Avoiding the Software Patent Problem: An Alternative Fix For TRIPS Junkies

Anton Hughes*

Summary

For members of the World Trade Organisation, the decision whether to allow patenting of software innovations may seem to be foreclosed by Article 27 of the Agreement on Trade-Related Aspects of Intellectual Property (TRIPS). However, a closer reading of Article 27 shows that this is not the case. This article considers how the structure of the software industry, and the nature of software development create difficulties when combined with the patent paradigm, suggesting that software patents are a problem best avoided. Although there are various attempts underway to mitigate the ills of software patenting, none offers a complete solution. By taking the path of implementing an alternative protection regime similar to that available for circuit layouts, it is possible to both meet TRIPS requirements and truly promote software innovation, without the extra baggage of a patent regime which struggles to adapt to the needs of the information age.

Are software patents a necessity for WTO members?

Most countries in the Asia-Pacific region, are either members of the World Trade Organization (WTO), or have been granted observer status, meaning that they must start accession negotiations within five years.[1] Australia has been a member since 1995. One constituting document of the WTO, the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) requires that WTO members implement certain minimum standards of IP protection or face potential trade sanctions under the WTO's formal dispute resolution procedures. According to Article 27.1 of TRIPS, "patents shall be

* Anton is a software developer who started programming 20 years ago when he got sick of all the games on his father's Apple][c. When not developing web applications using open source technologies, he is studying towards a PhD in Law at the University of Tasmania. Anton's thesis topic is on tailoring intellectual property regimes to meet the needs of the Australian software industry.

[1] A current list of WTO members and observers is available on the WTO website. See "Understanding the WTO – members," *World Trade Organisation,* <http://www.wto.org/english/thewto_e/whatis_e/tif_e/org6_e.htm> (31 August 2006).

available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application." Article 27 further mandates that "patents shall be available and patent rights enjoyable *without discrimination* as to … the field of technology" (emphasis added). Subject to limited exceptions,[2] this appears to require a technology neutral approach to patents.[3] This would seemingly foreclose the question of whether the burden of software patenting is one which all WTO members must bear.

Fortunately, this is not the only possible interpretation. Firstly, TRIPS requires only that patents be available for "inventions", but does not define the term. This suggests some leeway in excluding certain subjects from TRIPS as non-inventions.[4] Secondly, some have argued that software development is not a "field of technology".[5] The basis for such a distinction is stated in the 2003 version of the European software patent directive, which stated that "[t]he processing, handling, and presentation of information do not belong to a technical field, even where technical devices are employed for such purposes."[6] Of course, such an interpretation must overcome the popular perception of programming as an activity in the field of 'information technology'.

Article 27 may also contain another escape route from software patenting, grounded in a careful interpretation of the phrase "without discrimination". Discrimination can be defined as follows:

> **discrimination** n., (1) unfair treatment of a person, racial group, minority; action based on prejudice; (2) subtle appreciation of matters of taste; (3) the ability to see fine distinctions and differences.[7]

Nobody would argue that TRIPS seeks to address matters of taste, therefore the correct interpretation must be either the first or the third definition of discrimination. It is submitted that the third definition, the ability to see fine distinctions and differences, is clearly not the correct interpretation. TRIPS itself makes distinctions between fields of technology by allowing WTO members a general discretion to exempt particular inventions in order "to

---

[2] Where the prevention of commercial exploitation is necessary to protect "ordre public" or morality, or to avoid serious prejudice to the environment. See Articles 27(2) and (3); Christie A and Syme S, 'Patents for Algorithms in Australia' (1998) 20 *Sydney Law Review* 23 at 527.at 544-545.

[3] This interpretation was taken by many submissions to the recent Australian Law Reform Commission inquiry into the patenting of genetic materials. See Australian Law Reform Commission, 'Genes and Ingenuity: Gene Patenting and Human Health' (2004) ALRC 99. <http://www.austlii.edu.au/au/other/alrc/publications/reports/99/>

[4] See "The TRIPS Treaty and Software Patents," *Foundation for a Free Information Infrastructure*, 25 June 2004 <http://swpat.ffii.org/analysis/trips/index.en.html> (11 September 2006).

[5] Ibid. See also Lenz K, "TRIPS and European software patent legislation," *Foundation for a Free Information Infrastructure*, <http://swpat.ffii.org/papers/europarl0309/amends05/lenz05/> (11 September 2006).

[6] Article 2b, Directive on the Patentability of Computer-Implemented Inventions, 2003. The full text of the directive is available at < http://swpat.ffii.org/papers/europarl0309/index.en.html> (11 September 2006).

[7] Collins Australian Dictionary and Thesaurus, 3rd Edition, HarperCollins, 2004.

protect order public or morality",[8] such as where these inventions would harm ""human, animal or plant life or health or to avoid serious prejudice to the environment".[9] Further, TRIPS grants WTO members a power to exempt specific classes of inventions, namely, methods of medical treatment, and plants and animals.[10] Support for such an interpretation is also to be found in a 1997 WTO dispute resolution panel decision,[11] where it was acknowledged that the prohibition against discrimination does not preclude differentiation.[12] Further support for this position can also be found in a 1988 WIPO study, undertaken at the time the 'principles' of TRIPS were being hammered out by big business, revealing a wide range of variation in national exceptions to patentability.[13]

Thus the first definition is all that remains. Taking "without discrimination" to be equivalent in meaning to "without prejudice or unfair treatment" leads inevitably to a conclusion that technology-specific alterations to national patent regimes are possible, so long as the protection thereby afforded is equivalent to the minimum standard of patent protection set out in TRIPS. Some guidelines as to what might amount to equivalent protection can be found in the text of Article 7:

> "The protection and enforcement of intellectual property rights should contribute to the promotion of technological innovation and to the transfer and dissemination of technology, to the mutual advantage of producers and users of technological knowledge and in a manner conducive to social and economic welfare, and to a balance of rights and obligations."

