Debunking the Software Patent Myths

Jealousy and Envy deny the merit or the novelty of your invention; but Vanity, when the novelty and merit are established, claims it for its own . . . One would not therefore, of all faculties, or qualities of the mind, wish for a friend, or a child, that he should have that of invention. For his attempts to benefit mankind in that way, however well imagined, if they do not succeed, expose him, though very unjustly, to general ridicule and contempt; and if they do succeed, to envy, robbery, and abuse.
BEN FRANKLIN, 1775 [6]

PAUL HECKEL

The issue of software patentability is an important topic because it affects the environment in which programmers and designers work, software innovation, the health of the software industry, and U.S. competitiveness. While the writing of this article was motivated by "Against Software Patents," by the League for Programming Freedom in the Jan. 1992 issue of *Communications* it is an overall defense of software patents.

An Absurd Patent

U.S. Patent 4,736,308, the first patent under the heading "Absurd Patents" in "Against Software Patents," is described: "For example, Apple was sued because the HyperCard program allegedly violates patent number 4,736,308, a patent that covers displaying portions of two or more strings together on the screen, effectively scrolling with multiple subwindows. Scrolling and subwindows are well-known techniques, but combining them is apparently illegal." The League calls this an "outrageous result." Based on this description alone, any reasonable person would have to agree.

But I am that inventor and Apple was actually sued on a prior related patent, 4,486,857. Because my patents were misrepresented, I researched the other patents described in the League's article and am reporting my results.

There is much the League did not say about my patent and the circumstances surrounding it. First, it did not describe my background. In 1963 I worked on the software for the first computer designed to be a timesharing computer. I was at Xerox PARC in its early days, wrote two articles for *Communications* [17,20] and a book on user interface design [16]. My patent covers a commercial product called Zoomracks [19], which introduced a new computer metaphor called the card and rack metaphor. Zoomracks was marketed primarily on the Atari ST. Zoomracks developed a strong base of users who used it for a very broad range of applications, but it was a financial struggle largely because Atari did poorly. In Aug. 1987, Apple Computer introduced HyperCard, which is based on a similar, but more limited card and stack version of the metaphor.

I was then faced with having invested six years of raising money, developing a product, marketing it, and proving its value in the market, only to find I was in debt, my customer base was on a dying computer and Apple was giving away *free* a more polished and featured, although less elegant, version of the metaphor. While Apple may not have set out to rip off Zoomracks, it was aware of Zoomracks (having seen it under nondisclosure), of HyperCard's similarity to Zoomracks, and that Zoomracks was protected by patents.

HyperCard created expectations that Zoomracks could not meet, and other companies began to develop HyperCard clones. Meanwhile, I asserted my rights, sued and settled with Apple, licensing the patents. Apple is to be applauded for respecting my patents.

IBM was less respectful: We had twice brought our patent to its attention with respect to products like HyperCard and we had visibly asserted our patents and sued and settled with Apple by the time IBM decided to bundle what many consider to be a HyperCard clone. If this article has an anti-IBM patina to it, it is because I spent six months patiently trying to deal with IBM. Finally, IBM representatives flew to San Francisco to show us prior art—earlier technology—invalidating our patents that they claimed to have. When they arrived, they refused to show us the prior art, "for fear the patent office would recertify our patents in error." Even if IBM had been straightforward with me during the six months, to accept such an assertion without evidence would have been naive.

Faced with a choice of accepting IBM's offer of 0.2% of the $5 million IBM is said to have paid to license the token ring patent, or to accept its challenge to "sue us" if we wanted to see the prior art, IBM left me no choice but to fight. But I have chosen to fight in the court of public opinion where possible, rather than the civil courts where, because of its financial strength, IBM has the detailed advantage. I added a description of my dealings with IBM to my book [16] and later sent copies of my book to the members of the Commission on Patent Reform when they asked for comments.

Based on my experience I formulated Heckel's Principle of Dealing with Big Companies: *There is no such thing as a free lunch; unless you're the lunch.*

With Apple and IBM, I did battle against large companies who were sophisticated about intellectual property, rather than small ones that were not. I felt it was in everyone's interest to force companies and the courts to make decisions about software patents so the rules and the marketplace realities can be clear to all, not just the sophisticated few. This article is written in that same spirit. While it is a personal issue, I write to clarify the software patent issues in general, to raise the level of discussion and because like most good inventors, I am curious about what the truth is.

One can only understand the need for patents in light of the competitive marketplace. We need a heavy to show what the innovator faces, just as Humphrey Bogart needed Sidney Greenstreet in *The Maltese Falcon*. IBM has already presented itself in that role; it will reappear as did Sidney Greenstreet.

The Informed Opinion

We will visit the other eight patents mentioned by the League in its article and show that the patents it selected, on examination, disprove its case. But first, we take the broader view.

Should software be patentable like other technologies? The primary issue is a policy one and so we have been influenced by Neustadt and May and their book on governmental decision-making [33]. We ask: What analogies (to software) exist? What are the similarities and differences? What are the assumptions, explicit and hidden? "What is known?" "What is the history of the issues?" What are the interests of the various players?" We will follow the Goldberg rule and ask, not "What is the problem?" but "What is the story?" Most important, we should ask, "How did things turn out in the past?"

History and innovation economics, more than law and computer science, must be the foundation on which to make policy. We have framed 10 points which are, we believe, the consensus of informed opinion on software patents. We hope they help you crystallize your thoughts on patents and enable you to better articulate your differences, if any, with the informed opinion.

1. By creating property rights, patents promote innovation in non-software areas. They particularly promote innovation from small and mid-size companies.
Most of the arguments against software patents turn out to be arguments against patents *per se*. These arguments are advanced most credibly on the basis of established technologies where data and research already exist.

Patents have been accepted around the world as promoting innovation. Many giants of U.S. industry such as G.E., AT&T, Polaroid, Xerox and Hewlett-Packard, started as small companies that used patent protection to protect their inventions.

Yet, most of the articles on patents in the trade, business and even academic press read by the computer community [5, 13, 15, 26, 27, 28, 41, 42, 48] have an antisoftware patent bias. The reason is that for every patent there is one patentholder who is reluctant to speak because the issue is complex and what someone says could be used against him in litigation. And there are a dozen who might like to use the patented technology without paying for it and so are willing to malign the patent and patent system and pass on unsubstantiated rumors and misinformation.

Economists have researched innovation in other technologies [24, 30, 31, 41] and found the following: patents encourage innovation; and small entities—individual inventors and small companies—are a very important source of innovation. According to Jewkes et al. [24],

It is almost impossible to conceive of any existing social institution so faulty in so many ways. It survives only because there seems to be nothing better. And yet for the individual inventor or the small producer struggling to market a new idea, the patent right is crucially important. It is the only resource he possesses and, fragile and precarious as his rights may be, without them he would have nothing by which to establish a claim to a reward for his work. The sale of his ideas directly or the raising of capital for exploiting the ideas would be hopeless without the patent.

While several articles discuss software patents and copyrights [8, 46, 47], few have been written for the software, as opposed to the legal, community [11, 16, 37]. Such studies have only rarely been linked to software [7], and, we are unaware of any empirical studies of the effect of software patents on innovation other than this one.

If we are to reject patents in principle, we should argue that case. If we accept patents as promoting innovation elsewhere but not in software, then we should differentiate software from other technologies.

2. Patents have evolved to address concerns raised by those who suspect software patents.

The courts have developed a patent jurisprudence as a unifying mechanism to support many technologies and foster evolutionary improvement while balancing the rights of patentholders and potential infringers.

Patents have a long history (see sidebar). Most of the concerns about patents raised by the League have been raised long ago in the context of other technologies and addressed in case law and legislation and have stood the test of time. The patent system, like MS/DOS, is not perfect. MS/DOS has a long history of evolutionary improvement: It is a derivative of CP/M, which is a derivative of TOPS-20, which is a derivative of the SDS-940 timesharing system, which evolved from the first timesharing system developed at BBN about 1960. Patent jurisprudence has a similar history of evolution.

Part of the value of patents is they are a proven, public domain *standard* of intellectual property protection having a history of improvement over 500 years, compared to the 30 or 40 years of expe-

A Brief History of Patents

Until recently patents were thought to have originated in England and been used only there prior to America, an error propagated by Jefferson, Lincoln and as late as 1948 by the Supreme Court. Recent scholarship shows their Italian origin and their early use in France, Germany, the Netherlands as well as England. Venice granted 10-year monopolies to inventors of silkmaking devices in the 1200s. These early patents were *ad hoc* grants. In 1474 Venice passed its first patent statute. It recognized patents as a matter of right, rather than royal favor, and provided for fines and the destruction of infringing devices. Galileo was granted a patent [6]. In England, the Queen granted so many monopolies to her friends that citizens protested; so England, in 1624, passed the Statute of Monopolies. This prevented the granting of monopolies, but gave people the right to obtain patents on inventions and imports new to the realm. This statute distinguished between monopolies, which it outlawed as taking from the public what it already had, and patents, which it permitted as giving to the public what it did not yet have.

