DEVELOPMENTS

Book Review – Fixing the System: A Review of Philip Leith, *Software Patents in Europe* (2007)

*By James Gannon**

Professor Philip Leith's new book on software patents can be distinguished from the growing collection of works already written on the subject in two significant ways. First, unlike the majority of books on the subject, Leith starts with and vigorously defends a position in favour of software patenting in general. Certainly, the author is not restrained in pointing out the many failings of the current regime of intellectual property protection for software inventions; however, contrary to the majority of academic commentators,[1] Leith sees software patenting as an inherent good, a system with flaws rather than a flawed system. This perspective positions the author well to provide constructive criticism of the software patent regime without engaging in the calls for an entirely new protection scheme for computer software inventions. The second aspect of Leith's *Software and Patents in Europe* that provides a valuable contribution to the software patent debate is the book's European focus. For some time at the turn of the century, Europe was a focal point of the debate as the European Parliament was considering adoption of the, "Directive on the patentability of computer-implemented inventions"[2] (the "*EU Software Patent Directive*"). However, since the European Parliament rejected the proposed Directive by an overwhelming majority,[3] software patent law observers

---

Senior Student Editor, German Law Journal. LL.B. 2008, Osgoode Hall Law School. Email: jamesgannon@osgoode.yorku.ca

[1] See, for instance, BEN CLEMENS, MATH YOU CAN'T USE: PATENTS, COPYRIGHT, AND SOFTWARE (2005), ERIC STASIK, NOT SO PATENTLY OBVIOUS, THE BRIEF HISTORY OF PATENTING SOFTWARE IN THE U.S. AND EUROPE AND THE TROUBLE WITH PATENTS IN THE DIGITAL AGE (2007), or KENNETH NICHOLS, INVENTING SOFTWARE: THE RISE OF "COMPUTER-RELATED" PATENTS (1998).

[2] 2002/0047/COD.

[3] In her statement, the Commissioner stated that the Directive's proposed software patent model should be rejected and the EU should adopt "a non-sector-specific instrument and that it should seek the adoption of the Community patent." For the legislative history of the Directive, see http://ec.europa.eu/internal_market/indprop/comp/index_en.htm, last accessed 28 July 2008.

have returned their focus to the US.[4] While Leith certainly makes frequent reference to the American Patent and Trademark Office's practice of allowing software patents (ignoring the US model in any book on software patents would be akin to ignoring the role of the United Nations in a book on International Law), the focus here remains on the British software patenting regime in the aftermath of the *Vicom*[5] patent. Additionally, the author also deals with software patent applications in Germany, France and Italy, as well as the European Patent Office's (EPO) handling of software patent applications following the adoption of the European Patent Convention. Leith's European focus, along with the author's evident experience in computer programming and intellectual property law, and a refreshing optimism in the power of software patents to benefit the industry create a powerful combination that should be both informative and interesting to anybody with an interest in software patents from both the legal and technical worlds.

The first two chapters of the book are titled "Software as machine" and "Software as software." Here, Leith describes the current climate for software patents: that although both US and European laws expressly forbid awarding patents to what are purely software program or algorithmic inventions, the practice has emerged first in the US and later in Europe to obtain such patents under the guise of a machine invention. The author argues that this practice, which he amusingly refers to as "hardware dressing", is the root cause of many of the hardships faced by software patents since it creates a significant disconnect between the invention's core innovation and its accompanying protection. The now widespread practice of describing an inventive software program in terms of the machine on which it is run is characterized as a "legal fiction – that software and hardware become a new machine."[6] As a former programmer himself, Leith often takes the perspective of an everyday programmer in assessing the effectiveness and even sometimes the underlying rationality of awarding such patents. As he charts the history of software patents in Europe from computing's early days up to the rejection of the EU Software Patent Directive, he finds that this perspective was consistently ignored:

> "It is as though the programmer's view of technology were considered irrelevant. This is true – it was not relevant. In the

---

[4] The years leading up to the EU vote on software patents saw a flurry of analysis from high-profile intellectual property academics who normally focus on American IP law. See, for instance, Richard Stallman, *Saving Europe From Software Patents*, in Lessig et. al., FREE SOFTWARE, FREE SOCIETY 106, 108 (2002).