An alternative protection paradigm, which fulfils these requirements, is considered later in this article.

Why avoid software patenting?

Software is intangible

Patent law is supposed to protect the expression of an idea, in the form of an invention, not the idea itself. The reason for this is that "[t]he essence of the patent system is to encourage entrepreneurs to develop and commercialise

[8] Article 27.2.
[9] Ibid.
[10] Article 27.3.
[11] *Canada – Patent Protection for Pharmaceutical Products [EC – Canada]*, WT/DS 114/R.
[12] See UNCTAD-ICTSD, "Chapter 18 – Patents: Non-Discrimination," *Resource Book on TRIPS and Development*, Cambridge University Press, May 2005, <http://www.iprsonline.org/unctadictsd/docs/RB2.5_Patents_2.5.2_update.pdf> (31 July 2006), at 370-371.
[13] Relevantly, 32 of the 98 Paris Convention members excluded computer programs. Other exclusions included pharmaceutical products, animal varieties, methods of treatment, plant varieties, food products, and chemical products. See Drahos P with Braithwaite J, *Information Feudalism: Who Owns the Knowledge Economy?* Earthscan Publications, London, 2002, at 124.

new technology."[14] Simply allowing patents on ideas does little to ensure that these ideas are brought to market. When the subject matter of the invention has a physical manifestation, such as a new machine, it is very easy to separate the expression from the idea. With intangible subject matter such as software, this boundary becomes much less clear. The software development process generally involves a gradual move from the idea behind a program towards the more specific code actually implementing the idea. Knowing at which point the unpatentable idea becomes a patentable invention is a difficult one. If the patent is too specific, it will provide little protection against imitators, if it is too broad, the scope of the patent will be too broad, and will give the inventor control over a wide range of independently developed technologies – a windfall for the patent holder at the expense of other innovators. Unfortunately, the recent trend in patent law has been in the latter direction.

Network effects amplify the power of patent holders

Network effects are common in software development, where having a trained base of users for a particular product creates barriers to entry for competitor products. A network effect refers to "the social advantages that arise when all of the users of a particular type of technology adhere to the same standards and thus can share their work and move easily between machines and businesses".[15] A simple example is the Microsoft Word file format, which is a de facto standard for storing and exchanging word processor documents.

Network effects may also arise as a result of the adoption by the industry of a particular standard. For example, the TCP/IP protocol is the standard for the transmission of digital information over computer networks, particularly the Internet. Standards are generally used to encourage interoperability and collaboration in a technologically optimal way.

Network effects amplify the ability of their owners to control a market, by removing the bargaining power of competitors. For example, incorporating a patented technology into a standard allows the patent holder to enforce a licence fee from not only all competitors needing their products to interoperate with the standard, but also users of competing products.[16] This is because the patent cannot be 'invented around' without deviating from the standard.

Software is highly complex

Compared to the industrial-era machines which the patent system was originally designed to protect, a software product is a highly complex

[14] Commonwealth of Australia, 'Patents Bill 1990: Second Reading' Senate, 29 May, 1990 <http://parlinfoweb.aph.gov.au/PIWeb/view_document.aspx?id=562046&table=HANSARDS> (2 November 2004).

[15] Fisher WW, 'Intellectual Property and Innovation: Theoretical, Empirical, and Historical Perspectives' (2001) 37 Industrial Property, Innovation, and the Knowledge-based Economy, Beleidsstudies Technologie Economie, at 25.

[16] Technically, use of the product would fall within the scope of the patent holder's right to exploit the invention. It is assumed however that a patent holder is unlikely to sue its own customers.

construction. Whilst a physical machine may be made of thousands of individual components, a large computer program (such as an operating system) may contain millions of components.[17] The fact that each individual component may potentially be patented, makes infringement possible on a massive scale. Unlike a pharmaceutical product where one drug is likely to only infringe one patent, a software product could potentially infringe millions of patents.

Another aspect of this high component-to-product ratio is that the patentability of such vast numbers of patentable components can lead to an administrative nightmare for national patent offices. The USPTO put up stiff resistance to the introduction of software patents, on the basis that allowing software patents would leave the patent office unable to handle the volume of patent applications which would result. And that is exactly what has happened, with the current backlog of patents at the USPTO having grown to nearly 600,000 applications at the end of 2005.[18] The obvious result of such a backlog is the increased likelihood that individual patent examinations will not be carried out thoroughly enough, resulting in an across-the-board drop in patent quality. When doubtful patents are passed, the costs of pursuing litigation to see them overturned (usually at the risk of infringement proceedings) means that often such patents are successfully enforced against competitors.