These patent laws were enacted at the end of the dark ages just before the Renaissance in Italy and the Industrial Revolution in England, suggesting that they *stimulated* innovation.

The Founding Fathers also believed that inventions (and writings) belong to their creators inherently—rather than to the state to be granted at its pleasure. This principle was embodied in the Constitution [1, 6], where Article 1, Section 8 says,

> **The Congress shall have the power to promote the progress of science and the useful arts by securing for a limited time to authors and inventors the exclusive right to their respective writings and discoveries.**

Congress has the power, not to grant rights but to secure inherent rights. This is the principle expressed in the Declaration of Independence that "all men are endowed by their creator with certain unalienable rights." In the Federalist Papers, James Madison, in describing the patent powers observed that, "The public good fully coincides . . . with the claims of individuals.

The creators of the Constitution knew history and understood the ways of men and women, and patents—9 of the 13 colonies granted patents. We should give weight to findings of fact embodied in the Constitution. Two concern patents: Patent rights are inherent rights like freedom of speech; and patents promote innovation.

Obviousness: Polaroid vs. Kodak

As an example of how the patent system has evolved to address the concerns the League raises, consider the Polaroid patent which, they say, describes "differences in the number and order of layers of chemical in a film—differences between the technique Kodak was using and those described by previous expired patents." The League says such differences were obvious. The court held otherwise. Kodak could have avoided infringement by using the order described in the earlier, expired, patent the League refers to. Why would Kodak use the new order described in the later patent rather than the earlier one? Why would Polaroid patent it?

Could it have been better?

This demonstrates three things to those who must deal with patents. First, an active patent is a territorial warning. Second, technology described in expired patents is in the public domain. Third, patents protect the innovator. Polaroid was the innovator in instant photography. Kodak wanted a share of that market. The major obstacle was Polaroid patents. Kodak tried to get too near the fire. Kodak got burned and paid Polaroid over $900 million.

rience developing operating systems. TopView and OS/2, demonstrate how developing a new operating system and crystallizing a new infrastructure around it are fraught with dangers—known and unknown. An infrastructure has crystallized around MS/DOS. It includes developers and consultants who know it, books explaining its use, and commercial products based on it. Similarly, an infrastructure has crystallized around the patent system. It includes patent lawyers, case law examples of valid and invalid, infringed and not infringed patents, and books and articles explaining patents to both lawyers and nonlawyers.

3. Patents are not perfect.
There are problems with the patent system. Only that which is not real is perfect. The patent community and the Patent and Trademark Office (PTO) are aware of the problems and have been working to address them. A Commission on Patent Reform is considering improvements such as better examination procedures and automatic publication of patent applications after 18 to 24 months.

If lack of perfection were a reason to get rid of something, no one would survive his or her teenage years. Other industries find patents useful in spite of these problems; software will too.

Patents, it is said, inhibit standards. They do not; they inhibit the expropriation of intellectual property without just compensation in violation of the Fifth Amendment. Where patents exist standards are created in two ways:

- Where people want a standard that infringes a patent, the standards body usually negotiates an agreement whereby the patentholder in return for having his or her technology required as part of a standard, agrees to make a standard license and rate available to all.
- Often standards are agreed to which do not infringe any intellectual property. The QWERTY keyboard and the standard automobile controls (Steering wheel, brake and accelerator) demonstrate that patents don't inhibit standards creation. Both public domain standards were developed during the working lifetime of Edison who received 1,100 patents.

4. Software is not inherently different from other technologies in the way innovation or patents work.
Arguments that software is different should be treated critically; you can be sure those same arguments will be used by those who do not believe that the protections of the Bill of Rights extend to areas where computers and software are used.

Fred Brooks, following Aristotle, suggested the distinction between essence and accident [3], and that distinction has guided our analysis in the issues raised by the League and the academics (see sidebar). The question is whether the differences between software are essential or accidental in their encouraging innovation. The League says software is different and so should be protected differently. They present two arguments.

A. ***Programs are complex.***
Why, so they are; but so are airplanes, silicon chips, silicon chip fabrication plants, potato chip plants, oil refineries and many things. But people find the patent system beneficial in these other technologies.

B. ***Software is cheap to develop compared to other technologies because it is a cottage industry.***
Other industries have cottage manufacturers and they deal with patents. Outside of software much invention is a cottage industry; about 5,000 independent inventors belong to the 37 organizations that are members of the National Congress of Independent Inventors. And most cottage industries do not rely on invention.

We should no more optimize an intellectual property system for cottage developers than we should for Fortune 500 companies.

When one talks about marketing and maintaining commercial software products, the costs are much greater than the estimates made by the League. At the other extreme, IBM is reported to have spent 2.5 *billion* dollars to develop OS/2, including applications.

It is expensive to develop software if the task is to design it from scratch and make it a success in the market; it is cheap if the task is to clone something that already exists or is precisely specified.

Indeed, that clone software is so

much cheaper to develop argues for the necessity of patent protection if one wants to stimulate the development of products worth cloning.

Making software nonpatentable or subjecting it to a different form of protection creates practical difficulties, rather like a state seceding from the Union and setting up check points on its border. And if one state secedes, they all can. If each technology has its own *sui generis* (unique) form of protection, we would have to set up boundaries between the different technologies and would need rules for what happens at the boundaries.

This situation occurs in software development. Should programmers be able to define their own conventions or should they conform to the system conventions even where they are not optimal? Do new programmers get to define their own conventions just because they were not involved in the original decision? Aren't programmers expected to abide by the conventions so the code will integrate better and others can maintain it later. Of course, as problems surface, it is foolish to resist all change in conventions just because changes have repercussions. Changes are made, but as part of a deliberative process in which the burden of proof is on those who advocate the changes.

The evolution of the law works the same way: computer law is just another subsystem to be integrated into the fabric of jurisprudence.

The problem of having different conventions in different areas is demonstrated by the Cadtrak patent. It is a hardware, rather than a software, patent. It requires a display device but no software to infringe it. A computer can be designed so it does not infringe, although a simple program running on it can. This demonstrates that simple software programs can infringe almost pure hardware patents and suggests the difficulty of drawing a legal distinction between hardware and software.

Pamela Samuelson laments that "patent lawyers [do not] claim software-related inventions in a straightforward manner" [39]. Patent lawyers are faced with a Catch 22 situation because of the line be-

The Academic Debate: Considered Opinion and Advocacy

The Mainstream View

Donald Chisum has written the standard reference on patent law [9] and is frequently quoted in judicial decisions. His expertise is in integrating patent decisions into a coherent view of the patent law across mechanical, electrical, chemical and other technologies as they are handed down. His reputation rests on the soundness of his analysis in predicting how courts will rule.

Chisum views software as one of many technologies and says software is almost as patentable as anything else and to the degree that it is not, it should be [8]. He has two concerns. First, the current decisions uphold most software as patentable in a way that forces patent lawyers and the technical community to focus on legal technicalities rather than the technical ones. Second, he questions whether lack of patentability will create underinvestment in software innovation as compared to other technologies.

His approach is similar to Judge Schwarzer's in determining how to calibrate the credibility of possible "junk science" in a courtroom. The question is not "Does this make sense in isolation?" but "How does it fit into an organized body of knowledge?" [44]

The Contrarian's View

The League provides no citations for its position in "Against Software Patents." But a similar article by the same authors [15] has four citations: Mitchell Kapor's congressional testimony and articles by lawyers Pamela Samuelson [39, 41] and Brian Kahin [26] let us see how the antisoftware patent position fits in with "an organized body of knowledge."

While patent lawyers conversant with software are in virtual unanimity that software should be patentable, neither of these lawyers is a patent lawyer.

As background we should consider the legal education most lawyers get. First, very few study patent law and thus are not exposed to the fundamental concept that patent rights are property rights. Second, most law students do take antitrust law where they learn that monopolies are illegal. Later, when they discover patents they often mistakenly see monopolies. Third, law students are taught to be advocates. Their job is not to make a considered opinion, but to present their case as best they can; someone else—the judge, Congress, the public—makes decisions.

This author's opinion is that those who took an antipatent point of view did it as a knee jerk intuitive reaction. Away from the mainline of software business and patent law, distrustful of monopolies and inexperienced with intellectual property, these lawyers and software developers found support in one another. This view gained respectability as a contrarian's view and is always welcome in legal journals and at conferences.

Samuelson argues that the basis of software patentability is weak and says the primary issue is a policy one [39]. She presents two arguments against software patentability and Kahin a third:

A. The industry has flourished and innovated without patents.

A more accurate statement would be: "The industry has flourished and innovated *with little realization in the software community* that patent protection was available."

The growth of the software industry was not due to differences between software and other technologies, but to its synergetic relationship with the computer industry and the new business opportunities created by the computer's rapidly decreasing costs. Until recently software has been seen by computer companies as a loss leader. Software created the demand for computers which was, and still is, the dominant industry: but software developers were to be kept fat, dumb and happy: salaries were high, patents were not mentioned, and there was lots of technology to play with.

