005) <http://www.finance.gov.au/ctc/commonwealth_procurement_guide.html>. See also AGIMO, *Open Source Software*, above n 50.

[71] See AGIMO, *Open Source Software*, above n 50.

[72] Senator Helen Coonan, 'Government Leads the Way on Open Source Software' (Press Release, 31 August 2004) <http://www.agimo.gov.au/media/2004/08/35491.html>.

[73] AGIMO, *Open Source Software,* above n 50, 1. See also *Financial Management and Accountability Act 1997* (Cth); *Commonwealth Procurement Guidelines* (2005) <http://www.finance.gov.au/ctc/commonwealth_procurement_guide.html>; AGIMO, *A Guide to Open Source Software for Australian Government Agencies*, above n 8, 27–8.

playing field.[74] In late 2003, the Australian Democrats attempted to legislate consideration of open source software for public agency procurement contracts in various jurisdictions. An initial and unsuccessful attempt to legislate at a state level in South Australia[75] was refined and presented as a Bill to the federal Parliament, but this was also unsuccessful.[76] Only the Australian Capital Territory passed the Bill, in December 2003. The *Government Procurement Act* inserts s 6A into the *Government Procurement (Principles) Guideline 2002* (ACT).[77]

The Act requires government entities within the Australian Capital Territory to consider open source software and avoid 'software that does not comply with open standards'[78] or 'for which support or maintenance is provided only by an entity that has the right to exercise exclusive control over its sale or distribution.'[79]

The Act explicitly states that software does not comply with open standards unless

> the specifications for data representations used by the software (including, for example, file formats for data storage, transmission and network protocols) are completely and accurately documented and available to the public for use, application or review without restriction.[80]

It ties the definition of open source software to that of the OSI[81] and operates subject to a three year sunset clause.[82] The Act and proposed Bills aim to address concerns that 'a small number of software manufacturers have a disproportionate and restrictive hold on the supply, use and development of software.'[83]

The Initiative for Software Choice ('ISC') opposed the Australian Democrats' legislation. In response to the earlier Bill proposed by the Australian Democrats in the South Australian Parliament, the ISC wrote a letter to the state Premier, Mike Rann, stating:

> The ISC strongly supports the development and adoption of all kinds of software — [open source software], hybrid and proprietary. All models have a

[74] See Simon Hayes, 'Democrats to Make Money on Open Source Gig', *The Australian* (Sydney), 23 September 2003, 28.

[75] State Supply (Procurement of Software) Amendment Bill 2003 (SA).

[76] Financial Management and Accountability (Anti-Restrictive Software Practices) Amendment Bill 2003 (Cth). Originally introduced on 18 August 2003, the Bill was reintroduced into the Senate on 17 November 2004, following the 2004 federal election.

[77] For a more detailed examination of the Australian Capital Territory Act and the Commonwealth and South Australian Bills, see Ian Oi, 'Open Source and the Public Sector: Challenges in the Development and Implementation of Policy and Law' (Paper presented at the Linux and Open Source in Government: The Challenges conference, Adelaide, 12–13 January 2004) <http://www.linux.org.au/conf/2004/eventrecord/LCA2004-cd/mini/OS_in_govt/Ian_Oi_112348590_IO_0104_Adelaide_Paper_171203.pdf>.

[78] *Government Procurement Act* s 6A(1)(b)(i).

[79] *Government Procurement Act* s 6A(1)(b)(ii).

[80] *Government Procurement Act* s 6A(3).

[81] *Government Procurement Act* s 6A(4).

[82] *Government Procurement Act* s 6A(5).

[83] Financial Management and Accountability (Anti-Restrictive Software Practices) Amendment Bill 2003 (Cth) sch 1.

> place in the highly competitive software market. Only in this manner, through vibrant and open competition, does the whole of the market thrive, and consumers — both public and private — reap tremendous benefits.
>
> Standing in stark contrast to open competition are state-mandated software preferences. These 'preference' policies strip merit out of the process by using access to source code as a proxy for ICT project success …
>
> The result would be reduced options for software acquisitions, largely eliminating proprietary offerings that might be the best solutions for the given need. Additionally, constituents would suffer because the best solutions could never truly be acquired, with at least one development model — proprietary software — being restricted from agency consideration. Further, South Australia's primarily [proprietary] ICT industry would be harmed because of foreclosed access to important state market opportunities.[84]

The ISC group is reported as saying that such government mandates would be a barrier to free trade agreements.[85] Democrat Senator Brian Greig, who proposed the Bill, rebutted these claims, specifically referring to groups such as ISC.[86] Senator Greig pointed out that many current government systems, often unwittingly, mandate use of proprietary systems because software procurement choices have not considered open source alternatives and will not work with open formats or open source software. Greig argued:

> The forces of proprietary software and their supporters have tried to portray this bill as being protectionist in nature, one that tries to pick software favourites. It is in fact the complete opposite. Currently, we have a system that is largely based on proprietary formats, a system that does pick favourites. Removing this and opening up the playing field to all, is the raison d'être for this bill.[87]

Senator Greig points out that when the Thai government mandated use of open source software, it was able to acquire both hardware and software for a price similar to that which it previously paid for Microsoft's software licences alone. The result was that Microsoft dramatically reduced its prices in order to stay competitive in the government contract area. Greig claims that Microsoft would recoup lost revenue when they provided upgrades: 'Microsoft's actions echo the words of Henry Ford when he offered to give away his cars provided he could keep the monopoly on spare parts. It is this type of monopoly that the use of proprietary formats maintains'.[88] The key was to obtain, and then be able to control, the contract. By mandating consideration of open source alternatives, the Thai government removed what was previously a barrier to efficient market processes — ironically, a barrier erected by a policy of economic non-interference.

[84] Letter from Bob Kramer, Executive Director of the Initiative for Software Choice, to the Honourable Mike Rann, 10 June 2003 <http://softwarechoice.org/download_files/DearSouthAustraliaRann.pdf>.

[85] Simon Hayes and James Riley, 'Open Source Trade Clash', *The Australian* (Sydney), 1 July 2003, 25.

[86] Commonwealth, *Parliamentary Debates*, Senate, 18 September 2003, 15 520 (Senator Brian Greig).

[87] Ibid 15 520–1.

[88] Ibid.

Irrespective of whether it has occurred through legislative or administrative processes, there can be little doubt that governments are now more aware of the intricacies associated with procuring computer software. Emerging from this discussion is the need to effectively provide for the equal consideration of FOSS alternatives.