Software innovations are poorly documented

Programmers are notoriously lousy at documenting the programs they write. This is perhaps because the source code *is* the documentation, or perhaps because compared to the active problem solving involved writing code, writing documentation is just plain boring.[19] Even when documentation is properly written, the ubiquity of computers, the global nature of the industry, and the absence of any centralised knowledge repository (such as trade journals) means that it is very difficult to adequately represent the entire state of the art in documentary form at any point in time.[20] The absence of documentation presents serious problems to the patent examination process, as proper

[17] Stallman R, "The Danger of Software Patents," Transcript of Speech Given at Cambridge Unversity, March 2002, <http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html> (1 September 2006) at [90].
[18] Brandt A, "Patent Overload Hampers Tech Innovation," PC World Magazine, April 2006 issue, <http://www.pcworld.com/news/article/0,aid,124826,00.asp> (5 June 2006).
[19] For a representative range of views on the issue, see "Why Do Programmers Hate Documenting?" <http://discuss.fogcreek.com/joelonsoftware1/default.asp?cmd=show&ixPost=35336&ixReplies=61>
[20] There are at least 455,000 programmers employed in the US alone, across almost every industry. See Bureau of Labor Statistics, US Department of Labor, *Occupational Outlook Handbook*, 2006-7 Edition, Computer Programmers <http://www.bls.gov/oco/ocos110.htm>. Based on 2002 or later data on the relative size of the software industries in Brazil, China, India, Ireland, Israel, Japan and Germany, it is estimated that the number of programmers in those countries would be in the region of 660,000. See Arora A & Gambardella A, "The Globalization of the Software Industry: Perspectives and Opportunities For Developed and Developing Countries" National Bureau of Economic Research Working Paper 10538, June 2004 <http://www.nber.org/papers/w10538> (23 August 2006). The actual number of people writing software is likely to be much higher, as software may also be written by computer software engineers; computer scientists and database administrators; and computer systems analysts who are all classified separately by the Bureau. This estimate also fails to capture volunteer programmers, students (such as this author) and those working in seemingly unrelated disciplines, for example.

documentation is critical to the application of patentability standards. Without a complete prior art base, it is impossible for patent examiners, who are rarely software developers, to adjudge whether an alleged invention is truly novel, and whether such an advance would have been obvious to a professional working in the field (the nonobvious requirement). The result of poor documentation is thus that trivial advances over the state of the art will regularly be elevated to the status of patentable invention.[21]

Software innovations are largely cumulative

The low barriers to entry in software development mean that the market is massively decentralised. This decentralization leads to independent repetition, "because programmers use similar, if not identical, software and hardware tools to tackle common needs".[22] Because of this repetitiveness, and perhaps due in part to the sharing ethos developed in the early era of computer science,[23] and the importance of familiarity to users, "[p]rogrammers commonly adopt software design elements - ideas about how to do particular things in software - by looking around for examples or remembering what worked in other programs".[24] As such, most new developments of the technology tend to be cumulative, grain-sized innovations rather than big leaps forward.

The cumulative nature of software development has been consistently under-emphasised by the courts and patent offices, meaning that large rewards (in the form of patent licences) are available as the result of minimal research. The lure of large profits thus encourages opportunists to obtain and enforce broad patents at the expense of innovation in the field.[25]

The software market moves too fast for the patent regime

A patent grant involves the award of monopoly rights to individual entities, typically for a 20-year period. This is an eternity in the fast-paced software industry, where a typical product lifecycle is around 3-5 years.[26] Because patents are construed as broadly as the language of their claims will allow, however, patent holders may be able to assume control over technological developments that were far beyond the state of the art at the time the patent was granted.

---

[21] This is the criticism typically levelled at the Amazon 1-click patent, which many developers consider to be an obvious application of cookie technology. For an enlightening discussion of the controversy, see O'Reilly T, 'Ask Tim: Open Source, Patents & O'Reilly' O'Reilly Network, 28 February, 2000 <http://www.oreilly.com/pub/a/oreilly/ask_tim/2000/amazon_patent.html> (22 November 2004)

[22] Klemens B, "New legal code: copyrights should replace software patents" (2005) Spectrum Magazine, IEEE <http://www.spectrum.ieee.org/aug05/1686> (5 June 2006).

[23] See below.

[24] See Samuelson P, Davis R, Kapor M and Reichman J, 'A Manifesto Concerning the Legal Protection of Computer Programs' (1994) 94 *Columbia Law Review* 2308 at 2330-2331.

[25] See Fisher, above n15 at 18.

[26] For an overview see Wikipedia, 'Moore's Law' <http://en.wikipedia.org/wiki/Moore%27s_law> (14 January 2005). For a more in-depth analysis, see Tuomi I, 'The Lives and Death of Moore's Law' (2002) 7(11) FirstMonday <http://www.firstmonday.dk/issues/issue7_11/tuomi/> (14th January 2005)

Even the application process is too slow. The current waiting period for examination of a software patent application at the USPTO is almost 4 years,[27] during which time the application remains secret. Given the independent repetition in the industry discussed above, it thus becomes likely that somewhere an entirely independent yet theoretically infringing software product could be designed, implemented, brought to market, and even replaced by version 2.0 before any knowledge of a relevant patent was available. Infringement however, runs from the date of application, meaning substantial royalties would be owed. In this scenario, the product's developer reaped none of the benefit of the patent and bears all of the burden of establishing a successful market for their product, yet the patent holder is able to hold the product developer's business hostage until royalties and/or a suitable licence are negotiated.

The software industry doesn't want them

Another significant factor which weighs against software patenting is the fact most programmers believe that software patents "significantly hamper their work".[28] In fact, the League for Programming Freedom has been arguing for their abolition since at least 1991.[29] Unsurprisingly, the voices of key free and open source software figures such as Richard Stallman,[30] Linus Torvalds,[31] and Bruce Perens[32] are unanimous in arguing that the only solution to the problems which software patents cause software developers is to abolish software patents altogether. There is also empirical[33] and anecdotal evidence[34] to suggest that proprietary software developers would happily do without software patents. Why is this?