[5] EP0005954. "Method and apparatus for improved digital image processing", filed 1979. T0208/84.

[6] PHILIP LEITH, SOFTWARE AND PATENTS IN EUROPE 20 (2007).

> attempt to fit the new computing technology into an appropriate and patentable classification, his voice was ignored and the model which was used was that of the classical 'machine'."[7]

In the next chapter, "Software as software", the author attempts to distill a computer program invention down to its innovative core. This is perhaps the most technical chapter of the book as Leith provides several real-world examples from patent applications by companies such as IBM and AT&T. The technical details of these examples are discussed, followed by an analysis of the inventive step offered by the invention and how the application was handled (or more often mishandled) by the respective patent office. The author further develops the argument from the previous chapter that the "creative enterprise" of computer software does not lie in its attachment to a particular hardware embodiment, and that this view distracts from the true innovation that the invention provides: "the idea is not to fix any concept in concrete, but to view it as malleable, since this is where the power of programming arises."[8]

Although Leith comes out in favour of retaining patents as the primary conduit of intellectual property protection for software inventions, he takes a balanced approach in discussing policy reasons for and against the basic notion of protecting software inventions in the first place. While such debates appear in one form or another in most software patent publications, Leith is to be commended for providing a concise and balanced overview of the issues backed with relevant European examples. In short, patenting software inventions is said to promote investment in research and development, educates the public by disclosing previously unknown or non-obvious software inventions, and the production of patentable ideas can increase the valuation of smaller enterprises (SMEs). [9] Conversely, the author duly recognizes the negative aspects: that the cost of patenting takes investment away from research and development, that copyright protection alone of software inventions has often proven to be sufficient, and the practical reality that most patent offices are either incapable or have great difficulty examining software patent applications. Looking at the software patent debate from the point of view of the patent examiner is another unique perspective that underscores Leith's pragmatic approach in assessing the role of software patents, undoubtedly formed from his years of experience as both a programmer and patent

---

[7] *Id.* at 35.

[8] *Id.* at 68.

[9] *Id.* at 79. For more arguments in favour of software and business method patents, see "WIPO: Ways in Which Patents can Help Your E-Commerce Business", available at:
http://www.wipo.int/sme/en/e_commerce/pat_help.htm, last accessed 28 July 2008.

attorney. Indeed, he makes a convincing argument that it is not the patents themselves that should be blamed for the constricting effects on software development for which they have become infamous, but rather it is the mishandling of applications and lack of expertise in software design that prevails in patent offices such as the EPO.

The discussion of software patents broadens to other computer-related innovations that have also been the subject of patentability debates. Leith addresses these in a chapter titled "Algorithms, business methods and other computing ogres." As the central chapter in the book, it is here that the author is at his most convincing in attacking the "technical effect" requirement of software inventions post-*Vicom*, yet still defending the virtues of software patents in general. Leith adeptly argues that by making the "technical effect" or "hardware dressing" the focal requirement for patentability of software code, many innovative and worthwhile inventions will fall outside the scope of this requirement, while entities that would be undesirable for patenting will be considered, such as algorithms (which is tantamount to obtaining a patent on a mathematical formula) or business methods (the patenting of which serves no other purpose than stifling competition without any benefit to the public). He also defends the virtues of the EU Software Patent Directive by arguing that the small enterprises and open source communities that worked together in vigorous opposition to the Directive were working against their own interests, calling their victory "a success of form rather than substance"[10] by retaining the old *Vicom* model while stalling any real progress with the EPO vis-à-vis computer program patents. While Leith admits that ratifying the Directive would not fall into the best interest of these groups, he maintains that it would still be an improvement over the post-*Vicom* status-quo.