Accidental Empires shows that the personal computer software successes were achieved by amateurs who were lucky enough to be in the right place at the right time [12]. The successful early PC software companies (a) marketed innovations *pioneered by others* and (b) aggressively pursued their own intellectual property rights: Microsoft's MS/DOS is a derivative of CP/M; and Lotus' 1-2-3 of VisiCalc. Had Digital Research, VisiCorp and Software Arts asserted intellectual property rights as aggressively as Microsoft and Lotus, they might not have been eclipsed by them.

In the early stage of the industrial life cycle, the first person in his garage who acts on the opportunity starts with the biggest and most established company in the business. When there is no established competition, new companies can compete without patents. As the industry matures it becomes difficult for new companies to enter the market without a sustainable advantage such as patents.

The free enterprise system rewarded entrepreneurs whose personal computer software companies were successful, as it should, but that success should not blind us to its nature—bringing the innovations of others to market. We expect that if patents had been more widely used by the software industry, the true innovators would have received a fairer share of the rewards, thus rewarding innovation as well as business savvy. Had patents been more widespread, the software industry would have been more profitable, it would have grown with less of a boom-and-bust cycle, albeit less rapidly and there would have been a greater diversity of software product categories and features.

To base a software intellectual property system just on the experience of the last 10 years would be like raising teenagers in the expectation that their childhood will be repeated?

B. Many in the software industry say software patents will discourage innovation.

Samuelson says,

> . . . *it is primarily from the widespread concerns about the effects of patents from within the industry and the technical community that she has pursued this study questioning the patent protection for computer program-related inventions. [39]*

Samuelson addresses this perception by conducting a survey to see how widespread the perception is, at least in the related area of user interface copyrights [42], rather than attempting to determine if the perception reflects reality. In reporting the perception she gives it more credence, reinforcing any error, and creating more concern. It would seem to be more constructive to research the effect of patents on innovation and business formation in other industries to see how it might apply to software, and analyze software patents that exist for their effect on innovation and new business formation as we do.

It makes as much sense to devise theories of innovation by polling the software community as it does to devise laws of physics by polling people who walk. The intuitive answer is not necessarily the correct one. Ask people, "Assume you are walking at a steady pace holding a ball, and you drop the ball. Will the ball hit the ground in front of you, in back of you, or next to you?" most people will say "in back of me" [32]. A physicist will tell you that Newton's laws predict, "next to me," and can perform an experiment to prove it.

In contrast to Chisum and the mainstream of patent law where software is viewed from a broad perspective of many technologies, Samuelson views the law from inside the software community looking out. She has always advocated narrow protection—arguing as late as 1984 that CONTU's recommendation that software in machine readable form be copyrightable was ill considered [40].

Samuelson acknowledges that the conditions that promoted software innovation until now may be different from those that will promote it in the future.

Samuelson proposes to design a special *(sui generis)* form of protection for software, saying, "It is possible to design a law that is appropriate to the kind of subject matter that software is." How can one be certain of one's ability to build a skyscraper, if one is uncertain if one is building on bedrock or marshland?

Writing laws is like designing systems. They will have desired and undesired, intended and unintended consequences. The current law, especially *Benson* forces patent lawyers into a Catch 22 dilemma: Write a straightforward patent, and get it rejected as a mathematical algorithm; write one that is patentable subject matter and it will not be straightforward. Attempting to make only software unpatentable will no more prevent practical software patents from issuing and being enforced than prohibition will eliminate alcoholism. Samuelson laments that "patent lawyers [do not] claim software-related inventions in a straightforward manner [39]; but this is like blaming a program's users or the market for not behaving as expected—a sure sign of an inexperienced designer. Law, software, and marketing strategies must be designed the same way the Constitution was: based not on how we would like people to behave, but on how self interested people will actually behave.

Academics such as Samuelson, who pontificate on software patents and want to create a *sui generis* system of protection, seem ready to reinvent the wheel before they understand how a wagon works or the infrastructure of the highway system. Most show no knowledge or understanding of the processes of innovation over hundreds of years in a variety of technologies. Most have little, if any, experience prosecuting (filing), analyzing, or litigating patents. They argue that software is different from other technologies, not from the perspective of the history of innovation and patents over a range of technologies, but from the myopic view of the development of a single technology, largely in a single software marketplace in an atypical decade—the personal computer market in the 1980s.

Ben Franklin described a professor who was so learned he knew the word for horse in five languages. *equus* in Latin, *caballo* in Spanish, *cheval* in French, *cavallo* in Italian, and *Pferd,* in German. He then went out to purchase one, and returned with a cow. Given your experience observing complex software systems develop and how the hype manifests itself in reality, when you hear *sui generis* protection systems being proposed as transport into the twenty-first century, ask yourself: "Will I ride horse or a cow?"

Taking a contrary position and fighting for it, as Samuelson does, is in the highest tradition of the law. In part it is because truth emerges from the debate and if there is no one to debate with, a poor sort of truth will emerge. Yesterday's contrarian view may emerge as the mainstream view. It is like having advocates for a programming language like Forth. It forces identification and discussion of issues and influences the industry—PostScript is Forth-like. But software developers seriously consider programing in Forth only in unusual circumstances.

C. Software helps disseminate information

Brian Kahin, a research fellow in the Science, Technology and Public Policy Program at Harvard University's Kennedy School of Government offers a different reason software should not be patentable [25]:

A deeper, more disturbing problem in patenting programs was barely evident before computers became ubiquitous personal tools . . . the computer has developed into a medium for human expression and a mediator of human experience. Thus, what is increasingly at stake in software patents is the generation and flow of information.

The "barely evident" "problem" was addressed by Lincoln who, in his *Lecture on Discoveries* [23], said:

certain inventions and discoveries occurred, of particular value on account of their efficiency in facilitating all other inventions and discoveries. Of these were the arts of writing and of printing—the discovery of America, and the introduction of Patent-laws

Lincoln not only believed that the patent laws encouraged innovation, but he anticipated Kahin in realizing that inventions which promote dissemination of information are particularly important. Since Lincoln's speech, patents seem to have encouraged many inventions which engender the "generation and flow of information": the telephone (Bell), phonograph (Edison), movie camera (Edison), Xerography (Carlson), radio (Armstrong and Marconi), Phototypesetting (Scheffer), and TV (Philo T. Farnsworth).

Even the birth of patenting, Aldus Manutius, the famous Venetian scholar printer for whom the desktop publishing company is named, received two patents—one on a form of Greek type—and a ten year monopoly to use the italic font which he invented.

Having discovered that he is treading in the footsteps of Lincoln, albeit in the opposite direction, Kahin might, in planning his future travel, consult the work of his colleagues at the Kennedy School of Government, which has guided our analysis [33].

tween what is patentable and what is not: Write a straightforward patent, and get it rejected as pure software because of *Benson;* write one that is patentable subject matter and it will not be straightforward. Attempting to make software unpatentable will no more prevent practical software patents from issuing and being enforced than prohibition will eliminate alcoholism.

The PTO and those patent lawyers who prosecute software patents have much more experience in the nitty-gritty of protecting software than academics. And the PTO and the courts have more experience weaving new technologies into the fabric of the patent system than the software community has in creating forms of intellectual property protection.

The debate in the academic world is described in a sidebar. The mainstream view taken by the practicing and academic patent bar and most computer lawyers on one side and the contrarians led by Samuelson, argue against software having the same breadth of protection as other technologies. The League for Programming Freedom has launched an offense in the debate. Like the Battle of the Bulge, it might appear formidable when seen up close. But while it must be treated seriously, it is the last gasp of a dying cause.

The pioneers in each new technology see that technology as new, different, and central, and expect the world to accommodate it. To some extent the world does. But each new technology slowly becomes woven into the tapestry of knowledge encompassing other technologies—each distinctive in its picture—but using the same threads and the same weave.

5. A nonprofit Marxist economic system is not optimal in promoting innovation in software.

This is the paradox one must confront if one argues software patents decrease innovation. The essential difference between Marxism and capitalism is property rights. Patents and copyrights create intellectual property rights that can be brought, sold, rented and licensed like other property rights. Marxism may be better than capitalism in some areas—certainly not in Russia, but capitalism, with all its flaws, has outperformed Marxism.

The paradox of Marxism is not just a theoretical issue. Stallman, the founder of the *League for Programming Freedom,* heads the Free Software Foundation which is developing and planning to distribute a clone of the Unix™ operating system. AT&T has invested in Unix based on its ownership as manifest in patents and copyrights. AT&T cannot be pleased when Stallman gives away free copies of a clone of a product it invested millions in developing and marketing.

If AT&T had not used patents and user interface copyrights to protect its intellectual property rights, Stallman would have no trouble making and distributing a Unix clone. But AT&T must pay its bills with money it receives from customers and has asserted its rights. If it is acceptable to clone Unix or any program, will anyone invest in new ideas? Should we optimize an intellectual property jurisprudence for, not large entities, not small entities, but companies that distribute free clones of other people's software?