Firstly, despite the prevalence in the public consciousness of large US firms such as IBM, Microsoft and Oracle, the global software industry is largely populated by small-to-medium enterprises (SMEs).[35] In Australia, for example, a 2005 study found that 99.6% of Australian computer software and services companies had less than 100 employees.[36] These small players are

---

[27] The average time from application to award for high tech patents was 43.5 months for the 2005 fiscal year. See USPTO, "Other Accompanying Information" *Performance and Accountability Report Fiscal Year 2005*, <http://www.uspto.gov/web/offices/com/annual/2005/060404_table4.html> (1 September 2006).

[28] Ibid, at 25.

[29] See League for Programming Freedom, 'Against Software Patents' 28 February 1991 <http://lpf.ai.mit.edu/Patents/against-software-patents.html>.

[30] Stallman R, 'The Danger of Software Patents' 2004 Cyberspace Law and Policy Seminar (audio recording), Sydney, 14 October 2004 <http://images.indymedia.org/imc/sydney/stallman%20lo.ogg> (11 November 2004).

[31] Torvalds L & Cox A, 'Open Letter on Software Patents from Linux Developers' 21 September, 2003 <http://www.effi.org/patentit/patents_torvalds_cox.html> (15 January 2005).

[32] Perens B, 'Software Patents vs. Free Software' <http://perens.com/Articles/Patents.html> (15 January 2005).

[33] See Oz E, ""Acceptable Protection of Software Intellectual Property: A Survey of Software Developers and Lawyers." (1998) 34(3) *Information & Management* 161.

[34] The selection of anti-software-patent quotes collated on this page read like a virtual who's-who of the US Software Industry. See "Quotations on Software Patents," Foundation for a Free Information Infrastructure, <http://swpat.ffii.org/vreji/quotes/index.en.html> (4 June 2006).

[35] See Samuelson et al, above n24 at 2377.

[36] Centre for Innovative Industry Economic Research Consortium, "Executive Summary", *The Australian software industry and vertical applications markets: Globally competitive and domestically*

particularly vulnerable to the problems software patenting creates. Their limited budgets means that they can afford neither substantial licensing fees, the costs involved in challenging invalid patents and defending against infringement, nor the maintenance of defensive patent portfolios.[37] Thus, it is they who feel the pain of software patenting most sorely.[38] It is little wonder then that SMEs largely threw their weight behind the anti-software patenting camp in the recent battle to prevent the legitimisation of software patenting in the EU.[39]

The software giants are generally able to limit the likelihood of infringement proceedings by maintaining large patent portfolios. These portfolios are reminiscent of the "mutually assured destruction"[40] policy of the superpowers in the Cold War, in that any would-be litigant is discouraged from initiating proceedings lest they should be hit with a counter-claim for infringement themselves. However, software patenting, nevertheless, impacts negatively on the larger players. Maintaining such a strategy consumes time and money. Robert Barr, vice president and head of the patent department of Cisco Inc, in a frank admission to the FTC, put it this way:

> "The time and money we spend on patent filings, prosecution, and maintenance, litigation and licensing could be better spent on product development and research leading to more innovation. But we are filing hundreds of patents each year for reasons unrelated to promoting or protecting innovation."[41]

Defensive portfolios also do little to discourage so-called patent trolls – companies who do not engage in software development, and whose sole source of income depends on the opportunistic enforcement of the patents they acquire.[42]

Software patents threaten openness

Since its origins in the computer science departments of universities, openness and collaboration has had an influential effect on the state of the art in software. At the earliest stages, computer scientists shared the source code of software innovations, either directly or through publication in freely available journals. This ethos of openness amongst software developers continued even as commercial players began to get involved. For example,

---

undervalued, <http://www.dcita.gov.au/ict/ict_industry_information/the_australian_software_industry__and__vertical_applications_markets_globally_competitive,_domestically_undervalued> (1 September 2006).

[37] See below.

[38] On this point see Tang P, Adams J, Paré D, "Patent protection of computer programmes" (2001)

[39] "Alliance of 2,000,000 SMEs against Software Patents and EU Directive," Foundation for a Free Information Infrastructure, 16 September 2003, <http://swpat.ffii.org/papers/eubsa-swpat0202/ceapme0309/> (4 June 2006).

[40] See "Mutually Assured Destruction," Wikipedia, <http://en.wikipedia.org/wiki/Mutually_assured_destruction> (5 June 2006).

[41] Barr R, "Statement to the Federal Trade Commission", 2002, Foundation for a Free Information Infrastructure, <http://swpat.ffii.org/papers/ftc02/cisco/index.en.html> (5 June 2005)

[42] For more on how patent trolls operate, see "Patent Troll," Wikipedia, <http://en.wikipedia.org/wiki/Patent_troll> (5 June 2006).

the early history of the Unix operating system for a long time involved an open collaboration with the University of California at Berkeley.[43]

Perhaps the most important form of openness in modern software development, however, is the Free and Open Source Software (FOSS) model, which provides a mechanism for global, collaborative innovation based on principles of freedom and sharing.[44] FOSS represents an important subset of the software industry, because it embraces an alternative path to innovation and global collaboration which furthers the public interest in the development and disclosure of new technologies – the same goal claimed by the patent system.