B *Indemnities for Title and Warranties for Performance*

Regardless of whether a government agency contracts for supply or creation of free software, it should consider whether it needs indemnities against claims of intellectual property infringement from third parties. When contributions are made by community-based developers to a project controlled by a government agency, it will often be useful to require a declaration by each developer that they have written the code themselves or acquired it on a compatible licence. This is common practice for the large free software development groups. Jeremy Malcolm considers it sensible for developers to assume the risk when developing open source software because they are in the best position to ensure that the code does not infringe the intellectual property rights of any other persons.[89]

Where a government agency enters into a contract for the supply of free software from a large vendor, it would be prudent to seek both indemnities for title, and warranties that the software will work (and continue to work) as required and that the software will be repaired or replaced if required.[90] In most cases, indemnities, warranties and continuing support agreements provide the only reason to enter into a supply contract with a large vendor. If they are not required, deployment and training can be undertaken in-house or through a smaller technical organisation. Risk assessment should be undertaken before any supply contract is entered into, whether the supply is for open or closed source software. If these indemnities and warranties are required, it will obviously be important to assess whether the proposed supplier has the means to fulfil its potential obligations.

C *Requirements of the Australian* Trade Practices Act

Many free and proprietary software licences purport to disclaim all warranties, whether express or implied, in order to avoid the possibility of free software developers being held liable for any fault in the program. In Australia, the *Trade Practices Act 1974* (Cth) ('*TPA*') provides certain non-excludable warranties where a corporation is carrying on a business. The *TPA* applies to the Australian government and an 'authority of the Commonwealth' when either is carrying on a business, but only Commonwealth authorities can be fined or prosecuted.[91] The *TPA* will be of significance when a government is either a consumer/procurer or developer/supplier of software.

[89] See Jeremy Malcolm, 'Could *SCO v IBM* Happen to You?' (Paper presented at the Linux Conference Australia, Adelaide, 15 January 2004) <http://www.ilaw.com.au/public/scopaper.html>.

[90] AGIMO, *A Guide to Open Source Software for Australian Government Agencies*, above n 8, 29.

[91] *TPA* s 2A.

The *TPA* establishes several consumer protection measures. Importantly, it prohibits misleading or deceptive conduct[92] and the making of false or misleading representations,[93] and implies warranties as to title and quiet enjoyment.[94] It also imports requirements that goods will be fit for the purpose supplied, of merchantable quality, and, if supplied by reference, will correspond with the sample.[95] Finally, the Act sets conditions that services will be rendered with due care and skill and be fit for purpose.[96] These implied conditions and warranties cannot be excluded by contract.[97] Most of these provisions apply when a corporation[98] is supplying goods or services to a consumer in 'the course of a business'.[99] Peter James notes that

> [w]here software is supplied by way of gift, not sale, this requirement nevertheless would be satisfied if the software supply is part of a commercial dealing or if the supply is connected (even indirectly) with advancing or protecting the commercial interests of the supplier.[100]

This means that the implied conditions will generally only apply to suppliers of free software, and not individual developers.

If a government or government agency begins to engage in a commercial or a related supply of software to consumers, it must be aware that these provisions impose certain minimum levels of quality upon any software it provides, as well as regulating the manner in which that software is represented. On the other hand, if the government developer merely contributes to an open source project outside of a business relationship, no liability could arise.

Due to the loose wording of exclusion clauses found in free software licences, they may not be effective in limiting liability for negligence and consequential damages. Peter James notes that the 'courts look at the provision as a whole and, if the exclusion attempts to limit liability for the very purpose of the contract, it will need to be clearly and unambiguously drafted to survive challenge',[101] which the GPL is not. For these reasons, anyone supplying software under the GPL and similar licences may be liable for damages not only for direct losses, but also for consequential losses — including loss of profits or data — unless they adequately modify the relevant clauses in the GPL.[102]

[92] *TPA* s 52.
[93] *TPA* s 53.
[94] *TPA* s 69.
[95] *TPA* ss 70–2.
[96] *TPA* s 74.
[97] *TPA* s 67.
[98] 'Corporation' is defined in *TPA* s 2A to include the Commonwealth and authorities of the Commonwealth where they carry on a business.
[99] Note the extended operation of these sections given by *TPA* ss 5–6, which provide that in some instances the provisions will have effect even though neither of the parties is a corporation.
[100] Peter C J James, 'Open Source Software: An Australian Perspective' in Brian Fitzgerald and Graham Bassett (eds), *Legal Issues relating to Free and Open Source Software* (2003) 63, 78, citing *Fasold v Roberts* (1997) 70 FCR 489.
[101] Ibid 80.
[102] For an attempt to overcome this problem in relation to the BSD licence, consider the proposal of National Information and Communications Technology Australia ('NICTA') to 'port' the BSD

V Government as a Developer of Free Software[103]

When a decision is made to use free software for a government function, consideration must be given to whether the software should be developed internally, outsourced to hired contractors, or built upon existing software and customised by another supplier. Obviously, if the software required is already available in a form that is usable by the government, such as an office suite or desktop environment, governments should take advantage of the pre-existing code and have installation and training carried out by in-house staff or commercial vendors. However, where a substantial portion of the software needs to be created, it will be necessary to consider whether the department is capable of supporting its development and maintenance. There is also considerable momentum for the creation of shared government code repositories, so that one agency can create (or commission) a piece of software that is flexible enough to be reused by other agencies ('white-branding') and make it available for reuse.[104]

Section 176 of the *Copyright Act 1968* (Cth) provides that the Commonwealth and states are owners of copyright in original literary works made by, or under the direction and control of, the Commonwealth or the state. Effectively, the Crown will own the copyright in both the software it creates in-house and the software it causes to be created by contractual developers, subject to any agreement to the contrary.[105] Whether development is outsourced or not, the government should stipulate how the development is to take place. The majority of development could be completed by one group of developers, and released as free software after completion, or the core group of developers could act as a development hub for community-based free software developers. Each methodology has its own advantages and disadvantages.

Where development is completed by a core in-house or outsourced group, without the aid of other members of the free software community, the development will be easier to manage. Schedules and costs can be more accurately planned, features can be implemented in proportion to their importance to the government agency, necessary sensitive information can be made available to a select group only, and the product can be made available to the public at a stage where it is stable and has been cleansed of any sensitive information. On the other hand, the developers would lose access to some of the benefits of open source development — principally, the way in which work is distributed over a

licence to the Australian legal system: NICTA, *Australian Public Licence B (OZPLB) Version 1.0* (2004) <http://nicta.com.au/ozplb_diff.cfm>.

[103] See above Part III(B) and consider the points made concerning the government as developer in relation to indemnities and warranties and the operation of the *TPA*.