For all his talk about wanting to promote innovation, Stallman seems to get his ideas for technology from AT&T, 1969 and his ideas for intellectual property protection from IBM, 1965.

Many software developers do their work for the fun of it. But the distinction is based, not on the technology, but on amateurism: amateurs flourish in the early stages of a new technology. Professionals have accepted that others work for free, but they bristle if they are expected to work at the same rates.

6. Software, like every technology, has unique problems.

Software patents have unique problems: prior art libraries are limited, the search classification system was designed for hardware patents, few computer scientists are examiners. Still when it gets to specific cases, computer scientists and the PTO see invention similarly. (See Document Comparison)

For the last two years the PTO has been improving the situation. It is improving its prior art search facilities in software, has published a new software classification system, and is actively recruiting computer scientists.

The PTO has still not been able to rid itself of the prejudice against software patents, as patent practitioners in the software area will tell you. It still is conservative in its interpretation of what constitutes patentable subject matter, and has rejected several applications that are being appealed.

7. Legally, software is patentable and it will remain so.

Prior to 1982, about 30 different software-related patent cases went through the Appellate Courts. The range of technologies—seismic, medical, petrochemical, telecommunications, firmware, and software—demonstrate that software is both well grounded in patent law, and basic to the advancement of American Industry. Software has become pervasive in industry, which has been basing business decisions on software's being patentable for 10 to 20 years. This has created a sophisticated broad-based constituency for keeping software patentable. Congress has not given in to demands to make less pervasive technologies, such as biotechnology, unpatentable; it is less likely to do so with software. Software has been clearly patentable longer than it has been copyrightable (See Patents and Copyrights).

Chisum, the leading authority on patents, wrote an article on the patentability of software and concluded:

The continuing confusion over the patentability of computer programming ideas can be laid on the doorsteps of a single Supreme Court decision, Gottschalk vs. Benson, *which held that mathematical algorithms cannot be patented, no matter how new and useful. A* *careful analysis of that decision shows the holding is not supported by any of the authorities on which it relied, that the Court misunderstood the nature of the subject matter before it, and that the Court failed to offer any viable policy justification for excluding by judicial fiat mathematical algorithms from the patent system. The* Benson *decision is inconsistent with the later Supreme Court ruling in* Diamond vs. Chakabarty *that the patent system applies impartially to new technologies and that any policy issues for excluding new technologies should be addressed to Congress. Policy considerations indicate that patent protection is appropriate for mathematical algorithms that are useful for computer programming as for other technological innovations* [8].

Chisum is in the mainstream in saying that the courts made a mistake by making software unpatentable. But courts are reluctant to overturn previous decisions directly, and then only after their scope has been eroded. A similar situation where prejudice had been become part of the jurisprudence occurred earlier: *Plessy vs. Furgesson* [1896] held that "separate but equal" facilities for whites and blacks were lawful. The courts did not directly overturn it, but eroded its vitality on a case-by-case basis over a period of years in a number of decisions starting with *Murrey vs. Maryland* while appearing to show respect for *Plessy.* Finally, when faced with *Brown vs. Board of Education* [1954], ample precedent had been created for the Supreme Court to overrule *Plessy* directly.

Samuelson, having asked Chisum to write his article, now attempted to refute him, but after arguing for over 100 pages that the basis for software patentability is weak, was forced to conclude that:

. . . the only principle which seems to have guided the court's decisions is one of upholding the patentability of as many program-related inventions as possible while appearing to show respect for the Supreme Court's decisions. [39]

Samuelson's observation seems to be compelling evidence that while she has not been persuaded that *Benson,* like *Plessy,* is a fundamental error in giving prejudice the force of law, the Court has and will, in due course, reverse it and the original intent of Congress will again become law and statutory. Subject matter will "include anything under the sun that is made by man."

8. Whether or not one agrees that software patents are beneficial, patents are here to stay so we should plan to work with them.

The software community will be best served by articles about how to avoid infringement, how to deal with infringement notices, how to find prior art, how to use patents to protect new ideas, how to differentiate products, and how to make the patent system work better for software (based on experience rather than speculation). In brief, we should direct our energies toward making the system work in order to increase innovation and U.S. competitiveness, rather than fighting patents.

9. The practical effect of continuing to spread misinformation on software patents will be to hurt small developers and U.S. competitiveness in software.

Patents, like a cat's claws, function as weapons when necessary. A declawed cat will not survive in the wild; neither can a defenseless startup once it succeeds and attracts substantial competitors. Patents are not the only defense, but they are vital to innovative startups that must survive. In business, as in the jungle, respect is given only to those who can protect themselves.

Microsoft, IBM and others are applying for patents in quantity. Those who do not understand the situation are not. Many are happy to have software patents attacked. Why let your competitor in on a good thing? Why pioneer new

Document Comparison: An Obvious Patent?

A criticism of the patent system is that computer scientists are qualified to judge invention in software while the PTO is not. In his article [26] Kahin says, the PTO is,

awarding patents merely for automating familiar processes such as . . . comparing documents (Patent No. 4,807,182). But software developers have been routinely automating such [functions] for years.

In fact, the ACM published a refereed paper describing that (hashcoding) technique for comparing two text files albeit for source code, rather than document comparison [17]. It seems that the ACM and the PTO have similar standards of inventiveness.

It so happens I wrote that paper, and I brought it to the attention of the patentholder and (indirectly) to WordPerfect about 10 months before Kahin's article was published. Four companies put on notice about the patent brought the paper to the attention of the patentholder. This suggests that where prior art exists which narrows a patent's scope, it is likely to surface.

Advanced Software, was founded to develop and market DocuComp which uses the patented technology. Its inventor, Cary Queen, a Ph.D. in mathematics has filed over a dozen patents in genetic engineering where he is principle in a startup, Protein Design Labs. He also used the patented hashcoding technique to compare genes to identify similarities.

Cary Queen reports there is more prejudice against patents in software than in biotechnology, where hundreds of startups have been financed, as biotechnology patents are better respected.

product ideas when it is less risky to copy competing products and incorporate useful features once market success is proven.

From the perspective of large companies, a loud voice, such as the League, yelling against software patents can be useful as a means to destroy one's competition.

The Japanese are aggressively filing for U.S. patents on software. While our strength is innovation, Japan's is in adapting innovations and steady improvement. But if they have the improvement patents and we did not file for the basic patents, we lose. If we arrogantly dismiss the Japanese as incapable of creating good software or cavalierly dismiss patents as undesirable, then 20 years from now we will be trying to get back the software market from Japan just as today we are trying to get back the automobile and semiconductor markets. We are not even trying to get back the consumer electronics market.

Who is responsible for the misperception about the desirability and legality of software patents? In a certain sense, it is the League for Programming Freedom. But it knows not what it does. And its arguments are the ghosts of arguments for IBM's corporate self-interest of a bygone era. Is not the origin of the problem IBM's attempt in the 1960s to declaw a competing technology by depriving its practitioners of their constitutional rights as inventors? (See Software Patents: IBM's Role in History.)

If it were just a question of IBM outfoxing its competitors, we might learn our lesson and let it pass. But we think it useful to ask some questions: Is it in the interest of the United States to have strong, competitive, innovative software industry? Is it in IBM's interest? Did IBM use its position on the 1966 Patent Commission to put its corporate self-interest ahead of that of the U.S.? Should IBM be held responsible for its role in creating the current software patent mess? Some have proposed making software patents unenforceable. Might a law making IBM patents unenforceable make more sense? Or a law that would prevent IBM from obtaining patents for a period of time, say 5 or 10 years? At a time when competitiveness with Japan is a major concern, what kind of a message should we send about what happens to those who use their positions on government commissions to sacrifice their country's interest to their corporate self-interest?

Similarly, should we eliminate patents to avoid patent litigation as the League suggests; should we not eliminate all laws so as to avoid all litigation?

10. In considering the issues, we should deal with examples of real patents and, where possible, real infringement where facts for both sides are fairly stated.

If we are to have a meaningful debate on whether software should be patentable, I suggest we take our standards, both of debate and of where the burden of proof lies from Abraham Lincoln:

I do not mean to say we are bound to follow implicitly in whatever our fathers did. To do so would be to discard all the lights of current experience—to reject all progress—all improvement . . . if we would supplant the opinions and policy of our fathers in any case, we should do so upon evidence so conclusive, and argument so clear, that even their great authority, fairly considered and weighed, cannot stand . . .

If any man [believes something], he is right to say so, and to enforce his position by all truthful evidence and fair argument which he can. But he has no right to mislead others, who have less

Software Patents: IBM's Role in History

In the late 1960s when IBM's internal policy was that software should not be patentable, IBM vice president, J. W. Birkenstock, chaired a presidential commission on the patent system which recommended that software should not be patentable. We expect that the other commission members deferred to IBM's expertise on software, just as members of a commission designing an aviary would defer to its most knowledgeable member on birds: the cat.