The continued importance of openness to innovation in software development is at odds with the award of software patents. That software patents increase innovation in the field is also challenged by impressionistic evidence of the US software industry, which suggests that innovation levels were already high before software patenting began, and have not increased since.[45]

FOSS developers face the same problems as all software developers, as outlined above, but there are a number of ways in which FOSS projects are more vulnerable than their proprietary counterparts. Firstly, FOSS development is in most cases undertaken by small service-oriented companies who make their money out of implementation and administration of the software on customer projects, or in many cases volunteers.[46] Thus, the problems of SMEs discussed above are particularly relevant to the ongoing viability of FOSS projects. The problem is multiplied, however, as FOSS projects generally lack a central body to fund and harvest a large patent portfolio, or to negotiate licensing agreements to cover project participants en masse.

Further, FOSS projects harness the collaborative power of the Internet which allows co-operation on projects on a global scale exceeding even that of the software industry monoliths.[47] The geographical distribution of project

---

[43] For a more detailed history of the BSD project, see McKusick MK, 'Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable' in Di Bona C & Ockman S (eds) *Open Sources: Voices from the Open Source Revolution*, O'Reilly & Associates, 1999.

[44] For more on the nature of FOSS, see Wong K & Sayo P, "Free/Open Source Software: A General Introduction," International Open Source Network, 2004 <http://www.iosn.net/foss/foss-general-primer/> (5 June 2006).

[45] Fisher, above n15 at 24-25.

[46] The author bases this proposition on his personal experience with the Zope project community (http://zope.org/). Empirical evidence gathered in relation to Embedded Linux also supports this view, with 139 of 259 survey participants working for organisations with less than 50 people, and 81 participants working for companies with 200 or more staff. See Henkel J and Tins M, 'Munich/MIT Survey: Development of Embedded Linux' Institute for Innovation Research, Technology Management and Entrepreneurship, University of Munich, 10 May 2004 <http://opensource.mit.edu/papers/henkeltins.pdf> (20 January 2005) at 7.

[47] Roberts R, 'Why Linux is Wealthier than Microsoft' BusinessWeek Online, 19 November, 2003 <http://yahoo.businessweek.com/technology/content/nov2003/tc20031119_9737.htm> (18 January 2005).

participants together with the ability of any party to 'fork' the code,[48] a central tenet of the open source paradigm, illustrates the distributed nature of FOSS development in which no central body has ultimate control. Sometimes a project will have one or more leaders,[49] or a non-profit organisation which holds the intellectual property and hosts development services,[50] but the distributed nature of these projects are more analogous to Grokster than Napster.[51] This may be the key to the success of these projects, but the downside is that legal responsibility for patent infringement is likely to be directed at individual contributors and even users of the software. Whilst some may suggest that the sheer difficulty of pursuing a distributed target makes enforcement less likely,[52] many open source advocates have long felt that patents represent the biggest threat to this form of software development.[53]

The threat of even high profile FOSS projects being shut down by the spectre of patent litigation is a real one. In a recent study undertaken by Open Source Risk Management, it was found that the Linux kernel potentially infringes 283 US software patents which have yet to be tested by the courts.[54] Stallman has estimated that the Linux kernel represents only 0.25% of a complete GNU/Linux system,[55] suggesting a possible total of between 30,000 and 300,000 possible infringements overall.[56] Although a "a third of the patents are owned by Linux backers, including Hewlett-Packard, IBM, Novell, and Oracle, which are unlikely to assert claims," such a precarious position is cause for concern as it only takes one patent to shut down a project.[57]

---

[48] Code forking is a consequence of the liberal modification and redistribution rights given in OSS licences. If any one party is unhappy with the direction a project is taking, they can take a copy of the source code and use it to launch their own project. See Wikipedia, 'Code Forking' <http://en.wikipedia.org/wiki/Code_forking> (18 January 2005).

[49] For example, Linus Torvalds is the project leader for the Linux kernel project.

[50] For example the Apache Software Foundation (see <http://apache.org>) or the Mozilla Foundation (see <http://mozilla.org/>) host the project website, version control and mailing lists which allow project collaboration in addition to holding IP rights in the project code.

[51] On the legal implications of distributed versus centralised control see *Metro-Goldwyn-Mayer Studios, Inc v Grokster*, Ltd US District Court Central District of California (25 April 2003).

[52] Williams P, 'Patent threat to open source is limited' vnunet.com 8 September, 2004 <http://www.pcw.co.uk/news/1157906> (5 January 2005). On the other hand, the Recording Industry Association of America in the US are certainly having a good shot at pursuing similarly distributed filesharers with copyright infringement suits. See Jaques R, 'RIAA Launches P2P file sharing legal blitz' vnunet.com 19 November 2004 <http://www.vnunet.com/news/1159534> (20 January 2005).

[53] See for example Babcock C and Greenemeier L, 'Open Source Stress' Information Week, 9 August, 2004 <http://www.informationweek.com/story/showArticle.jhtml?articleID=26806464> (3 January 2005).

[54] Ravicher D, 'Position Paper: Mitigating Linux Patent Risk' Open Source Risk Management 2 August, 2004 <http://www.osriskmanagement.com/pdf_articles/linuxpatentpaper.pdf> (3 January 2005) at 1.

[55] This is what people mistakenly refer to as the Linux operating system, which Stallman insists is more correctly called GNU/Linux. See Stallman, above n30.

[56] Ibid.

[57] For example, the preamble to the GNU General Public Licence has for a decade recognised that "any free program is threatened constantly by software patents". See Free Software Foundation, 'GNU General Public License' 16 February 1998 <http://www.gnu.org/copyleft/gpl.html> (3 January 2005)

Is avoidance the only viable solution?

