# The Evitability of Software Patents

*W. Neville Holmes*
University of Tasmania

**Hobbling software with a medieval artifact like the patent may stifle innovation, benefiting the moneyed few at the expense of everyone else.**

An Australian patent examiner more than 40 years ago, I've maintained a sporadic interest in the law of intellectual property and been disturbed by the developing saga of software patents. Although the intellectual property laws of any two countries—such as Australia and the US, for example—exhibit significant differences, my disquiet goes far beyond such differences.

Kenneth Nichols's recent *Computer* article[1] brought several of my concerns into sharp focus. It was a splendid tutorial, and letters published in the subsequent two issues nicely augmented it. Given that Nichols expressly aimed to "outline the current state of software patenting in the US," it would have been quite unfair to take issue with him had he not led with his chin. Twice. First, he flatly asserted that "software patents are neither inherently good nor bad." Second, he declared that "software patents are here to stay."

## QUESTIONING PATENTS' MORAL NEUTRALITY

In the moral sense at least, digital technology and thus software itself clearly are neither inherently good nor bad—although plenty of *technically* good and bad software can be found. But software *patents* are not impersonal technology. The entire intellectual property system is a social artifact. As such, any part of it may be considered good or bad to some degree. Not only is it reasonable that an interested party examine the ethics and morality of any branch of the legal system, but professionals in relevant areas have a social duty to do so.

### Inconsistencies and ambiguities

Yet ethical issues are not simple. Nor is the law simple, or even internally consistent. For example, the granting of letters patent confers a monopoly, but most trade practices legislation postulates that monopolies are inherently unethical. Nichols himself writes disparagingly of the "near-monopoly positions of IBM in the 1970s and Microsoft in the 1990s."[1]

Patent law and practice is a linguistic nightmare, an important ethical point almost always overlooked in popular articles. The pseudonymous Leonard Lockhard wrote stories on the US patent system that appeared occasionally in *Astounding Science Fiction* magazine during the 1950s. Clearly an insider, in "That Professional Look,"[2] Lockhard described patent law as "not founded on any known system of logic," but founded instead on "an iron-clad, invariant system of exceptions to a set of ever-changing, quasi-existent rules." These stories delighted patent examiners and attorneys around the world, who knew that ordinary readers would find the everyday occurrences of their professional life ridiculous and incredible.

0018-9162/00/$10.00 © 2000 IEEE

### Claims the pivot

The linguistic problem springs from the role of the claims, a role Lockhard emphasized. The monopolies granted are defined only by the several claims of a patent application. The description must support each claim, but that is all. Patent procedures focus on the claims, and patent actions typically focus on the claims' precise wording. Experts in law, not technology, make the legal judgments. These experts must often interpret the meanings of words in claims in bizarre ways because they are required to respect the precedents set by relevant prior judgments, which are themselves often bizarre.

Such was the situation half a century ago, and it's worse now. Having laws and legal practices that citizens at large cannot understand is unethical and dangerous. Indeed, even ordinary lawyers can't understand patent law and so, in Australia at least, the law forbids them to write patent specifications.

## QUESTIONING PATENTS' INEVITABILITY

Nichols's assertion that software patents are here to stay, a position intellectual property attorney Gideon Gimlan[3] also endorses, could simply be taken to mean that commercial interests have the political and economic clout to make sure these profitable monopolies persist. Although this implication may well be true, it is improper for computing professionals or patent attorneys to make such political predictions. Such prognostications are best left to the historian or political scientist.

On the other hand, Nichols's assertion could, despite his disclaimer, be interpreted by a naïve or careless reader as an affirmation of the rightness and goodness of software patents, particularly since the popular press so often publishes similar assertions with jubilation rather than qualification. Several questions must be answered before we can accept this second interpretation. First, what is a software patent? Second, are distinctive software patents justified? Third, should the patent system be extended to provide monopolies for distinctive aspects of digital technology? It is both responsible and professional for computing and intellectual-property-law professionals to pursue these questions.

## THE SHAKY CASE FOR SOFTWARE PATENTS

The common use of the phrase "software patents" implies that they are a distinctive patent class. Indeed, Nichols claims that software requires separate treatment. But does the US grant distinctive software patents? The relevant literature seems to indicate that the basis for software patents in the US derives not from legislation, but from a new set of guidelines issued in 1996 by the US Patent and Trademark Office.[4] These guidelines "do not have the force and effect of law," and do not themselves mention software patents as such.