Congress rejected this view, but three paragraphs of the Commission's recommendations (e.g., IBM's corporate policy) found their way into *Gottschalk vs. Benson*, the Supreme Court Decision that limited the patentability of software. At this time IBM had 70% of the computer market, so it is not surprising that CBEMA, the Computer Business Equipment Manufacturers Association filed an *amicus curie* brief against software patents in *Benson*.

From this historical perspective we can see that the conventional wisdom that "software has not been patentable," should be more accurately stated as "it was not in the interest of IBM or other computer manufacturers for people to think software is patentable." We have never seen it pointed out in the debate on software patents that the idea that software is not patentable subject matter was formed in the crucible of IBM's self-interest and corporate policies of an earlier time.

IBM and CBEMA have now rejected the Stallman-primal IBM view [7]. But the damage has been done. The PTO and the industry have not taken software patents seriously until recently, which explains the problems the PTO has had in examining patents and the prejudice against software inventors who assert their patent rights. Many in the software community have been suckered into believing software should not be patentable, while IBM has aggressively but quietly been getting software patents and become the company with the largest software sales.

access to history, and less leisure to study it, into [a] false belief . . . thus substituting falsehood and deception for truthful evidence and fair argument.

What I find most frustrating in this debate is that the mode of argument used against software patents by so many [15, 26, 27, 28] is to throw as much mud against the wall as possible and hope some of it will stick. I have expended some effort here removing some of the mud. I do not claim to have removed it all, but I hope I have wiped away enough to show you the rest will wash off too.

A Study of Nine Software Patents

In its article the League lists nine patents—mine and eight others to make its case. It is unlikely that the members of the League considered the positive side of any of the patents they cited. It is as if they went searching for quarters with heads showing, and finding several, reported their findings without turning any of them over. Here we turn over the other eight quarters in an attempt to produce some empirical results.

U.S. Patent 4,197,590: The inventor founded a company to develop and market what appears to be the first personal computer to write directly from memory to the display. This invention has been widely licensed to the personal computer industry by Cadtrak. The "XOR" is only part of the invention. Cadtrak filed and has won at least one lawsuit against a larger company. The idea behind the "XOR" claims of this patent is simple. A program XORs a cursor icon onto a display device; later a second XOR to the same place erases the cursor, restoring the original display. To move the cursor one XORs the cursor to its old location, then XORs it onto the new location. There are many ways to get around this patent. One can use an underline as a cursor or "logically or" the cursor onto the display, erasing later by rewriting the display with its original information. This approach is fast, lets you change cursor icons easily, accesses the minimum possible data, and requires no space be reserved on the screen for the cursor.

The League says this patent can be infringed in "a few lines of a program." It can be, *but not on a computer that was commercially available at the time the invention was made.* The invention is largely the invention of the frame buffer. As such, it requires hardware which has since become common, making it possible to infringe the XOR claims with a few lines of code. Many, if not most, computer manufacturers including Apple and IBM have taken out licenses which cover programs running on their computers.

This patent illustrates that it is usually easy to design around a patent one accidentally infringes. If this patent, a hardware patent, is a "bad" patent as some claim, it only demonstrates that the electronics industry tolerates "bad" patents because it finds patents beneficial on balance. Software should be able to tolerate "bad" patents similarly. To discard the patent system because some bad patents exist would be the same as suppressing free speech to stamp out lies.

U.S. Patent 4,398,249: This is what the League has mischaracterized as fact, in 'Refrac recalculation patent'. In 1970 Rene Pardo and Remy Landau invented the concept of an array of formulas that would enable businesspeople to write their own programs to create business applications. Although the word was not used, their invention is in essence the modern computer spreadsheet. The fact that the claims cover recalculation is an artifact of how the patent claims were written. Pardo and Landau marketed a commercial spreadsheet-like product based on this technology. This invention has been widely adapted in the personal computer industry—over 250 spreadsheets have been marketed.

This patent was originally rejected by the PTO as a mathematical algorithm and thus unpatentable subject matter. Pardo and Landau felt so strongly about their inventive contribution that they appealed their case *pro se,* which means they, not a lawyer, wrote the brief and argued it before the appeals court. The decision, *In re Pardo* [23] is a major legal precedent which establishes that an invention is patentable whether or not the invention involves software "novelty." If their experience was typical, they were stonewalled when they tried to enforce their patents. This would explain why they approached Refac—a white knight in the fight against the patent pirates. If Pardo and Landau have the same deal Refac offered others, then they can expect to collect royalties only as Refac does.

U.S. Patent 4,633,416: This patent

Patents & Copyrights

Kahin says, "Never before has an industry in which copyright was widely established suddenly been subjected to patenting" [25].

In fact, patent protection for software was established before copyright protection. *Diamond vs. Diehr* (1981) preceded the two most important software copyright cases, *Apple Computer vs. Franklin Computer Company* (1982) and *SAS Institute Inc. vs. S&H Computer Systems Inc* (1985). Even today the case law on user interface copyrights is sparse. Software, or "computer-related" patents were obtained in the 1960s. Martin Goetz of Applied Data Research received U.S. patents 3,380,029 in 1968 on a Sorting System, and 3,533,086 in 1970 on AutoFlow, an automatic flow charting program. (When IBM started giving Roscoe, a flowcharting program, away *free,* Applied Data Research sued for antitrust and, in 1969, settled for about $2 million in damages.) The recalculation patent filed in 1970 was granted in 1983.

is held by Quantel, a company that developed a line of commercial video editing products protected by its patents. Quantel filed for patents when it was small, and it has grown from being a small to a large company because it has used its patents to prevent competitors from using its technology. The first company Quantel sued was much larger than it was.

U.S. Patent 4,777,596: The League tells us XyQuest was notified that its product XyWrite infringed Productivity Software patent, protecting the ability to accept an abbreviation or correct a spelling error by hitting a space bar. When licensing negotiations failed, XyQuest removed the feature from future releases.

Productivity Software was founded in 1984 to develop data input systems where minimal keystroke data input is important. Based on its patented technology, Productivity Software has grown to seven employees and markets 31 specialty products. It has found niches in the medical and legal transcription and the handicapped marketplaces.

Patent problems are generally minor compared to the other problems pioneering companies face. At about the same time XyQuest had postponed introduction of its latest

How Patents Work

Exclusive or territorial rights bestow on their owners a long-term outlook, and create a simple test for determining whether or not to fight. This leads to stable solutions and minimizes inefficient disputes. Such rights occur in many areas. A miner stakes a claim; salespeople have exclusive territories, and professors specialize in areas and are given tenure. Such territorial rights stimulate diversity by encouraging competitors to stake their territorial claims at a distance. Lee De Forest, for example, invented the triode, the amplifying vacuum tube, to avoid infringing Fessenden's spade detector patent [29].

Many mistakenly believe that the patent system protects only "flash of genius" insights. That is not true. In 1952, Congress overrode the "flash of genius" doctrine. Patents are designed, not to stimulate invention directly, but to stimulate their commercialization by giving exclusive rights for 17 years to anyone who invents something new and not obvious. Just as an author need meet a standard of creativity to get a copyright on an original work, a patentholder need meet a standard of nonobviousness to get a patent on something new.

The following discussion of patents is useful in providing an overall understanding of how patent infringement is determined, especially where an overly broad (e.g., "bad") patent may be involved.

Before the PTO will allow your patent application, it does a search (rather like a title search when you buy a house) to find prior art on your invention—what others have done earlier that is disclosed in publications or products. The PTO examines the prior art it finds along with any you send it. If what you did is sufficiently different, it issues claims that delimit the territory of your invention.

The process is rather like finding new territory. Suppose you suddenly landed in Left Fork, North Dakota and found that no one lived there and wanted to claim it as yours. You might try to claim all the land west of the Mississippi. The PTO will likely find that people have lived in nearby states and may issue you a claim to say, North Dakota. Of course, the PTO could allow your claim in error. The too broad claim—all the land west of the Mississippi—will look impressive and could be useful as a source of cash from people impressed by surface rather than substance.

In practice, if you try to enforce the patent against, say, Californians who just discovered gold, they would show the court that people lived in California before you landed in Left Fork. The court will declare your broad claim invalid. Practically speaking, you would not go to court once you realized that people lived in California earlier. You might go back to the PTO to get the patent reissued, showing them the prior art and claiming a smaller territory. The PTO might only allow narrower claims that cover eastern North Dakota, or maybe only Left Fork, North Dakota.

A patent does not necessarily give you rights to what it says it does. Undiscovered prior art might considerably narrow its scope. The advantages of being issued a too broad patent are (a) potential infringers might keep a greater distance than they have to, and (b) you can wait to define the limits of your territory until you know the terrain better. The disadvantage is that you might make business decisions based on your belief that you had rights you did not possess.

There are two ways you can respect a patent: you can avoid infringement or you can take out a license. If you are infringing, the patentholder will usually forgive past infringement if you agree to remove infringing capability. Your show of respect for the patent gives its holder credibility with other infringers.

As a possible infringer you have several courses of action when you face an overly broad (or "bad") patent, or indeed any patent:

1. Ignore the patent problem until confronted with it.

version for about a year so it could upgrade to IBM standards as part of an agreement in which IBM would market XyWrite exclusively. At the last minute IBM reneged on the deal [25].