[104] For example, the Australian government has released MySource Matrix, a white-branded content management system, which is made available at no cost to government agencies and non-profit organisations. It has been argued that this white-branding is not free software on the basis that, first, the licence for the core system requires that notification be given for modifications and copyright in modifications must be assigned to the developer; and, second, the extra modules distributed by the Australian government are licensed under proprietary licences: see Eric Abetz, 'AGIMO Open Source Solution "Whitebranded"' (Press Release, 27 April 2005) <http://www.agimo.gov.au/media/2005/april/42223.html>; Brendon Chase and Renai LeMay, *Government OSS Not Really Open: Lawyer* (3 May 2005) ZDNet Australia <http://www.zdnet.com.au/news/software/0,2000061733,39190311,00.htm>.

[105] *Copyright Act 1968* (Cth) s 179.

broad developer base. Distributed development can provide not only cheap labour but potentially also a more productive and inventive team, leading to more efficient, secure code. Additionally, if the government's intention is not to release code until after development has been completed, and the developing agency is building upon GPL-licensed code, it must take care to avoid earlier distribution of the software in order to prevent the obligation to distribute source code from arising.[106] This is particularly important when testing or evaluation versions of the software are provided.

Alternatively, software can also be effectively developed through government-sponsored, community-based development. According to this methodology, the core (in-house or contracted) development group forms the nexus for development, providing the framework, momentum and guidance to a wider community of free software developers. However, one of the major disadvantages of this approach is the extra overhead associated with managing a large distributed community, whose aims and schedules do not always align with those of the agency. Accordingly, this methodology is probably best applied to large projects that are likely to be of immediate use to a large number of people, where interested and motivated developers can provide substantial help in development.[107]

Finally, government agencies might also contribute to and customise a pre-existing free software project, making any required changes without taking control of the development process. Under this approach the agency would not be responsible for management of users, but would be able to build upon pre-existing work, extend the software to meet its requirements, and give the end product back to the community as a simple, one-off gift.

A *The Obligation to Redistribute Source Code*

The obligation to redistribute must be clearly understood by any user of free software. If a government decides to use free software, it must be aware of the circumstances in which it will be obliged to disclose its modifications to the program source code. For restrictive free licences like the GPL, a government will be obliged to disclose the source for any derivative works it makes and distributes.[108]

Due to uncertainties in the licence (the effect of which will be examined below),[109] it is not clear exactly when a derivative work will be created. Modifications to the software are clearly derivative works and will be treated as such. The difficulty lies in determining when new programs, which simply make use of free software or are designed to operate with free software — in short, mere use of licensed code — will be treated as derivative works. Stallman argues that any use of code released under the GPL by another program creates an obligation

[106] FSF, *The GNU General Public License: Version 2.0*, above n 5, cl 3(a).

[107] See Jan Sandred, *Managing Open Source Projects: A Wiley Tech Brief* (2001) 37; Benkler, above n 6; David McGowan, 'Legal Implications of Open Source Software' [2001] *University of Illinois Law Review* 241; Raymond, above n 26.

[108] FSF, *The GNU General Public License: Version 2.0*, above n 5, cls 2(b), 3(a).

[109] See below Part V(B).

upon that other program.[110] However, because the GPL appears to carve out a set of rights from copyright law, it would appear that not all forms of incorporation are capable of giving rise to a derivative work. As Rosen argues, '[t]he primary indication of whether a new program is a derivative work is whether the source code of the original program was used, modified, translated or otherwise changed in any way to create the new program'.[111] However, he adds that '[t]he meaning of derivative work will not be broadened to include software created by linking to library programs that were designed and intended to be used as library programs.'[112] Accordingly, it is possible to create new software that uses and relies upon free software components without creating a derivative work.

The distinction, though fine, is important. If a program is a derivative of another work which is licensed under the GPL, any distribution of the new program must also be under the GPL. On the other hand, if the new software is not a derivative, the developer is free to release the software on any terms. For governments, this can be very important because it may oblige the release of sensitive or confidential information. To safely avoid disclosure, software that may contain or process such information should be designed to operate independently from any software licensed under terms which would compel disclosure of that information. In many cases, the sensitive parts of code can be built into separate modules, which do not form part of the main application, and therefore are not required to be licensed under the GPL.

Simply creating a derivative work, without more, will not give rise to an obligation to publish it under a free software licence. The derivative work must be 'distributed'. However, what constitutes a 'distribution' is not clear. It is apparent from the FSF's own comments that internal distribution within a single organisation will not be considered a 'distribution' under the GPL.[113] Similarly, Eben Moglen, general counsel for the FSF, takes the view that 'Federal Government agencies may share free software without making a "distribution".'[114] Thus, in Australia, sharing of code between federal government departments would probably not give rise to an obligation to make the source code available. The same might also be true for sharing between state government departments, and possibly between federal and state, or state and state governments.[115]

[110] Richard Stallman, *Why You Shouldn't Use the Library GPL for Your Next Library* (February 1999) The GNU Project <http://www.gnu.org/licenses/why-not-lgpl.html>.

[111] Lawrence Rosen, 'Derivative Works' [2003] *Linux Journal* <http://www.linuxjournal.com/article/6366>.

[112] Ibid. Library programs perform generic computational functions and are commonly distributed freely and in source code form together with a programming language or Application Programming Interface for use in software development by other programmers.

[113] FSF, *Frequently Asked Questions about the GNU GPL* (2005) The GNU Project <http://www.gnu.org/licenses/gpl-faq.html#InternalDistribution>.

[114] Email from Eben Moglen to Brian Fitzgerald, 3 December 2003.

[115] For such a sharing between federal and state governments not to be deemed a distribution, we would need to rely on the notion of Australia being 'one indissoluble Federal Commonwealth under the Crown', as stated in the Preamble to the *Australian Constitution*. However, this argument runs counter to the accepted constitutional jurisprudence that conceptualises each emanation or jurisdiction of the Crown as being a legal entity in its own right: *Bropho v Western Australia* (1990) 171 CLR 1, 22 (Mason CJ, Deane, Dawson, Toohey, Gaudron and McHugh JJ).

However, if the software is shared with or by a statutory corporation, there will be a stronger argument that a 'distribution' has taken place. Where a commercial body exists to carry out a public function, but is otherwise independent from the government, it is probable that any sharing of software between it and another such entity would not be taken to have been made between two parts of the Crown or government; rather, the presumption would arise that a distribution between two separate entities had taken place. Accordingly, any software that contains sensitive or confidential information, if it forms a derivative of any restrictive free software, can be shared between government departments without requiring disclosure of the source. Even so, care must be taken to avoid distribution to third parties, including statutory corporations.