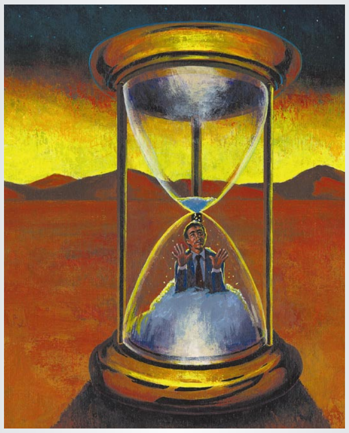
**Nichols led with his chin twice. First, he asserted that "software patents are neither inherently good nor bad." Second, he declared that "software patents are here to stay."**

Traditional thinking would deny patentability to software per se, as it has been denied to mere algorithms. In response to Nichols's article, US patent attorney Bruce E. Hayden[5] asserts that software is best claimed under three of the four existing types of utility patent claim. "Processes are methods of doing something, and translate very easily from software. The typical machine claimed is a general-purpose computer executing the subject software algorithm. Finally, article of manufacture claims cover a medium ... containing the software."[5]

As Gimlan observes,[3] in practice software cannot be distinguished strictly from hardware: What software can do, hardware can do equivalently. Since a software process can be routinely converted to hardware, at least in principle, there is no reason to restrict a process claim to its software implementation. Note that the monopoly of a process claim covers the use of the process.

A machine claim grants a monopoly that applies to the use of the machine carrying out a process, not to the process itself, and certainly not to the algorithm. Indeed, under traditional English patent law at least, mere algorithms cannot be patented. There must be a vendible product. If the machine claim merely covers a general-purpose computer performing a particular process embodied in some particular software, the restriction to a general-purpose computer is pointless and not in the inventor's interests.

The 1996 US PTO guidelines seem to be silent on article-of-manufacture claims. I infer that they claim a monopoly on the article's use—the medium containing the software—as part of the process or



**Patent protection for digital technology lasts 20 years—an eternity in software development. This span virtually ensures that no public benefit of the invention's free use after the patent expires will ever be realized.**

machine of the invention. Otherwise, copyright law would be the appropriate monopoly source.

No patentable invention could reside in the software itself, just as no patentable invention could reside in any particular pattern of holes punched in the cards used to operate a Jacquard loom. Traditional English patent law explicitly denies patentability to a "mere scheme or plan," and the US guidelines suggest that a similar principle applies there. In this case the appropriate source of monopoly would be industrial design registration.

## The self-modification argument

So far, then, there seems to be no need to make specific provisions for software in patent law. Both Hayden[5] and Jim Geringer,[6] another US patent attorney responding to Nichols, appear to agree on this point. But Geringer finds bemusing Nichols's assertion that "'self-modifying' software cannot be adequately described by the 'device-process' model prevalent in patents." Nichols's response that "This ability of software to modify itself has no parallel in the patent's world of physical devices and processes" ignores the venerable art and science of servomechanisms. For example, one servomechanism, the governor, brought the steam engine under control and led, in some historians' opinions, to the Industrial Revolution. The governor functioned as that part of the engine that, in response to an alteration in the engine's speed, changed its physical configuration to bring the speed back into the desired operating range.

Ultimately, the patentable invention lies in the self-modifying process or machine—the modification is the product of the invention, not the invention itself. An automatic product is not, by definition, inventive and so cannot be validly claimed per se in a patent application. Or so common sense would have it.

## The new-entity argument

Nichols claims that "Software is a new kind of entity, with the ability to transform all other technologies, including the creative arts, politics, and economics. It therefore requires separate treatment."

The implied major premise of this syllogism is that new kinds of entities need separate treatment. When I was examining applications for patents, one of the classes of invention I was responsible for was electrical connections. At the time, a completely new kind of connection—the crimped connection—came into vogue, accompanied by a torrent of patent applications. No one suggested for a moment that this new and different kind of connection needed separate treatment. Software is a mechanism and differs from other mechanisms in the same limited way that crimped connections differ from other connections. Anyway, software isn't particularly new: Shortly after leaving my job as a patent examiner, I found myself busily plugging and later coding programs for a living—40 years ago.

Software is just one facet of digital technology, and, in the long run, possibly not the most significant one. Digital technology has clearly become important. Should the patent system therefore be extended to provide monopolies for distinctive aspects of digital technology? We cannot answer this question until these distinctive aspects appear.