While the first five patents are held by small entities, the last four patents are held by large entities and they also protected commercial products:

U.S. Patent 4,558,302: UniSys licenses this, the LZW compression patent, for 1% of sales. It has threatened a large entity with a lawsuit, but no small ones.

U.S. Patent 4,555,775: The League describes AT&T's backup store patent as "Too Obvious to Publish." Yet, in a letter in this issue of *Communications*, Dennis Richie points out that this technology was published in the ACM [36] and was recently called "a seminal paper" whose ideas are seen in X Windows Macintosh and many other windows systems [14]. While AT&T has sent notification letters on this patent, it has put the patent into reexamination and has not threatened suit or sued anyone on this patent.

U.S. Patent 4,656,583: This is an IBM patent on compiler speedup.

U.S. Patent 4,742,450: This is an IBM-shared copy on write patent.

Why look for trouble you might never have to face? When and if a patent is brought to your attention you can decide what to do. If you do a search and find a patent, you might spend effort designing around a patent that its owner would never have asserted against you. If you do not design around it, you might be liable for treble damages because you were aware of the patent. Of course if you are competing against products protected by patents, you might want to check into their patents before you design your product, as you can expect your competitor to examine your product for infringement, thus you will probably have to face the problem one way or another. Here, it is good accounting practice to set aside a reserve for infringement.

2. Stay outside the claimed territory.

If the patent claims all the land west of the Mississippi and you stay on the east of the Mississippi, you will not infringe.

3. Go where people were.

If you know people were in Bismark before the patentholder landed in Left Fork, settling in Bismark will protect you. Distrust rumors about earlier settlers. Make sure the prior art is documented in a published paper or was obviously used in a product. If you ask around the industry you are likely to find pointers to prior art. You might want to send the prior art to the patent owner, or the PTO for insertion in the file wrapper. The file wrapper is a file containing all the correspondence on the patent with the PTO. Your patent lawyer might consult it to find prior art which might help you design around a patent or understand its scope.

4. Make a business deal with the patentholder.

Generally you can license, or cross-license a patent or find some other way to get rights.

5. Break the patent.

You can attempt to get the patent invalidated by proving it is invalid over the prior art, the disclosure was inadequate or it was otherwise invalid. This is risky and expensive where the patent is good and the patentholder determined.

A patent must claim something new, lest its owner usurp others' rights; it also must be on something nonobvious to prevent giving protection to insignificant improvements.

As technical people, we often look at a patent differently from the way entrepreneurs and judges do. We see Left Fork, North Dakota after it has become a thriving town, and are likely to say it is obvious—there are lots of places like Left Fork, and lots of them have similar buildings; thus constructing buildings in Left Fork seems obvious. It does not belong to the entrepreneur. Invite everyone!

The Left Fork patentholder, the entrepreneur, feels this is like arguing that it was obvious that land in Silicon Valley or Microsoft stock would appreciate in value. Given the advantage of hindsight, it is obvious, but the person who invested in the land, the stock, or the technology should benefit from its appreciation in value. The entrepreneur says, "I built the buildings based on my being granted rights to them and my having a vision of what I could make of it. And now you are looking for loopholes in my deed so others can move in! It may be a poor thing, but it is my own."

As technical people our immediate bias is to find inventions "obvious," because we focus on the technical sophistication, evaluated with the advantage of hindsight. The value of land, a patent or a copyright has to do with how the market evaluates it. If the land is in the Mohave desert; the copyrighted work banal, or the patent on technology people don't want, then it may be worthless. If the land is in downtown Manhattan, the copyright on Donald Duck, or the patent on technology others want, it can be very valuable. The important thing about an invention is not so much that it be inventive, but that it be new if it is to be patented, and that it be useful if people are to buy it.

These two patents are what IBM calls Group 1 patents whose royalty is 1% of sales. They have been licensed by IBM as part of general licensing agreements but have not been licensed individually. (About 50 of IBM patents are Group 2 patents. Group 2 patents can be licensed for 2% each; the entire Group 1 portfolio, for 2%; and the entire IBM patent portfolio for 5%.)

These two patents have not been litigated and I do not believe IBM has aggressively asserted these patents against anyone. IBM, like most companies, normally files for patents only to protect what they expect to become commercial products. We treat these patents as protecting commercial products.

Most patents are never asserted. Much of the value of patents, like that of the Swiss Army, is that they act as a deterrent. The patents described here are typical of the small number where the patentholder forces a resolution: the infringer may take a license, design around the patent, or produce prior art showing there was no infringement.

Many letters asserting patents are no trespassing signs, putting potential infringers on notice should they infringe or telling them not to. They require no action. The notified companies might send prior art back to the patentholder, who might send it to the PTO for reexamination. The "infringer" may ignore the notice, waiting to see the reexamined patent or for the patentholder to become more assertive. The resolution may be hidden, in that an infringer may design around the patent. Rarely, a product is withdrawn from the market. The statistics on the cited patents are summarized in Table 1.

Whether any patent including those described here, is valid and infringed is a complex legal and technical question. An advantage of the patent system is that the question is an objective one based on the patent, prior art, and the "infringing" device. Such a dispute is less acrimonious than one in which the task is to evaluate testimony where one person yells "thief," and other "liar." Whether infringement actually occurred in any of the cases is irrelevant. The relevant question is did the original patentholders bring commercial products to market based on the patented technology and motivated by the rights a patent bestows?

Analysis Results

The nine patents cited by the League summarized in Table 1 lead us to these conclusions:

1. Software patents stimulate companies to bring commercial products to market.

All nine patents protected commercial products.

2. Software patents stimulate new business formation.

Four of the nine patents were from startups founded to exploit the patented technology. A fifth filed for

Trademarks: Apple paid $30 million to use the name "Apple"

Trademark law is like patent law in that the first one who claims it gets to own it. The Beatles recorded on their own label, Apple Records. When Apple Computer was founded it agreed not to use the *Apple* name in the music business. Later, when the Macintosh played music, the Beatles sued. Apple Computer settled, paying about $30 million dollars to use the name "Apple."

I have been publicly accused of extorting Apple. Did I extort Apple? Did the Beatles extort Apple? Should the computer business have its own *sui generis* trademark law?

Table 1.

Company Size	Large		Small		Total	
Patent Activity						
Patents granted	4	100%	5	100%	9	100%
Protected commercial products	4	100%	5	100%	9	100%
License appears to be available	4	100%	4	80%	8	89%
Firm founded to develop technology	0	0%	4	80%	4	44%
Sued Large Entities	0	0%	4	80%	4	44%
Sued Small Entities	0	0%	2	50%	2	22%
First Suit against Small Entity	0	0%	0	0%	0	0%
Suits threatened	1	25%	5	100%	5	67%
Patent Asserted (Notice sent)	2	50%	5	100%	7	78%
Resolution (Patent Asserted)						
Infringement Removed	0	0%	1	20%	1	11%
Product removed from market	0	0%	1	20%	1	11%
Licenses	1	25%	2	40%	3	33%
Unresolved	1	25%	1	20%	2	22%
Nothing to Resolve (No notice)	2	50%	0	0%	2	22%
Total	4	100%	5	100%	9	100%

Patents cited in Against Software Patents. We treat multiple patents covering the same technology as a single patent. With the exception of Quantel, the small entities were less than a dozen people at the time the patent was filed, and the large entities were Fortune 1000 companies. We assume that if a lawsuit was threatened, a notice was sent; and if a lawsuit was filed, a lawsuit was first threatened. We assume that if a patent has been asserted—people have been sent notices—there is a matter to be resolved. Even if one characterizes Cadtrak and Refac as being in the business of litigating patents as some do, the relevant fact is that the original patentholders were small entities introducing commercial products protected by patents.

its patent in its seventh year. All five companies struggled for years.

3. Software patents stimulate the commercial introduction of fundamental advances by small entities.
The technology pioneered by at least three of the small patentholders was significant in that it started new product categories or were widely adopted in the industry.

4. Licenses are usually available where companies enforce patents.
Only Quantel seems to be unwilling to license its patent.

5. Where similar-size companies had a dispute, they settled differences quickly without litigation.
The only patent dispute between similar-size companies (XyQuest) was settled readily. No small entities were faced with a lawsuit brought by a large entity without the advantage of the patentholder having settled earlier with a large infringer.

6. Small entities incurred little if any royalty and litigation costs for infringing patents.
The only disputes in which a small entity paid patent royalties or was sued were those in which the patentholder had previously settled disputes with larger companies. No case was cited in which a big company aggressively went after a small one over patents, unless a large company had respected the patents first. The only such instance the author knows of is IBM (see sidebar).

7. Patent piracy by large entities appears to be common and small entities have a tough time getting their rights respected.
Four of the five small entities had large entities use their technology without first licensing it. *All four were forced to sue.* This makes it unlikely that all the patent disputes were an honest difference of opinion, although some probably were. For this reason "piracy" seems a fair characterization. These same small entities have had their patents mischaracterized and their motives inpugned in the academic, trade and business press read by the software community.