Finally, on the subject of sensitive or confidential information, it must be made clear that merely using free software to create or store the information will never give rise to an obligation to disclose. The concern only arises when such information is used to create or modify the software itself and that information becomes embedded in the code. As such, an end user who does not modify source code will never be under such an obligation.

B *Enforceability of the GPL*

There is considerable debate over the enforceability of the GPL and whether it is to be construed as a licence or a contract.[116] Specifically, if it is a contract, is there valid consideration to create an enforceable contract? On the other hand, if it is considered to be a copyright licence, is it possible to enforce the requirements that users distribute any derivative works under equivalent terms? Ben Giles argues that since the only promise that a free software user makes is to redistribute under the GPL if and only if they *choose* to distribute a derivative work, that promise is not sufficient and there is no consideration to support a valid contract.[117] This argument rests on the doctrine of illusory consideration, which means that promises that are only to be carried out at the promisor's discretion cannot create a binding contract.[118] There has been no significant interpretation or modern restatement of this doctrine in Australian law. Arguably, due to significant changes to the way in which parties do business online, the doctrine has lost some relevance in recent years.

In contrast, Moglen suggests that the GPL is a copyright licence, not a contract: '[l]icenses are not contracts: the work's user is obliged to remain within the bounds of the license not because she voluntarily promised, but because she

[116] See, eg, Thomas Hoeren, *The First-Ever Ruling on the Legal Validity of the GPL — A Critique of the Case* (2004) Oxford Internet Institute <http://www.oii.ox.ac.uk/resources/feedback/OIIFB_GPL3_20040903.pdf>; Julian P Höppner, 'The GPL Prevails: An Analysis of the First-Ever Court Decision on the Validity and Effectivity of the GPL' (2004) 1 *SCRIPT-ed* 662 <http://www.law.ed.ac.uk/ahrb/script-ed/issue4/GPL-case.asp>.

[117] Ben Giles, '"Consideration" and the Open Source Agreement' (2002) 49 *Computers and Law Journal* 15, 16.

[118] *British Empire Films Pty Ltd v Oxford Theatres Pty Ltd* [1943] VLR 163, 167–8 (O'Bryan J). See also *Placer Development Ltd v Commonwealth* (1969) 121 CLR 353.

doesn't have any right to act at all except as the license permits.'[119] The exclusive rights of the copyright owner can be used to restrict reproduction, making a derivative work and distributing the software, and any user who does these things must do so in accordance with the terms of the licence.[120] Any obligations in the GPL that purport to do more than this will need to be supported by contractual consideration.

As yet, there has been no significant litigation concerning the enforceability and classification of the GPL, though in 2004 the Munich District Court issued a preliminary injunction against Sitecom Deutschland GmbH for alleged infringement of the GPL.[121] Moglen suggests that 'there have been no such controversies because nobody thinks they're going to win them'.[122] Maureen O'Sullivan notes that the threat of damage to a firm's reputation from the watchful open source community, as well as the possibility of a lengthy court case, has been successful over the last decade in ensuring that firms comply with the terms of the GPL.[123] It thus seems clear that even though the GPL has not been tested in court, questions about its technical legal enforceability are not barriers to its widespread use, because substantial compliance with its terms can be expected to continue well into the foreseeable future.

The other concern about free software licences is that a gratuitous licence can normally be revoked at will.[124] This means that, in the case where one single entity controls a significant portion of the copyright in the source code for a free software package, that entity may be able to terminate the licence and users will no longer be entitled to copy or redistribute the software. Jeremy Malcolm calls this 'one of the best kept secrets of the open source movement',[125] and notes the potential danger that an upstream developer could revoke the licence. This would cause all derived projects to be rendered invalid to the extent that they are derived from the original.[126] In practical terms, however, it would be hard for any single licensor to revoke a licence partially supporting a program — especially one which forms part of a large, distributed project.

[119] Eben Moglen, *Free Software Matters: Enforcing the GPL, I* (12 August 2001) <http://moglen.law.columbia.edu/publications/lu-12.html>. See also *Sun Microsystems Inc v Microsoft Corp*, 188 F 3d 1115, 1121 (Schroeder J) (9th Cir, 1999).

[120] Note that although these rights emanate from the international *Berne Convention for the Protection of Literary and Artistic Works*, opened for signature 14 July 1967, 828 UNTS 222 (entered into force 29 January 1970), their operation under Australian or other national law will be slightly different in terminology and effect.

[121] *Harald Welte v Sitecom Deutschland GmbH* (Unreported, Landgericht München I, Kaess, Müller and Rieger JJ, 19 May 2004) <http://www.jbb.de/urteil_lg_muenchen_gpl.pdf>. An English translation is available at <http://www.oii.ox.ac.uk/resources/feedback/OIIFB_GPL2_20040903.pdf>. See also Harald Welte, *Netfilter Project Was Granted a Preliminary Injunction against Sitecom GmbH* (15 April 2004) Netfilter Project <http://www.netfilter.org/news/2004-04-15-sitecom-gpl.html>.

[122] Moody, above n 15, 313.

[123] See Maureen O'Sullivan, 'Making Copyright Ambidextrous: An Expose of Copyleft' [2002] 3 *Journal of Information, Law and Technology* <http://www2.warwick.ac.uk/fac/soc/law/elj/jilt/2002_3/osullivan>.

[124] See *Trumpet Software Pty Ltd v OzEmail Pty Ltd* (1996) 34 IPR 481.

[125] Jeremy Malcolm, *Problems in Open Source Licensing* (Paper presented at the Linux Conference Australia, Adelaide, 24 January 2003) <http://www.ilaw.com.au/public/licencearticle.html>.

[126] Ibid.

In the event that a licence is revoked, it is likely that the doctrine of estoppel would prevent the copyright owner from asserting his or her rights. Equitable estoppel has been developed to prevent a person from unconscionably denying an expectation where they induce in another party (here the licensee) an assumption that a particular legal relationship exists between them, and that party subsequently acts, reasonably, in reliance upon that expectation.[127] If a licensor releases software under a free software licence, they are essentially inviting others to perpetually use, reproduce, modify and distribute that software. If another person does in fact make use of the software, and the original licensor purports to revoke the licence (a departure clearly to that person's detriment), the doctrine of equitable estoppel would arguably prevent the licensor from denying that the licence could not be revoked.[128] Again, to reach this stage in legal proceedings would be quite rare. While revocation may be technically possible, it is unlikely to occur in the face of public opposition and a vigilant open source community. Regardless, as has been demonstrated over the last 12 months by *The SCO Group Inc v International Business Machines Corp* litigation,[129] the developer community is more than willing to replace any code for which the licence has been revoked or that otherwise infringes copyright. For these reasons, the issue of revocability is much more a theoretical than a practical concern.