Avoiding software patenting is not the only possible approach to the problems caused by software patents. But it is the only one that stands a chance of solving the problems just discussed. To understand why, it is useful to consider the limitations of alternative strategies which various commentators have proposed.

Option 1 – Stricter disclosure

By enforcing the disclosure requirement much more strictly, the scope of patent claims could be considerably narrowed. The most extreme version of this reform would require disclosure of source code or pseudocode[58] of an implementation of the invention. This would limit the effective life of the patent to a period consistent with the market life of software innovations. A more extensive disclosure of the invention may result in better quality documentation of the prior art in the field. Also, such an approach would increase the likelihood of patent literature becoming a useful source of technical information for programmers. The problem with narrowing patent scope, however, is that with computer programs it is "quite possible to produce functionally indistinguishable program behaviours through the use of more than one method."[59] Thus such a solution replaces an overprotection problem with one of underprotection.

Option 2 – Improve the examination process

Other solutions focus on the possibility of improving patent quality by improving the patent examination process. There have been recent moves in the US to improve funding[60] to the USPTO and address the high attrition rate[61] of patent examiners. Given the ever-increasing presence of software in daily life and its accompanying growth in complexity, one should wonder whether the patent office budget can keep up. More recently, the USPTO has been considering a system of peer review of patent applications as a way of providing greater scrutiny by creating "a peer review system for patents that exploits network technology to enable innovation experts to inform the patent examination procedure."[62] Whilst such a reform is a positive step towards filling the gap in the documented prior art base, it is not without problems. One commentator has voiced concerns about the ability of such a system to deal

---

[58] Pseudocode is an informal way of describing the algorithm behind a computer program independent of the syntactic constraints of a programming language. See for example "Pseudocode," *Wikipedia*, <http://en.wikipedia.org/wiki/Pseudocode> (23 August 2006).

[59] Samuelson et al, above n24 at 2345.

[60] Stirland SL, "Bush makes new push for patent office to keep fees" GovExec.com, 6 February 2006, <http://www.govexec.com/story_page.cfm?articleid=33317> (4 June 2006).

[61] Olsen F, "Patent examiners battle stress," Information Today, 25 July 2005, <http://www.fcw.com/article89658-07-25-05-Print> (4 June 2006).

[62] "About Community Patent," The Community Patent Project, <http://dotank.nyls.edu/communitypatent/about.html> (4 June 2006). For more detail, see Noveck B, "'Peer to Patent': Collective Intelligence and Intellectual Property Reform" NYLS Legal Studies Research Paper No. 05/06-18, 25 April 2006, <http://ssrn.com/abstract=898840> (23 August 2006).

with the huge volume of patent applications.[63] Based as it is on social software such as Wikipedia,[64] the system must also deal with similar concerns as to the potential for fraud and bias within such a system,[65] particularly since patent applicants have more at stake than a favourable review in an online encyclopaedia. It is also relevant to note that such a system is national in nature and thus depends on having a sufficient number of relevant experts within a country's borders, or else relying on volunteer experts from other jurisdictions who may not be aware of local developments. Such a national system is of course at odds with the global nature of the software development community. All this aside, however, perhaps the biggest limitation of Peer to Patent is that it will only apply to the examination of up-and-coming patents. This means that the current crop of bad patents will dominate the software development landscape for at least the next 20 years.

Option 3 – Compulsory licensing

Compulsory licensing is one means of addressing the problems which the power to exclude causes in an industry characterised by sequential innovation and a high component-to-product ratio. Theoretically, the availability of a compulsory licence limits the ability of a patent holder to hijack a competitor's business by refusing to licence a patented invention by allowing the competitor to obtain a licence from the Crown. However, the need to pay a 'reasonable' licence fee, and a requirement of a court order mean the transaction costs in using such a scheme are impracticably high. Further, compulsory licences do little to improve the quality or scope of awarded software patents.

Option 4 – Protect reverse engineering

Other solutions borrow from the copyright approach, in order to recognise a distinction between unfair imitation and legitimate reverse engineering. O'Rourke advocates creating a 'fair use' defence, along the lines of that available in US copyright law.[66] According to this reform, courts would have the power to balance innovation incentives against social benefits in the context of a particular case to determine whether an otherwise infringing activity should be allowed. Such a reform could perhaps bring the patent system more into line with cumulative innovation, and to acknowledge the importance of openness to software development. However, there is a fair amount of uncertainty inherent in the operation of this scheme in that the party

---

[63] Jason Schultz, attorney for the Electronic Frontier Foundation. Quoted in Terdiman D, "Web Could Unclog Patent Backlog," *Wired News*, 14 July 2005, <http://www.wired.com/news/technology/0,68186-0.html> (23 August 2006).

[64] Ibid.

[65] See Demsyn R, "Wal-marts Wikipedia War," *Whitedust Security*, 28 April 2006, <http://www.whitedustnet/article/55/Wal-marts_Wikipedia_War/> (23 April 2006); "Wikipedia Criticised By Its Co-Founder," *slashdot.org*, 3 January 2005, <http://slashdot.org/articles/05/01/03/144207.shtml>; Mehegan D, "Bias, sabotage haunt Wikipedia's free world," *The Boston Globe*, <http://www.boston.com/news/nation/articles/2006/02/12/bias_sabotage_haunt_wikipedias_free_world/>. Despite such criticisms, this author continues to cite Wikipedia, even in this article, when the author feels the referenced material can be confirmed by external sources, but explains the material well.

[66] See O'Rourke MA, 'Toward a Doctrine of Fair Use in Patent Law' (2000) 100 *Columbia Law Review* 1177.

claiming fair use must be prepared to brave a patent infringement suit in order to establish their entitlement to continue in their activity.