It can cost over a million dollars to litigate a patent through to trial. The data shows that large entities are quick to use their power to try to intimidate small ones into abandoning their rights or accepting nuisance settlements rather than address infringement issues on their merits. It appears that this high rate of patent piracy is caused in part by the *Federal Rules of Civil Procedure* which tilt the scales of justice against the weak.

Our results confirm the League's suggestion that big companies will readily bully small ones, but refutes its suggestion that a patentholder who asserts a patent will get showered with a gold. The yellow matter is not gold.

8. U.S. companies are slow to accept software innovations from outside sources.
The Japanese adapt innovations from sources outside the company twice as fast as U.S. companies [31]. The technology protected by at least two of the patents (CadTrak and HyperRacks) was exposed to companies that later became the first infringers.

Japan's ability to accommodate outside [the firm] innovation may be one of the reasons it has been so

Big Companies Do Sue Small Ones

While the League says that big companies will use patents against small ones, it cites no example. Two came to my attention. (I was contacted because I had arguably relevant prior art on the first patent.) In both, IBM sued former employees to get ownership of patents on technology developed on their own time, unrelated to their work and only *after* the technology proved to have value in the market.

***IBM vs. Goldwasser* Civil 5:91 00021 D. Conn.: IBM encouraged employees to develop software products on their own time and seek patent protection for them so IBM could evaluate them for marketing. Goldwasser developed such a software product, but IBM rejected it; he left IBM stating he intended to pursue his technology, as he did. Six years later, another company introduced a product that seemed to be infringing his patents and he sued them. That company claimed it was covered under its cross-license with IBM; IBM sued Goldwasser to get ownership of the patent.**

***IBM vs. Zachariades* C-91 20419: Zachariades before and while working for IBM developed on his own time a plastic valuable to the medical industry. He kept IBM informed about what he was doing, applied for patents, started his own company, and licensed the technology to a medical prosthesis company. When he was not paid, he sued and a jury awarded him $99 million. IBM "suddenly" found out what was happening and fired, and sued, him for the patents, telling him they did it in part to "terrorize" other IBM employees.**

Companies like to hire litigators who know what it is like from the other side. In both cases IBM is represented by the same firm that represented Edwin Armstrong, the great inventor of modern radio when David Sarnoff and RCA were refusing to respect Armstrong's rights. Ken Burns tells the story in his PBS documentary, *Empire of the Air:* On January 31, 1954 Edwin Amstrong under the strain of RCA's tactics—well dressed as always, in a suit, overcoat, scarf and gloves—jumped from his 13th floor apartment onto the third story roof of the River Club below [29]. His widow won all the patent suits.

Having hired a firm that experienced firsthand the tactics that caused a great inventor to kill himself, IBM should be able to, by suing Goldwasser and Zachariades, "terrorize" its employees.

Table 2.

Innovation Source	Commercial Products	Efficacy (Patents Asserted)	Cost (000,000/yr)	Cost-effectiveness
Large entity	4	2	0.03	67.
Small entity	5	5	0.01	500.
Commercial Sector	9	7	0.04	175.
Federally funded	1	0	$487.0	0.0021

Cost Effectiveness to Taxpayers of Innovation Sources. None of the nine patents appears to result from federal funding. However, we arbitrarily allocate one patent to this category so as to prevent zero results. While the PTO is virtually self-funded, $1.8 million of its $419 million budget comes from taxpayers. Since about 25% of the patents are from small entities and 75% from large entities we distribute the $1.8 million accordingly. We assume that 2% of the patents are software related; it is probably less. Government R & D in computer science was $487 million in fiscal 1989 out of total federally funded research of $61 billion.

successful in dominating markets. If the U.S. is to exploit its strengths in innovation, it must learn to adapt outside innovations without the inefficiency of legal confrontation. Fast and efficient patent enforcement should encourage U.S. companies to license outside technology early rather than wait until they have an infringing product in the market and face legal exposure.

If large companies are forced to deal with infringement issue early, they might see it is to their advantage to work with the inventors, using their knowledge. Now, the legal system keeps the patentholder and infringer at war until such time as the patentholder's knowledge is of little value to the "infringer," thus wasting one of our most valuable resources—the creativity and experience of innovators.

9. Developers do not seem to be infringing multiple patents on a single product.
The only example that was cited in which someone faced infringement issues from more than one patentholder seems to be XWindows facing the CadTrak and AT&T patents, but this has not been resolved and no lawsuits seem to have been filed or threatened.

10. The patent system seems to reject bad patents early in the patent assertion process.
We think the League is right in alleging that bad patents have been issued. The League however, fails to identify a patent that was rejected by the courts. We think this is because issues of prior art and patent invalidity are considered early in the patent assertion process. Patentholders rarely continue to assert patents in the face of solid evidence of invalidity or noninfringement.

11. If software patents were more widely respected we would probably have had fewer variations on a theme, and more themes to vary on.
Product development effort seems to have focused on creating many versions of an invention once its value was proven. Over 250 different spreadsheets and at least four products generally considered to be HyperCard clones were marketed.

12. Big companies' patents do not seem to inhibit small developers.
The innovations protected by small entity patents listed here seem to have been more widely adopted than those of big companies in their industries. Big companies are better at commercializing and protecting their minor innovations, than their major ones.

That small entities seem to introduce the more fundamental innovations to the market is telling. Big companies are often unsuccessful in transforming innovations into commercial success: Xerox PARC pioneered much of modern-day personal computer and its software. Although IBM invented a predecessor to the spreadsheet (expired U.S. Patent 3,610,902), it did not market a commercial product based on it; it also did not assert the patent even though its claims seem to read on (i.e., be infringed by) modern spreadsheets. These technologies became major product categories primarily through the efforts of small entities.

13. Small entities using patents are exceptionally cost-effective in encouraging innovation—especially compared to federal funding.
Table 2 shows a rough estimate of the efficacy of three major sources of innovation: federally funded, large entity, and small entity. Our results show that small entities are 7.5 times as cost-effective at stimulating innovation as large ones, and 200,000 times as cost-effective as federal funding. The U.S. grants patent rights to universities as part of its research contracts; thus patents are issued in all these areas and patents asserted is a reasonable measure of innovation. We believe a more scientific study would refine these results, but doubt it would change the basic conclusion.

As a software developer, you might review the patents discussed and put yourself in the place of each of the parties involved. If your product finds satisfied users, do you think better financed companies with stronger marketing organizations will market competitive products, using your innovations? If so, will patents be useful to you? If a patent is enforced against you, do you think you will be able to design around it? If you have to license it, do you think your competitors will also have to license it, thus passing the cost on to the end customers? Which problem would you rather have: a big company entering a market you developed, or finding out you were accidentally infringing a patent? Do you think the effect of software patents might be more innovation, higher software prices and an industry with more long-term profitability?

If you are protected by patents, your success depends in part on your patented inventions as others must deal with them. If you accidentally infringe a patent, designing around it is within your expertise. If you do not have patents, success depends much more on the ability to finance and market products—capabilities outside of your expertise and control. If you are a software developer, don't patents benefit you by manifesting your contributions in rights you can bring to the bargaining table, while confining the problems largely to your area of expertise and control? *Issues in Science and Technology* (Winter 1992) contains a letter from Commissioner of Patents Harry F. Manbeck who said of another article by the same authors that they [15]:

demonstrate they do not understand the current law . . . Most of their statements . . . do not appear to be the result of a balanced and reasoned inquiry and do not appear to be supported by the facts. . . . they cavalierly dismiss the view of those who appear to have used the patent system successfully and impugn their motives . . .

The PTO issued about 89,000 patents in 1990 from which the League, with the advantage of hindsight, can pick and choose the ones to attack. Consider the information presented here on patents the League selected to demonstrate the PTO's mistakes. Whose standards are higher, the League's or the PTO's?

Recommendations

After reviewing our results we can make some general recommendations.

1. Policy should be made on the assumption that innovation occurs in software as in other technologies until compelling evidence to the contrary is found.
This is consistent with the results described here. The operational implications are to continue to let the system operate as it is accepting evolutionary changes based on experience rather than speculation.

2. The PTO should be viewed as a source of innovation that competes for funding with other federally funded sources of innovation.
PTO fees should be reduced, especially for small entities, and the PTO should receive a higher level of funding to improve its ability to examine patents so it can issue better quality, more timely, patents in software and other technologies. European patent offices are much better equipped and much better funded. It seems that the PTO should compete with the NSF and other organizations for federal funding on the basis of their cost effectiveness in encouraging innovation.

In 1990 only $2 million of the PTO budget of $419 million came from federal funding; the remainder came from user fees. Superficially, it might seem that investing in the patent system will have a multiplier effect of 80,000 in creating innovation as compared to federally funded science. We suggest no such thing. We do however ask the question: If taxpayers were to spend an additional $160 million per year to support innovation, we could either increase the $64 billion federal funding on science by one-fourth of one percent (0.25%) or increase PTO funding by 40%, enabling the PTO to issue better patents and restore reduced user fees for small entities. Which will likely produce more innovation? Which will achieve a greater multiplier effect by encouraging additional private investment?