C *Layering and Combining of Licences*

There are many different FOSS licences, a number of which are nearly identical to the copyleft GPL or the permissive MIT/X11 and BSD licences. Unfortunately, some of the minor differences render them legally incompatible with other licences. This is particularly the case when combining code released under the GPL licence with code released under licences which are not considered to be GPL-compatible. The copyleft nature of the GPL will not allow further restrictions to be placed upon software that is derived from code released under the GPL. It prevents code forking under different licences, which means that downstream developers cannot take the benefits of free software and create a closed product. Accordingly, if another licence imposes additional restrictions, source code released under that licence cannot be combined with other source code licensed under the GPL. Such a licence is said to be 'GPL-incompatible'.

The problem is accentuated when a single distribution makes use of software that is licensed under a large number of free software licences. Peter James recognised this problem, noting the main licence groups in Red Hat Linux 7.1:[130]

[127] *Waltons Stores (Interstate) Ltd v Maher* (1988) 164 CLR 387, 428 (Brennan J).

[128] See generally *Commonwealth v Verwayen* (1990) 170 CLR 394. Further issues about the kind of remedy to be awarded (for example, specific performance as opposed to damages), and who could enforce the estoppel (for example, the direct recipient as opposed to downstream users), would also need to be considered.

[129] No 2:03CV294 DAK (D Utah, 1 July 2005) ('*SCO v IBM*').

[130] Red Hat Linux is a popular open source distribution of the Linux operating system.

there are more than 17 different licence types (as well as public domain software) governing different parts of the source code. The break down on licences is:

55%	GNU's General Public Licence (GPL)
10%	GNU's Lesser General Public Licence (LGPL)
9.4%	MIT open source licence (MIT)
7.5%	Berkeley Software Distribution licence (BSD)
6.8%	Mozilla Public Licence (MPL)[131]

There is usually no tension between free software licences where a software package is merely bundled together (as is the case in self-contained software such as Red Hat Linux). The problem arises where source code is combined from several sources or individual pieces of software in a way that creates a derivative work, or, much less often, is expressly forbidden by one or more of the software licences. Since the GPL is the most popular of the free software licences, releasing software under a licence that is incompatible with the GPL effectively places it out of reach for a large part of the free software community. It negates the benefits of code reuse and collaborative production.

Two principles are of paramount importance if governments are to successfully deploy free software. First, if they combine code released under the GPL with code released under an incompatible licence, they may not distribute the derivative work. The GPL will not apply if any incompatibly-licensed code is contained in identifiable sections of the whole work, and these 'can be reasonably considered [as] independent and separate works in themselves'.[132]

Second, when governments release software under a free or open source licence, they should do so under a licence that is compatible with the GPL.[133] Failure to do so would result in the software being excluded from incorporation into other free software projects that use a GPL licence, and prevent the project from reutilising externally-developed code that has been released under the GPL. Because the GPL is the single most popular free software licence, releasing software which can not make use of, or be incorporated into, GPL-licensed code greatly decreases the utility of free software. Importantly, however, it is not necessary to release code under the GPL itself, so long as the licence chosen is GPL-compatible.[134]

[131] James, above n 100, 68, citing David A Wheeler, *More than a Gigabuck: Estimating GNU/Linux's Size* (29 July 2002) David A Wheeler's Personal Home Page <http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>.

[132] FSF, *The GNU General Public License: Version 2.0*, above n 5, cl 2.

[133] The FSF maintains a list of licences that it considers to be compatible with the GPL: see Thorsten Sauter, *Various Licences and Comments about Them* (28 June 2005) The GNU Project <http://www.gnu.org/licenses/license-list.html>.

[134] Any licence which is no more restrictive than the GPL itself will be GPL-compatible. The easiest solution is to choose the pre-existing popular compatible licence that best fits the government's needs, such as the GPL (strong copyleft), LGPL (weak copyleft), or original MIT/X11 or revised BSD licences (non-copyleft): see David A Wheeler, *Make Your Open Source Software GPL-Compatible. Or Else* (16 February 2005) David A Wheeler's Personal Homepage <http://www.dwheeler.com/essays/gpl-compatible.html>.

D *Dual Licensing*

In order to avoid the problems associated with incompatible licences, it is possible, and increasingly common, to release software under two or more licences. The first licence is generally a strong copyleft licence, like the GPL, which prevents downstream developers from restricting any further freedom in the source. The second licence is often a more traditional, closed source software licence. This method effectively means that people who want to use and modify the software for further free software applications are able to do so, but those who wish to use the software in closed source proprietary applications must purchase a licence from the copyright owner. This is an effective way to commercialise software without sacrificing the benefits of releasing free software. Of course, this method will only work if there is a commercial market for the software when it is embedded or combined in other proprietary products; where the software is going to be distributed separately, or where it is clearly separable from proprietary software, further developers will not need to purchase the closed licence.[135]

Dual licensing must occur at the top level of code distribution. Copyleft licences prevent downstream developers from forking and relicensing code; only the owner of all the copyright in software can validly create dual licences. Where code is developed in an open manner, by many otherwise unrelated contributors, ownership of the copyright in the software is generally not vested in any one organisation — each contributor owns the copyright in the code they submit. To overcome this obstacle, organisations commonly require assignment of the copyright in each submission from the contributor to the organisation.[136] That organisation then provides guidance to the developers and decides the manner in which the software evolves. Developers wishing to take the software in another direction are able to take the code at any time and produce 'unofficial' versions or 'forks' of the product, but are not able to change the licence and close the source. In practice, both developers and users continue using the official version in all but the most extreme cases. Organisations that have no need to use dual licences will generally not need to claim copyright in contributed code.

VI THREATS TO THE FOSS MOVEMENT

Although the popularity of FOSS continues to grow, recent developments suggest that governments need to be mindful that the model (like any other) is not without opposition. This section considers the challenges posed by software patents, which remove code from the common pool, and the *SCO v IBM* litigation, which recently threatened to destabilise the free software model.

[135] See Mikko Välimäki, 'Dual Licensing in Open Source Software Industry' (2003) 8 *Systemes d'Information et Management* 63 <http://www.soberit.hut.fi/~msvalima/dual_licensing.pdf>.