## The distinctive-features argument

To date, digital technology's main distinctive features may well be those that Nichols lists as practical shortcomings:

- Take-up of digital products or protocols occurs so quickly that if the first to market enjoys patent protection, competing products can easily be excluded.
- The wide distribution and varied uses of digital products make patent infringements extremely difficult to detect, prove, and prosecute for.
- Digital technology's ubiquity makes it nearly impossible for people using the technology, directly or through the Internet, to determine whether they are infringing a patent or not, even with the best will in the world.
- With ideas and software proliferating so swiftly and invisibly to so many recipients across the

Internet, any patent infringement will likely spread quickly and widely among "small violators," whom it will be difficult, expensive, and unprofitable to pursue.

If these distinctive features justify separate treatment for software patents, it is far from clear what separate treatment would eliminate the shortcomings. Neither Nichols nor his respondents have anything to propose, although Hayden dismisses the problems as "speculative in nature and ... not borne out by experience,"[5] a shade prematurely, I think. The absence of evidence of guilt is not proof of innocence, in logic if not in law.

Whether they are shortcomings or distinctive features, these four points make a case for the removal of patent protection from at least some kinds of digital inventions, or at least for a drastic reduction in their term. Nichols verges on making this suggestion himself when he points out that patent protection for digital technology lasts 20 years—an eternity in software development. This span virtually ensures that no public benefit of the invention's free use after the patent expires will ever be realized. Yet this public benefit is supposed to be the sole compensation for the otherwise unethical monopoly the patent grants.

Along similar lines, some have advocated abolishing the application of copyright to digitized materials altogether. John Perry Barlow[7] describes copyright as protecting the bottle, not the wine, and observes that in the digital world we no longer need the bottle. And truly, the common availability of photocopiers, scanners, and color printers makes policing even general copyright farcical.

## THE CASE AGAINST SOFTWARE PATENTS

The arguments against software patents are many and, to a degree, also apply to software copyright. Both Nichols and Barlow advance the most obvious one: Such intellectual property protections are impractical. Digital representations of programs or other text can be copied or otherwise manipulated with trivial ease; their transmission presents an insubstantial pageant which mocks any pretension that we can control it.

A more serious argument goes to the heart of the matter. *Intellectual property law is founded on securing the public good, and software patents or copyright plainly do not secure the public good.* The justification for granting a monopoly to an inventor or importer of a technology has long been that the grantee trains others to exploit the invention after the grant expires. Thus, the inventor or importer gets a short-term benefit from the monopoly, while the public gets a greater long-term benefit from the invention's later, unfettered use.

Yet the public gains no more benefit—and arguably

## Software Patent Resources

There are two kinds of Web resources for information on intellectual property law related to computers: *questioning*, usually compiled by outsiders, and *unquestioning*, usually compiled by insiders. Sites that question the law are fewer, and so appear first here.

### Questioning Sites

**The Electronic Frontier Foundation's Intellectual Property Online Archive**
http://www.eff.org/pub/Intellectual_property/
Many articles covering the legal aspects of intellectual property in software or digital media formats; also contains links to relevant Web sites.

**The Linux Journal Article Archive**
http://www2.linuxjournal.com/articles/
The *Currents* section includes articles on software patents, for instance *currents/014.html*. The recent article *currents/016.html* is also relevant and interesting.

**United Nations** *Human Development Report 1999*
http://www.undp.org/hdro/report.html
Chapter 2, "New Technologies and the Global Race for Knowledge," examines the implications of current technology trends across the globe and warns that some developed nations may be using patents and other intellectual property protections to stifle development in less technologically advanced countries.

### Unquestioning Sites

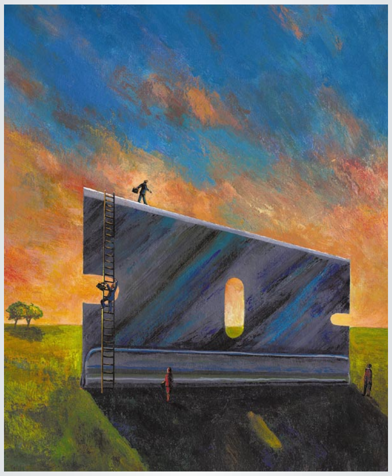
**As Software Patents Take Over, Expertise Is Key**
http://www.ljextra.com/practice/computer/0420softpat.html
Suggested guidelines for obtaining software patents.

