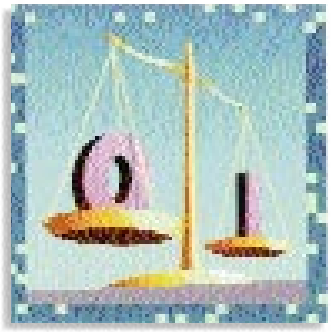
# Software Patents: Protection We Don't Need



Software may be a unique class of inventions that shouldn't be patentable.

In recent months, the Java programming language from Sun Microsystems has taken the industry by storm. Microsoft, Netscape, Borland, Lotus and dozens of other companies have licensed Java for use in Web browsers and other products. This new object-oriented language promises to assume a place next to C++ in importance in our industry and to become the premier language for World Wide Web development.

There is latent trouble in paradise, however. Here is an excerpt from a recent press release:

> Eolas Technologies, Inc. [of Chicago], announced today that it has completed a license agreement with the University of California for the exclusive rights to a pending patent covering the use of embedded program objects, or "applets," within World Wide Web documents. Also covered is the use of any algorithm which implements dynamic bidirectional communications between Web browsers and external applications.

If this patent is issued, Eolas will have the right to demand that Sun, Microsoft and the others either stop using or distributing embedded Java applets, or pay it a royalty. If this effort succeeds, the end result will be the exact opposite of what the patent laws were designed to do.

The U.S. Constitution provides that Congress has the power to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries." Patent and copyright laws are based on this constitutional authorization. While the copyright law gives a 75-year monopoly on the particular expression of an idea, patent law gives a 17-year monopoly on the use of the idea itself.

## Some Examples

Let's examine two hypothetical examples to make this clear. Company A publishes a windowed operating system for personal computers. Company B does the same. Company A has a copyright in its operating system but no patent. Company B created its work from scratch and did not copy any of Company A's code, command set or icons. Company B has not infringed Company A's copyright.

However, if Company A had obtained a patent on its invention, the "idea" of a windowed operating system for personal computers, Company B would be unable to field its own completely original product. For 17 years, Company A would have the exclusive right to be in the windowed OS marketplace, unless it chose to grant a license to someone else.

Broadly speaking, copyright arose to protect artistic expressions—such as painting, writing and photography—and patenting grew up to protect the products of engineering—printing presses, cameras and so forth. Software is unique because it straddles the boundaries between these two categories. During the 1980s, courts recognized (at first haltingly, then emphatically) that software was a proper subject for copyright protection, just like a book or picture.

Courts were much slower to apply patent protection to software. During the infancy of the computer software industry, patents were denied on the grounds that software only solved algorithms, and that an algorithm, as a law of nature, is no more patentable than the law of gravity.

## Benchmark Ruling

The breakthrough case was Diamond v. Diehr, in which the U.S. Supreme Court recognized that a patent should not be denied to an otherwise patentable process—in this case, a rubber-curing machine—just because software (which decided when to stop the curing process based on analog readings) was a part of the invention. By the mid-1980s, the floodgates had broken, and the U.S. Patent and Trademark Office (PTO) would accept patents for almost any type of program, as long as it met the twin requirements of being "novel" and "nonobvious."

*Novelty* in this usage means that the invention represents something new, an improvement upon prior art. An applicant for a patent is required to disclose to the patent examiner any prior related inventions of which he or she is aware, whether reflected in a patent, a technical article or some other form of disclosure. A patent can be invalidated at any time after issuance if "prior art" comes to light which the applicant failed to disclose.

*Nonobviousness* is a second hurdle the applicant must still surmount after showing novelty. There are some inventions which apparently are too obvious to patent, even though no one else has thought of them before. The leading case on this issue—taught in law school class-

By Jonathan Wallace

es on the subject—involves the wooden frame that used to be used to drag groceries down the checkout counter in the days before conveyor belts. It was "novel" but too "obvious" to patent.

Once the PTO began accepting software patents, it soon acquired a reputation for accepting almost anything presented to it. It simply didn't have the track record or the experienced personnel necessary to weed out applications for software that was not really novel. Everyone in the industry knows that there are patents floating around (some of them belonging to IBM) that cover basic, obvious and old innovations in the programming art, such as:

- a data entry screen with mandatory and optional fields;
- a spreadsheet in which each cell has a "next cell" attribute defining the next cell to which the cursor should jump after data has been entered in the current cell;
- a word processor that allows you to shade portions of text by enclosing it within commands that turn shading on and off;
- a "backing store" function that saves the position and contents of a window when it is not visible.

## Not as Simple as It Seems

Some readers may feel that the obviousness of these innovations is a matter of hindsight. Someone labored to think of them first; why shouldn't the original inventor be rewarded with an exclusive?

The answer is threefold. First, the individual holding the patent on many software innovations is not actually the inventor. The PTO is known to have issued many bad patents in ignorance of prior art that might have been obvious to examiners skilled in computer science. If the owner of a bad patent brings a lawsuit against other users of the innovation, they will typically incur hundreds of thousands of dollars in legal fees, and may even be driven out of business, before proving that the plaintiff is relying on an invalid patent.

Second, the 17-year monopoly granted by a patent is completely out of synch with the fast pace of the software industry, where products are obsolete within a year or two. Third, the early history of our industry proves that we had far more innovation without patent protection than we would have had with it.

Some years ago, I attended a talk by Dan Bricklin, codeveloper of Visicalc, the first spreadsheet program, which was released around 1980. I asked him if he had ever thought about patenting his product. He replied that he was eager to but that his attorney had advised him (correctly, based on the state of the law at the time) that it was not patentable. He added, "In retrospect, I'm glad I couldn't, because I would have strangled an industry." Just think about it: If Bricklin had gotten a Visicalc patent, there would have been no Lotus 1-2-3, no Quattro Pro, no Excel. Does anyone think that Visicalc, occupying a field with no competition, would have evolved as much as these other spreadsheets have?

Imagine if every one of the leading software products of the early 1980s had been patented. You would still be storing your data in dBase; there might be no Access, Paradox, Sybase or Informix. You would be word processing in WordStar; WordPerfect, MS Word and FrameMaker would not exist.

Software publishers were always happy with copyrights, until they knew patents were available. Even today, many industry innovations are only possible because publishers refrain from getting patents or (in some cases) hold back from enforcing the ones they have. If NCSA had patented Mosaic, Netscape and Hot Java would not be possible.

Although some opponents of the present system propose a shorter software patent of a year or two, the early experience of our industry proves that software publishers do not need patents to thrive and grow, and that users receive a wider variety of innovative products without them. The patent laws should be amended to exempt computer software. IT

***Jonathan Wallace*** *is vice president and general counsel of Pencom Systems, Inc., in New York City. He can be reached at jw@pencom.com.*

## The Analyst's Couch

### Plenty Is Great About Electronic Commerce

(CONTINUED FROM PAGE 64)

*cybernetic worker*. More and more, these workers will become their own resource with their own special skills. They will make up the organization based upon tasks to be accomplished. Once a job is complete, they will become part of a new organization or task. Thus the creation of the Virtual Enterprise. New skills, coupled with global online communications, make it happen. The migrant cybernetic worker can live where he or she wants and work where he or she lives. He or she also will have more time to smell the flowers and look directly into the eyes of a child. IT

***Jim Johnson*** *is founder and chairman of the Standish Group International, Inc., in Dennis, MA.*

*Did something in this column press one of your hot buttons? Then let us hear what you think by sending a response to pubs@uniforum.org. We'll consider it for publication in "Letters to the Editor."*