[136] For an assignment of copyright to be effective it must be in writing and signed by or on behalf of the assignor: *Copyright Act 1968* (Cth) s 196. This is also the position in the United States: 17 USC § 204 (1978).

A *Software Patents*

FOSS code, like other types of software code, may be subject to patent protection.[137] Patents grant a limited term monopoly to make, sell, hire or use an invention.[138] An organisation that receives and uses software that is encumbered by a patent will generally be liable for infringement of that patent, if it is valid. The prospect of being sued for patent infringement provides another reason for governments to consider outsourcing their software development; contracting with a large vendor may allow governments to allocate the risk of subsequent intellectual property claims. The larger concern about software patents arises from their likely effect upon the development of free software.

Clause 7 of the GPL states that:

> If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all.[139]

Clause 7 prevents a person from redistributing code licensed under the GPL if it is encumbered by patents. However, it is not until 'conditions are imposed on you' that redistribution is prevented. In practice, this will mean that until patent infringement is asserted, distribution under the GPL — or any other software licence — is permitted. If many software patents are indeed invalid,[140] then distribution may not be significantly impaired by patent claims, as it is less likely that such claims will be asserted. Only where a court or patentee imposes actual restrictions upon proposed distributors will redistribution of infringing software be prevented; patents that may cover the code but are not enforced, or patents whose applicability or validity are suspect, will not prevent the distribution of GPL-licensed code. However, a distributor of software does take a risk when distributing or using code in relation to which a patent may be enforced in the future.

Clause 7 is only applicable to GPL-licensed code. In other cases, it will not prevent a developer from releasing patented software under a free licence where that software does not build on GPL-licensed code. Because the majority of free software licences are silent about patents, it is conceivable that an enterprising developer could release software under a free licence with the intention of

[137] See *CCOM Pty Ltd v Jiejing Pty Ltd* (1994) 51 FCR 260; *International Business Machines Corp v Commissioner of Patents* (1991) 33 FCR 218.

[138] *Patents Act 1990* (Cth) s 13(1).

[139] FSF, *The GNU General Public License: Version 2.0*, above n 5.

[140] See generally Mark A Lemley, 'Rational Ignorance at the Patent Office' (2001) 95 *Northwestern University Law Review* 1495. Recent empirical studies indicate that almost half of all fully-litigated patents are held invalid: see John R Allison and Mark A Lemley, 'Empirical Evidence on the Validity of Litigated Patents' (1998) 26 *American Intellectual Property Law Association Quarterly Journal* 185, 205–6.

enforcing patent rights at a later date. However, because the licences are silent as to patents, a patent licence may be implied when a developer releases software freely.[141]

In *Hewlett-Packard Co v Repeat-O-Type Stencil Manufacturing Corp Inc*,[142] the United States Court of Appeals for the Federal Circuit held that where a vendor sells a product without restriction, there is an implied licence granted to exploit any patent right held by the vendor to do the things for which the parties reasonably expect the product to be used.[143] There is no difficulty in extending this principle to the supply of software. Accordingly, where a developer releases software under a free software licence, it would appear similarly to allow the use of related patents by anyone who receives the free software.[144]

Daniel Ravicher, senior counsel for the FSF, argues that the scope of the implied licence is even greater under the GPL (and any other free licence that is silent as to patent claims) because the reasonably contemplated uses of the code explicitly include making derivative works, while other free software licences explicitly grant patent licences only for the code as distributed by the licensor.[145] For example, the Apache licence (version 2.0) grants an explicit patent licence over patent claims held by a contributor that are 'necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted.'[146] There is no grant for similar infringing code in the program, nor for modified or derivative works.

Finally, there is significant concern among the free software community that software vendors will begin to enforce their large patent portfolios to eliminate the competition posed by free software. Most of the large software developers have been aggressively acquiring patents for much of the software they develop, primarily in order to be able to use those patents defensively in the event of another developer enforcing their own patents.[147] Given that all of these large software developers cross-license their patent portfolios to each other, they are safe to continue to develop unhindered by the majority of software patents. Small free software developers, on the other hand, have little money to either license patents from the large developers or to apply for patents themselves. The

[141] In any case, the doctrine of equitable estoppel may operate to prevent a subsequent claim of infringement: see above Part V(B).

[142] 123 F 3d 1445 (Fed Cir, 1997).

[143] See also *Solar Thomson Engineering Co Ltd v Barton* [1977] RPC 537; *Bottom Line Management Inc v Pan Man Inc*, 228 F 3d 1352 (Fed Cir, 2000); *Surfco Hawaii v Fin Control Systems Pty Ltd*, 264 F 3d 1062 (Fed Cir, 2001).

[144] Alternatively, as a fall-back position, one could argue that the doctrine of equitable estoppel operates to prevent a distributor of free software from enforcing patent infringement claims in respect of that software. Note, however, that a term implied in the licence may be simpler to enforce and rely upon than an equitable remedy.

[145] Daniel Ravicher, 'Patents and Open Source Software: Forever Foes or Actually Allies' (Paper presented at the Phillips Fox–Red Hat Free and Open Source Software seminars, Sydney, Melbourne, Canberra and Brisbane, November 2004) 6.

[146] Apache Software Foundation, *Apache License: Version 2.0* (January 2004) cl 3 <http://www.apache.org/licenses/LICENSE-2.0>.

[147] See Kenneth Nichols, 'The Age of Software Patents' (1999) 32 *Computer* 25; Matt Hicks, *IBM Leads in Patent Race* (13 January 2004) eWeek.com <http://www.eweek.com/article2/0,4149,1435083,00.asp>.

biggest fear is that small or individual free software developers have been left behind in a patent arms race and are open to be sued for infringement of an ever-increasing stockpile of patents at any time.

Free software developers may, however, have several advantages when defending patent infringement claims that other software developers do not. First, as Ravicher notes, it would be very difficult to obtain a preliminary injunction to stop the use and distribution of software that allegedly infringes patent rights. This is because of the large number of users relying on the software and the practical impossibility of enforcing such a broad injunction against a distributed network.[148] Accordingly, Ravicher argues, a litigant would be forced either to seek damages, where it could choose to sue a small user or group of users who do not have large assets, or sue a large user or developer with the assets to defend themselves.[149] In many cases where a small developer is sued, they are likely to receive support from larger and better-resourced users of the software in question, or from the community in general, allowing the smaller developer to defend themselves.