Another possibility is to explicitly limit the interpretation of patent claims so as to exclude independently implemented functional equivalents.[67] This may be another path to the same objective as the stricter disclosure reform outlined above. It thus shares the likelihood that it will result in the underprotection of software inventions.

Option 5 - Private initiatives

A number of private initiatives also seek to address the patent problem. IBM, in 2005, started the ball rolling by pledging not to enforce 500 of the patents in its portfolio against FOSS developers, and promised to enforce their patents against any organisation pursuing infringement action against an OSS project. Similar pledges have since been made by Sun, Nokia, Novell, Red Hat and Computer Associates.[68] Putting aside questions of whether these pledges are just publicity stunts,[69] such moves are to be applauded, in that they provide open source developers with at least some level of protection against infringement suits. The problem with such initiatives is that they cannot prevent so-called 'patent trolls' from enforcing their patents against FOSS developers. The typical defensive patent portfolio strategy, which such schemes extend to the FOSS community, is ineffective against trolls, since they do not generally engage in software development and are hence safe from cross-claims for infringement.

Another initiative seeks to raise standards in future software patents by addressing the prior art gap. The Open Source as Prior Art project[70] hopes to improve patent quality by "improving accessibility by patent examiners and others to electronically published source code and its related documentation as a source of prior art."[71] This is a commendable effort to address one of the root causes of poor patent quality in software, yet it is far from a complete solution, and it can do nothing to resolve the problems with dubious patents which have already been awarded.

A final strategy directs the global collaboration model of OSS development at defeating bad patents. The Public Patent Foundation[72] is a not-for-profit organisation which volunteers to search for prior art which will invalidate bad patents. The Foundation has had some recent successes in challenging

---

[67] Cohen JE and Lemley MA, 'Patent Scope and Innovation in the Software Industry' (2001) 89 *California Law Review* 1 <http://www.law.georgetown.edu/faculty/jec/softwarepatentscope.pdf> (2 November 2004).

[68] See Cover R (ed), "Open Source Development Labs (OSDL) Announces Patent Commons Project," *Cover Pagesˆ,* 10 August 2005, <http://xml.coverpages.org/ni2005-08-10-a.html> (8 August 2006).

[69] The motives of IBM in particular should be questioned, given the small size of their pledge compared to the size of their portfolio, and the prominence of their role in pushing the pro-IP big business agenda. On the latter point, see Drahos, above n13, at 169-173.

[70] <http://osapa.org/> (4 June 2006).

[71] "Overview," <http://osapa.org/information.html> (4 June 2006).

[72] See <http://pubpat.org/> (18 January 2005).

patents on the JPEG image format,[73] the drug Lipitor[74] and the FAT file system.[75] However, this is a mere drop in the vast ocean of software patents, and amounts to little more than "swatting mosquitoes to cure malaria".[76]

Although these many alternatives have a degree of merit, the common element they share is that each one is targeted at a particular symptom of the software patent problem. None of them remove the problems entirely, as they fail to address fundamental inconsistency between the patent paradigm and the nature of software and the software industry. They are simply aimed at limiting the damage. This may be the only possible way forward in jurisdictions where software patenting has become the status quo. But in jurisdictions where the decision to award software patents has not yet been made, it must be asked whether taking on such a difficult problem is worth the trouble.

With the exception of the examination improvements, the reforms discussed so far also face a deeper problem. They are technology-specific solutions aimed at tweaking the patent system to meet the needs of the software industry. On a practical level, the consequence of this is that these reforms are likely to face strong opposition from the powerful lobby groups of other industries where the patent system is well supported, most notably the pharmaceutical industry. The best approach is to remove software innovations from the patent system entirely. It is to that approach that we turn our attention next.

Avoiding software patents, TRIPS-compliant style

An alternative protection paradigm

Ten years ago, a group of leading computer science and intellectual property scholars turned their efforts to a "normative analysis of the kind of legal protection that would be socially desirable for software and how it might best be accomplished".[77] The authors are critical of the way in which discussions of the software protection problem have focussed on adapting existing regimes, showing how such an approach leads to recurrent cycles of under- and overprotection.[78] Their conclusion was that a *sui generis* regime was the only way to correctly strike the balance between creating incentives to innovate and upholding the public interest.[79] Their proposed regime is based around the following four precepts:

---

[73] See <http://pubpat.org/Chen672Rejected.htm> (1 June 2006).

[74] See <http://pubpat.org/LipitorPatentNarrowed.htm> (1 June 2006)

[75] See <http://pubpat.org/Microsoft_517_Rejected.htm> (1 June 2006)

[76] Stallman R, above n30.

[77] Samuelson et al, above n24.

[78] Ibid, at 2356-2357.

[79] Ibid.

- "Traditional copyright protection for literal code;
- Protection against behavior clones for a market-preserving period;
- Registration of innovation to promote disclosure and dissemination; and,
- A menu of off-the-shelf liability principles and standard licenses."[80]

The preservation of traditional copyright protection is an important facet of a software protection solution, because it protects innovators against free-riding imitators, whilst not interfering with independent creation. Copyright also lies at the heart of the FOSS paradigm, and thus this regime would allow FOSS projects to continue in their current form.