Throughout many parts of his book, Leith points out the numerous flaws in the way software patents are awarded at both the EPO and in individual European countries, while still extolling the benefits of computer program patentability to the industry. This critical yet optimistic approach to the subject positions him well to make recommendations for improvement and to assess whether it is either possible or desirable to build a patenting system that would allow for the patenting of "mere data processing inventions". Such forms of hybrid patent-copyright systems have been proposed in the past;[11] once again, however, it seems that Leith's proposals involve making pragmatic adjustments to the existing system rather than a complete overhaul that would result in billions of dollars in wasted efforts from the

---

[10] *Id.* at 155.

[11] See, for instance, P. Samuelson, R. Davis, M.D. Kapor and J.H. Reichman, *A Manifesto concerning the legal protection of computer programs*, , 94 C OLUM. L. R EV. 2308 (1994).

major software outfits. Leith first reviews many of the proposed alternative methods for protection of software inventions[12] and proceeds to then contrast them to his own solution to the software patent debate, which he calls the European Utility Model.[13] While any proposed alternative method for software protection would be easy to criticize, Leith makes a convincing argument for the move to a utility model for such protections in Europe, while still recognizing some practical limitations of such a "higher-level" protection method system, particularly at the application stage of the patenting process, by noting that, "the problems which have generally been highlighted as problematical in the granting of software patents will be more so in the granting of utility model protections."[14] However, he still maintains that such a system would be beneficial to the European software industry and would counter the undesirable effects of the current lower-level software protections, which the author argues, "simply retard European innovations, particularly in the SME field."[15]

There is something very wrong with the way property rights are granted to software inventions. This much is agreed upon by almost all commentators on the subject, both in Europe and the US. This is perhaps why the subject has attracted so much attention in the last decade from many in the legal community with engineering and programming backgrounds.[16] We see a broken system, and we design solutions to fix it. However, as plainly obvious as some of the flaws with current methods for software patenting may seem, it is not necessarily the design of the solution that has problems, but rather it is the implementation of that design where roadblocks undoubtedly arise. The main obstacle to efficient reform in software patenting is undoubtedly that of many competing interests. The pharmaceutical industry, dominated by a few large firms, who profit greatly from the existing patent system lobby extensively against any change to the status quo, even if such changes have little effect on their own patents. Software development, especially in Europe, has more traditionally been the realm of SMEs, who see the growing patent portfolios of a few large companies such as Microsoft, IBM and SAP as a threat to their own ability to innovate. Up until recently, these players were

---

[12] In addition to the Manifesto at *supra* note 9, Leith also reviews the EU Software Patent Directive and the Software Petite Patent Act. See M.A. Paley, *A Model Software Petite Patent Act*, COMPUTER AND HIGH TECH LAW JOURNAL 12 (1996).

[13] See P. Leith, *Utility Models and SMEs*, 2 JOURNAL OF INFORMATION, LAW AND TECHNOLOGY (2000).

[14] LEITH, *supra note* 4, 178.

[15] LEITH, *supra note* 4, 178.

[16] For an excellent, recent review of the subject by programmers-turned-legal academics, see JAMES BESSEN & MICHAEL MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK (2008).

often powerless in high-level reform debates. But as was seen during the deliberations over the EU Software Patent Directive, with help from the internet and the open source communities, these smaller firms have been very successful recently in combining their resources to form an effective counter-lobby. With so many interested parties vigorously pursuing their own vision for reform, as well as considerable international pressure primarily from the US, it is without a doubt that any progress on software patent reform in Europe will be hard-fought. It is thus a shame that having taken such a practical, programmer-centric approach to crafting a suitable fix to the software-patenting dilemma, Software and Patents in Europe is silent on how such a remedy could be implemented, either in individual European states or at the EPO. However, given the author's excellent treatment of the software patent debate in the book and evident expertise with the issues at stake, I would certainly be interested in reading any thoughts he may have on the subject.