The second great advantage that free software developers have is the willingness of the community to overcome patent claims, either by designing around the claims or finding prior art that invalidates them. One of the prerequisites of a valid patent is that it is novel and involves an inventive step, as compared to the prior art base as it existed before the date of filing.[150] The size of the free software community makes it easier for someone to provide an example of how any given patent is similar in function to something that has been published or used before, or how the invention claimed was obvious to anyone skilled in the art and hence did not involve an inventive step. Some of this work is starting to be done proactively, with free software developers maintaining a database of prior art,[151] or providing services to help challenge patents both before and after infringement claims have been lodged.[152] Where a patent's validity is contested, it is estimated that it will be found to be invalid in about half of all cases.[153]

If some free software is found to infringe a patent, the software must be modified in order to allow it to continue to be freely distributed. A valid, well-constructed patent is, by necessity, quite specific in its claims, and skilled developers are usually able to design around the patent if necessary. Given the lengthy process of litigation, it is likely that by the time any significant patent infringe-

[148] Ravicher, above n 145.

[149] Ibid.

[150] *Patents Act 1990* (Cth) ss 18(1), (1A).

[151] See, eg, FSF, *Prior Art Database* (2002) Savannah <http://www.nongnu.org/padb>.

[152] See Jason Schultz, *The Patent Busting Project* (2005) Electronic Frontier Foundation <http://www.eff.org/patent>; Public Patent Foundation, *The Public Patent Foundation* (2005) PUBPAT <http://www.pubpat.org>.

[153] Allison and Lemley, above n 140, 205–6. There is some suggestion that the percentage of invalid software patents is higher due to the difficulty of identifying prior art and reverse engineering software code for examination. See generally Julie E Cohen, 'Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Programs' (1995) 68 *Southern California Law Review* 1091, 1177. See also Lawrence Lessig, *The Problem with Patents* (23 April 1999) The Industry Standard <http://www.lessig.org/content/standard/0,1902,4296,00.html>.

ment claim is completed, some members of the free software community will have designed a non-infringing solution. An example of this willingness to code around can be seen in the community's initial reaction to the *SCO v IBM* litigation, where prominent members of the developer community promised to rewrite any infringing code they were shown.[154]

Software patents pose a significant threat to the software industry as a whole, particularly because the proliferation of incorrectly issued or invalid patents unreasonably raises transaction costs for all developers. When acquiring free software, governments should include future patent claims in their risk analysis, and should seek indemnities from large vendors if the risk justifies the cost. In particular, governments should consider software patents when releasing free software, but should also remember that the risk of infringement is generally no greater (and perhaps less) than that associated with in-house development of proprietary software. Indeed, the patent considerations are comparable to those affecting government research in other fields.

B *The* SCO v IBM *Litigation*

The recent and ongoing litigation between The SCO Group (Caldera Systems) and IBM has received great attention both inside and outside the free software community. Significant doubts have been raised as to the legitimacy of code within GNU/Linux, as well as concerns over the potential liability of users and developers of free software to third parties for infringement of intellectual property and other rights. To properly understand the lawsuits, it is necessary to briefly examine the history of UNIX and its relationship to Linux.

1 *UNIX and GNU/Linux — A Brief History*[155]

Development on UNIX officially commenced in 1969 by AT&T Bell Labs ('AT&T'), in conjunction with MIT; the University of California, Berkeley; and private and public developers. Individuals at universities and large organisations all around the world were helping to write UNIX tools and UNIX-like operating systems. AT&T was unable to commercialise UNIX due to a 1956 consent decree with the United States government over antitrust issues, which restricted its business to providing common carrier communication services.[156] Instead, the product thrived by distributing development across the world. AT&T was able to license its rights in UNIX to universities and large organisations, but only for nominal fees.

[154] See Roberto J Dohnert, *The SCO Threat: A Professional Linux User's Perspective* (10 June 2003) OSNews.com <http://www.osnews.com/story.php?news_id=3752>. Admittedly, it may be easier to rewrite code to prevent infringement of copyright than of patent rights.

[155] For a more detailed history, see Éric Lévénez, *UNIX History* (20 July 2005) <http://www.levenez.com/unix>; Eric S Raymond, *The Art of UNIX Programming* (2003) ch 2 <http://www.faqs.org/docs/artu>; Kelly, *Handout for the UNIX Industry: A Brief History* (2000) College Resource and Instructor Support Program <http://snap.nlc.dcccd.edu/learn/drkelly/hst-hand.htm>.

[156] See United States Department of Justice, 'Justice Department Agrees to Terminate Last Provisions of IBM Consent Decree in Stages Ending 5 Years from Today' (Press Release, 2 July 1996) <http://www.usdoj.gov/atr/public/press_releases/1996/0715.htm>.

In 1983, the United States Department of Justice won a second antitrust case against AT&T and broke up the conglomerate, leaving AT&T free to commercialise its interests, which it promptly did. Most universities and commercial distributions licensed UNIX from AT&T. Distributed development slowed because of licence issues and development mainly continued in the large licensed distributions, which caused a fragmentation among UNIX and UNIX-like operating systems.

Stallman launched the GNU project in 1983, with the goal of creating a free, UNIX-compatible operating system. The FSF, a tax-exempt charity, was created in 1985 to financially support free software. Despite developing many free tools, it did not complete a free operating system kernel.[157] Around the same time, Intel's low cost computer chips came into the market and were adopted by Microsoft. Microsoft released Windows 3.0 in 1990, and grew to dominate the desktop computer market. The large UNIX vendors kept working on the more elegant, more expensive microcomputers, but both the hardware and software proved too expensive to compete with Microsoft and Intel. UNIX development slowed once again.

In 1991, Linus Torvalds, a university student from Finland, began work on the Linux kernel, citing the high price of commercial UNIX distributions as a motivating factor. Linux provided the kernel that Stallman's GNU project had been missing, and distributed development on GNU/Linux, a free operating system that ran on cheap Intel hardware, began in earnest.

At the same time, the corporate UNIX market began to feel the pressure levelled by Microsoft and Intel, and many interests were disposed of and consolidated. Importantly, AT&T sold all its rights in UNIX to Novell. In 1995, Novell transferred some of its rights in UNIX, including the administration of the commercial UNIX licences, to the Santa Cruz Operation ('SCO', sometimes known as 'old-SCO', to contrast with the later 'The SCO Group', and later renamed Tarantella).