**Avoiding Intellectual Property Problems**
http://www.fplc.edu/tfield/avoid.htm
General advice on measures you can take to guard your intellectual property and forestall patent infringement problems.

**Patents for Software-Related Inventions**
http://www.tkhr.com/articles/pat4sof.html
Background on the history and intent of US patent law; analysis of software patent case law and precedents; detailed, specific advice on maximizing intellectual property protection for software-related inventions.

**Protection of Software-Related Inventions in Europe and Japan**
http://www.ladas.com/GUIDES/COMPUTER/Computer.EPOJP.html
**Statutory Protection for Computer Software in the United States**
http://www.ladas.com/GUIDES/COMPUTER/Patents.USA.html
These two articles examine from different regional perspectives the issues that impinge on software intellectual property.

**Software Patent Institute Software Technologies Database**
http://www.spi.org/
Home page for a nonprofit corporation that provides courses and prior art about software technology to help improve the patent process.



It is sad that the US government overlooks that the basic arguments in favor of free trade, which it supports so fervently in the World Trade Organization, apply equally to the free movement of ideas, which it opposes so avidly through the WTO and the World Intellectual Property Organization.

far less—from the patent system in its current form than it would enjoy if no patent system existed. The current situation benefits business concerns exclusively, either giant corporations that seek advantage over their competitors, or opportunistic corporations that acquire software patents solely to extract licensing fees from their competitors. As Brian Kahin[8] observes, the whole situation is a dreadful mess.

Given the volatility of software development and the industry's speed of innovation, software's long-term benefit (if any) to the public is of much shorter duration than the benefit of the monopoly a software patent grants. Worse, corporate pressure and, internationally, US government pressure have forced the extension of patent monopoly terms to 20 years, further reducing if not eliminating the possibility of public benefit.

With regard to the public benefit of copyright—at least in the Jeffersonian US—the intent was that "ideas should freely spread from one another over the globe." Thus, the intended public benefit is now absent from copyright as well. It is sad that the US government overlooks that the basic arguments in favor of free trade, which it supports so fervently in the World Trade Organization, apply equally to the free movement of ideas, which it opposes so avidly through the WTO and the World Intellectual Property Organization.[9]

The justification typically given nowadays for special software protection is that, without it, no one will produce any software because programmers will have no incentive to do so. Clearly, this is self-serving nonsense. Corporations have long paid programmers to produce software in the absence of software protection. Open source software continues to thrive in the open and even to outpace closed, commercial competitors.

Perhaps the strangest thing about discussions of intellectual property law and digital technology is that the two least practical forms—patents and copyrights—get the most attention, whereas the two most appropriate forms get the least. Industrial-design registration gives a monopoly over a vendible product's visual appearance, while trademark registration gives a monopoly over a mark used to identify a product's source. Both these protections aim to prevent unfair competition; neither blocks the free spread of ideas and techniques.

The evitability of software patents does not ensure they will be avoided. However, we can make good arguments for avoiding software patents, and for other more practical and just forms of monopoly for software. If members of the computing profession widely disseminate those arguments, there is at least some chance we *can* avoid such patents. ✷

## References

1. K. Nichols, "The Age of Software Patents," *Computer*, Apr. 1999, pp. 25-31.
2. L. Lockhard, "That Professional Look," *Astounding Science Fiction*, Jan. 1954, pp. 96-110.
3. G. Gimlan, "Relax, Not Everyone Is Out to Sue You," Letters, *Computer*, May 1999, p. 4.
4. US Patent and Trademark Office, "Examination Guidelines for Computer-Related Inventions," http://www.uspto.gov/web/offices/com/hearings/software/analysis/computer.html.
5. B.E. Hayden, "The Claims Hold the Key," Letters, *Computer*, May 1999, pp. 4, 6.
6. J. Geringer, "Are Software Patents Really Different?" Letters, *Computer*, June 1999, pp. 7-8.
7. J.P. Barlow, "Selling Wine Without Bottles," http://www.eff.org/pub/Intellectual_property/idea_economy.article.
8. B. Kahin, "The Software Patent Crisis," *Technology Rev.*, Apr. 1990, pp. 52-58.
9. "The Standard Question," *The Economist*, 15 Jan. 2000, p. 91.

*W. Neville Holmes is a lecturer under contract at the University of Tasmania's School of Computing. Contact him at neville.holmes@utas.edu.au.*