It has previously been pointed out that copyright protection was sufficient for the early successes of the software industry.[81] So why go beyond copyright? The reason is that there is a differential in the level of protection which copyright offers compared to patent law. Copyright protects literal copying only, whilst patent law protects against functional imitation as well. Merely arguing for abolition without considering this weakness in the copyright paradigm opens supporters up to a valid avenue of criticism. On a practical level, we should at least entertain the possibility that software patenting would not have been pursued if copyright truly was sufficient. There is a related danger here, that the absence of behavioural protection will create pressure to adapt the copyright regime to incorporate some form of behavioural protection. Such expanded notion of copying would distort the copyright paradigm beyond its traditional scope, and could well give rise to the aforementioned ills of software patenting, this time with a much longer lifespan.

The *sui generis* system directly addresses the behavioural protection issue because, like the patent regime, it provides some protection against functional imitation, or "behaviour clones". The system works in a similar fashion to the trade secret paradigm by creating an artificial lead-time for the registrant of a grain-sized innovation, in exchange for disclosure of the details of the innovation. All those wishing to use the innovation during the lead-time must pay a royalty to the innovator, or else wait out the liability period.[82]

---

[80] Samuelson P, Davis R, Kapor M & Reichman J, "A New View of Intellectual Property and Software," (1998) 39(3) *Communications of the ACM* 21.<http://doi.acm.org/10.1145/227234.227237> (25 August 2006).

[81] See for example Stallman R, "For submission to the Patent & Trademark Office," 1994, <http://lpf.ai.mit.edu/Patents/rms-pto.html> (13 September 2006); Gates B, "Challenges and Strategy," Microsoft Internal Memo, <http://www.bralyn.net/etext/literature/bill.gates/challenges-strategy.txt> (13 September 2006; Brotz D, (Principal Scientist, Adobe Systems), *Public Hearing on Use of the Patent System to Protect Software Related Inventions*
Transcript of Proceedings Wednesday, January 26, 1994 San Jose Convention Center, <http://www.gordoni.com/software-patents/statements/adobe.testimony.html> (13 September 2006).

[82] Reichman has elsewhere observed that many of the major problems facing the patent system are due to a failure to protect these small grain-sized innovations. See Reichman JH, 'Of Green Tulips and Legal Kudzu: Repackaging Rights in Subpatentable Innovation' (2000) 53 *Vanderbilt Law Review* 1753.

These behavioural protections have been demonstrated by Reichman to be equivalent to patent protection in that no one is worse off under this system than they would be if software patents were available.[83] Registrants are protected against functional imitation for a period consonant with the market life of their product. Follow-on innovators contribute to the research and development costs of upstream inventors, without the inconvenience of the large transaction costs of licence negotiation. Developers of independent products can stand on their own in the marketplace, without fear of ambush. Thus, it is submitted that such a system achieves "the promotion of technological innovation and to the transfer and dissemination of technology, to the mutual advantage of producers and users,"[84] albeit in a slightly different fashion to the patent system. In fact, the software industry is likely to be better off in that this system alleviates many of the difficulties of software patenting described above. Thus, we can say that software innovations are protected "without discrimination" and thus in a TRIPS-compliant fashion.

Further, support for the TRIPS-compliance of the regime comes from its similarity to the Circuits Layout protection regime set out in Articles 35-38 of TRIPS. This similarity perhaps owes its origins to the high degree of substitutability between hardware and software, and the recognition given to the importance of openness in the formulation of that regime.[85]

Samuelson *et al* did not elaborate on the exact details of how such a system should work, preferring to focus on the framework of the system with a view to "facilitat[ing] and direct[ing] the political debate".[86] A question thus arises as to how such a market-oriented system could be configured to meet the needs of the local industry. The first stage in such a process would be the gathering of empirical data as to the life cycle of software innovations. Such a process would be enhanced by a consultative process, gathering the views of all interested parties in the national software market. The recent Australian Law Reform Commission inquiry into availability of gene patenting provides a model of how this might work.[87] To bolster the TRIPS-compliance argument, the best approach for the initial iteration would be to err on the conservative side, ensuring that those holding software patents were not disadvantaged.

The key to the ongoing success of this system, however, is to make use of Internet technologies both to provide easy access to software innovations, and to gather data on the use of these innovations. This data should be used to regularly assess and tweak system parameters, for example, altering the

---

[83] See Reichman, above n82, wherein Reichman considers the market life of a hypothetical innovation in the plant-breeding industry, showing how the original innovator is adequately compensated for his contribution to the industry, whilst open access facilitates follow-on innovation. Reichman's conclusion is that rather than just providing equivalent protection, some innovators are likely to be better off.

[84] TRIPS Article 7.

[85] McKeough J, Stewart A and Griffith P, *Intellectual Property in Australia*, 3rd ed, LexisNexis Butterworths, Sydney, 2004 at 268.

[86] Samuelson, et al, above n24 at 2315.

[87] See above n3.

term of protection to match the average useful life of innovations registered in the system.[88]

Conclusion

The award of patent rights is incompatible with continued innovation in the software industry. Both the nature of software and the structure of the software industry cause unique problems the patent system is not capable of dealing with. The only way that national governments can promote local software industries is to put an end to software patenting. In order to remain TRIPS compliant however, the best way forward is to offer an alternative protection scheme such as that set out above, which combines the literal copy protection of the copyright paradigm with the behavioural protection of patents, thereby offering an equivalent protection to the software industry, without the many difficulties which software patents create. Such a scheme is likely to do much more to promote local innovation, rather than simply maintaining the global status quo. As such it deserves much more detailed consideration.

[88] On the operation of a similar regime for the protection of copyright, see Fisher WW, *Promises to Keep: Technology, Law and the Future of Entertainment*, Stanford University Press, 2004.