By the late 1990s, GNU/Linux had emerged as a viable competitor to the commercial UNIX distributions. IBM, Intel and SCO announced a joint project in 1998 to finally merge the proprietary UNIX distributions and revive the commercial UNIX industry, but the project failed in 2001. By this stage, most development was being carried out on GNU/Linux, and the commercial vendors (and particularly IBM) joined in, recognising the benefits of a business model built around selling hardware and support solutions incorporating GNU/Linux systems. IBM now carries on a large amount of development for GNU/Linux and other free software projects.[158]

2 *The Litigation*

Caldera Systems Inc was a company that manufactured GNU/Linux distributions. In 2001, it bought the rights to UNIX from the Santa Cruz Operation, and

[157] A kernel is the heart of an operating system: it provides software programs with access to a computer's hardware resources.

[158] For a list of technical contributions IBM has made to Linux, see IBM, *Open Source Projects* (2005) DeveloperWorks <http://www.ibm.com/developerworks/views/opensource/projects.jsp>.

later changed its name to 'The SCO Group' ('SCO'). In March 2003, SCO commenced an action against IBM in the United States District Court for the District of Utah. SCO alleged that it was the successor in title of all rights and interests in UNIX, which it derived from AT&T through a series of corporate acquisitions, and hence controlled the rights of all UNIX vendors (including IBM) to use and distribute UNIX. SCO's causes of actions stemmed from its allegations that IBM wrongfully used code and expertise developed by SCO (and its predecessors) in developing some aspects of the Linux kernel.

Novell claimed that SCO was not the successor in title of all rights and interests in UNIX, but instead acted as an agent or franchisee for Novell. Novell accordingly registered copyrights in UNIX, and SCO filed suit in the Utah State Court for slander of title. The suit was removed to the Federal Court and dismissed on the ground that SCO had not adequately specified special damages. The case was dismissed without prejudice; SCO can refile at a later date.[159]

In its amended complaint against IBM,[160] SCO is seeking US$3 billion in damages, alleging that IBM breached the terms and conditions contained in several Software Agreements relating to Unix System V source code by copying or adapting code into the Linux kernel. SCO further alleges that IBM engaged in unfair competition by aiding the development of Linux, and argues that in doing so IBM misappropriated SCO's trade secrets, particularly the knowledge and design developed by SCO for running a UNIX-based system on Intel processors.

IBM has counterclaimed, alleging that SCO breached the terms in the Software Agreements by purporting to terminate IBM's perpetual and irrevocable UNIX rights and that SCO has publicly misrepresented the legitimacy of IBM's Linux-related products and services, in violation of the *Lanham Act* 15 USC § 1051 (1946), and that SCO infringed four of IBM's software patents. IBM also alleges that by distributing Linux products, SCO agreed under the GPL not to assert certain proprietary rights over the Linux source code, and that SCO has breached its obligations under the GPL. The case is currently in the discovery phase.[161]

In response to SCO's claims that it will charge licence fees for commercial users of GNU/Linux systems,[162] Red Hat, a GNU/Linux distributor, has sued SCO for false advertising and deceptive trade practices, and has asked for a declaratory judgment of non-infringement of SCO's copyright. This case has been stayed pending the resolution of the case against IBM.[163]

SCO has also filed suit against two users of UNIX, DaimlerChrysler and AutoZone. DaimlerChrysler was granted summary judgment against almost all

[159] *The SCO Group Inc v Novell Inc*, No 2:04CV139DAK (D Utah, 9 June 2004). See also *The SCO Group Inc v Novell Inc*, No 2:04CV139 DAK (D Utah, 27 June 2005).

[160] Brent Hatch, David Boies and Stephen Zack, Plaintiff's Amended Complaint, *The SCO Group Inc v International Business Machines Corp*, No 03-CV-0294 (D Utah, 22 July 2003) <http://pl.caldera.com/scoip/lawsuits/ibm/ibm-25.pdf>.

[161] See *SCO v IBM*, No 2:03CV294 DAK (D Utah, 1 July 2005); *SCO v IBM*, No 2:03CV294 DAK (D Utah, 19 April 2005). See generally The SCO Group, *SCO v IBM Legal Filings* (2005) The SCO Group Inc Intellectual Property <http://www.caldera.com/scoip/lawsuits/ibm>.

[162] The SCO Group, *SCO Registers UNIX® Copyrights and Offers UNIX Licence* (21 July 2003) Investor Relations <http://ir.sco.com/ReleaseDetail.cfm?ReleaseID=114170>.

[163] *Red Hat Inc v The SCO Group Inc*, Civ No 03-772-SLR (D Del, 6 April 2004).

of SCO's claims (the claim that DaimlerChrysler took too long to respond to discovery is still on foot).[164] The case against AutoZone has been stayed pending resolution of the IBM, Red Hat and Novell cases.[165]

The suits filed by SCO have outraged the free software community. They do not, however, seem to pose as great a threat to Linux as was first imagined. SCO's claims are mostly rooted in breaches of the contract its (alleged) predecessors in title entered into with commercial vendors and users of UNIX, and breaches of fiduciary duties between those same parties. The possibility that SCO could have some proprietary claim to Linux is countered by the free software community's willingness to quickly rewrite any offending code.

VII Conclusion: The Choice to Be Made

There are significant advantages to a broad government adoption of free software. These include potential cost savings; adoption of open standards and protocols; wider use of stronger, more flexible and more secure software; and the social benefit derived from promoting a contributory commons of free software. However, governments ought to be aware of the obligations that may be imposed by the use and redistribution of FOSS, and when exactly these obligations will arise. Governments must also be mindful of the effect that implied warranties may have upon the sale or supply of free software by virtue of the *TPA*, or similar consumer legislation, and the limitations inherent in indemnity clauses in many free software licences.

Where a government is using public funds to develop a software application, great care must be taken when choosing a licensing strategy. If there is a large commercial market for the unmodified application, a traditional closed source licensing approach can be used to generate income. If the only commercial market for the software consists of software developers who would heavily modify or integrate the software, then a dual licensing approach could be taken to provide an income stream from those developers while still allowing the benefits of publicly-funded software to flow back to the community. Finally, where there is no commercial market for the software, and any sensitive or confidential information has been removed, there is a strong argument that the government should release the software under a free licence.

The evaluation of whether a government should use free or open source software for any given application is a complex matter. However, with the continual increase in quality and quantity of available solutions, coupled with increased understanding of the advantages and obligations involved, we can expect to see more widespread use of FOSS by governments across the world. In this context, the challenge for lawyers and government officials will be to fully understand the intricacies of this emerging area of law. This article is but one step in gaining an appreciation of the legal landscape involved.

[164] *The SCO Group Inc v DaimlerChrysler Corp*, Civ No 04-056587-CKB (D Mich, 9 August 2004).

[165] *The SCO Group Inc v Autozone Inc*, Civ No CV-S-04-0237-RCJ-LRL (D Nev, 29 October 2004).