An example of federally funded science is fusion power research, which has been going on for at least 25 years, has cost hundreds of millions of dollars and has produced little practical result. Pons and Fleishman developed (and filed for patents on) cold fusion without government funding yet they, having invested their own money and not being in the mainline of governmental funding, are heavily criticized. While it is not clear that Pons and Fleishman have produced cold fusion, respected people in the field believe that they have, even if no one yet understands what is happening. This is just an example of how the system is biased in favor of government funding of expensive conventional solutions, and against individuals and small companies who risk their own time and money to innovate.

Individuals taking a contrary view have been the major source of new ideas in both science [38] and engineering [24]. This is why small entities and the patent system are so important. Most will fail, but the successes more than make up for the failures.

It would be interesting to evaluate the results of federally funded science to see which projects are worth the cost. Some projects may have become like those welfare mothers who, generation after generation, are entrapped in a governmental support system.

3. The patent laws should be modified to make it possible for small entities to assert their patent rights more effectively.
The data show it is commonplace for large companies to pirate the technology of small entities. No case was cited where a large company licensed a small entities' technology without first being sued suggests the existing laws do not motivate large companies to resolve patent disputes with small companies quickly. The issue here is not just fairness to inventors and improved efficiency in settling disputes. Rather, it is concerned with avoiding the waste that occurs because U.S. companies are so much slower at adopting new innovations than Japanese companies.

Congress responded with antipiracy legislation where software copyrights were concerned; we would hope it would similarly pass legislation to prevent patent piracy. Remedies similar to the criminal penalties for copyright infringement and Rule 11 sanctions for attorneys who file frivolous suits are worth considering. We suggest the following as possible remedies for patent disputes to stimulate discussion:

- After being put on notice, an "infringer" would have six months to file any prior art to be used to defend the infringement suit with the PTO. (I find it difficult to believe it is well-known art if it can not be found in six months.)

• If a patentholder prevails in a lawsuit, the remedies should include an extension of the period of exclusivity against that infringer equal to the length of time the suit was in progress.

• Discovery should be limited.

These suggestions, which should induce speedier resolution of patents disputes, are suggested for all patents disputes, not just software.

The patent system, an enormously productive system for inducing innovation, is being stymied by a cumbersome dispute resolution process. Is it in the public good to have a system of conflict resolution that discourages conflict resolution? Should innovators spend their time innovating or litigating? If the courts could resolve software patent and copyrights issues more quickly, it would clarify the law so everyone can make decisions with some predictability. The problems are not unique to patents but occur in all litigation. That the judicial and even the legal, community are beginning to address the inefficiency of dispute resolution and litigation is grounds for cautious optimism.

4. Further study of the role of patents and federal funding in software innovation is useful.
We are keenly aware that the sample is small and unscientific, and thus our results should be considered suggestive rather than definitive. A more definitive study should be useful in bringing out facts that would be useful in evaluating future changes to the patent law.

These recommendations can be summarized thus: Redress the balance of incentives so innovators will prefer to develop their ideas commercially, using patent protection rather than search for federal funding.

The Software Patent Confrontation

The software industry is getting more competitive. Almost every company that has hit products uses its cash flow to develop entries in other product categories. As a result, product categories are getting very competitive. Since most software companies have confined their intellectual property to source code copyrights, user interface copyright and trademarks, whenever they come up with a successful innovation, their competitors will often quickly replicate it. As a result, the impetus is toward similarly featured products competing on price, differing only in the mistakes which the originators must maintain to support their existing customers.

Now companies are recognizing that by using patents they can compete on features and function—not just tactically, but strategically. Even if competitors do replicate the features, they will likely make them different enough to avoid infringement. Companies following this approach will support standards, but their products will have a substantial proprietary component engendering products with more diverse feature sets. This will enable the industry to compete more on the profitable playing field of unique capabilities and market position and less on price. This is consistent with standard business school product marketing, where product differentiation and market segmentation are basic.

Intellectual property has already driven the market for those who got in early and established standards. Lotus owns the 1-2-3 standard. Novell owns a major network standard and WordPerfect, a major word processor standard. Apple owns the Macintosh User interface standard and Intel and Microsoft own the IBM compatibility standard. Patents give new companies the opportunity to establish and own something of value in the market based on their innovativeness rather than their marketing and financial capabilities.

While the problem of people accidentally infringing software patents has been greatly exaggerated, several patents will be successfully asserted against existing products. This will be primarily between those companies that focused on innovation and have patents, and those that focused on exploiting recognized business opportunities. This kind of confrontation occurred earlier in the aircraft and other industries [21].

During these confrontations, the businesses with a large volume of infringing products will understandably, feel "extorted" since they did not anticipate patent infringement. Such businesspeople will take support for their position from those who argue against software patents and advocate or suggest invalidating existing software patents ([15, 39]).

The software innovators who advanced the technology and made business decisions based on their patent rights will similarly feel cheated especially where they pioneered commercial products based on their inventions. When depositors made decisions based on gov-

ALPHA: Abraham Lincoln Patent Holders Association

This organization, founded in January 1992, supports the use of, and educates people about, software patents. Already, ALPHA's members include two software patentholders whose patents have been litigated, four patentholders whose patents are mentioned here, two former board members of the Software Publishers Association and lawyers from Merchant and Gould, Baker and McKenzie, Welch and Katz and the Franklin Pierce Law Center, and a former commissioner of patents and trademarks.

ernment guarantees of S&L deposits, no one suggested that the government default on its obligations to insured depositors, as people suggest the government invalidate existing software patents. No one vilifies the S&L depositors because the government has to pay them money; yet software innovators find themselves vilified with lies and half truths.

In this confrontation, both sides start out feeling cheated.

Many "infringers" will react emotionally and view it as a problem to be gotten rid of and many will fight to the bitter end. This raises the stakes, since a company having been put on notice may be liable for treble damages and attorneys' fees. Patentholders will not likely pursue these cases for four or five years to let the infringers' liability build up. After a suit is filed these companies will be getting much of their advice from those who have most to profit from the litigation: their litigators. This seems to be what is happening to Lotus.

Some software developers on finding out that the rules were not what they thought, face the problems of infringing others' patents while not having patented their own successful innovations. Some will chalk it up as one of many risks and uncertainties of business. Those who react emotionally might find it useful to first ask: Which of the players have acted in good faith? Which have not? Which have been responsible for the patent mess? Which have been innocent victims? Having answered these questions, such developers can more effectively target their wrath.

Companies that act rationally will analyze the patents to ascertain their scope and validity, whether infringement is occurring, and how easy it is to remove the "infringing" capability. They will check with other licensees. They will consider the obvious options, such as taking out a license, removing the infringing capability, finding prior art and showing it to the patentholder, or fighting in court if that is the only possibility. They will probably try to address the problem early, before the liability builds and consider negotiating a license or using some form of alternative dispute resolution to resolve infringement and validity issues. If the problem is associated with purchased product, most companies will stand in back of their products and provide a license, warrant it against infringement, or provide guidelines on how to avoid infringement. It will put its "no problem" in writing.

Astute companies will view infringement as an opportunity in disguise. If the patent is good and competitors are, or soon will be, infringing it, the first licenses can generally get an inexpensive license forcing competitors to pay more if they want to use the technology. It may be possible to get an exclusive license on some feature which differentiates a product from the competitor's. It can be worthwhile to see if the patent covers useful capability which could be added to the product. The best time to negotiate for the license is when you do not have the liability of infringement but can offer to create a demand for that capability by incorporating it into a product. Invention being the mother of necessity, your competitors will be faced with the choice of paying a higher price to license the technology or leave it out, thus differentiating your product from theirs.

In brief, what superficially looks like another problem to be dealt with in the increasingly competitive, commodities-oriented software business, might prove to be what makes products less *price* competitive. Many industries have worked on this basis all along: patents make industries more diverse in their offerings, more profitable, more innovative, and ultimately will

What You Can Do

Insist that the issues be debated. Don't let one side present its case unchallenged. If you need literature to distribute or a developer to join a debate on software, contact ALPHA.

One way to help correct the misinformation about software patents is to join ALPHA, The Abraham Lincoln Patent Holders Association. ALPHA is trying to correct the misinformation on software patents and to provide a forum for people to deal with issues of software patents, such as how to avoid infringement, setting licensing fees, finding prior art. Contact ALPHA at:

ALPHA
146 Main St., Suite 404
Los Altos, CA 94022

You can help get the issue discussed intelligently in Congress. Tell Congress it is important that the issues be discussed in open hearings where everybody gets to present his or her side, hear the issues debated and make up their own mind. Write:

House Subcommittee on Intellectual Property
2137 Rayburn Bldg.
Washington, D.C. 20515

Senate Subcommittee on Patents, Trademarks and Copyrights
U.S. Senate
Washington, D.C. 20510

E.R. Kazenske
Executive Assistant to the Commissioner
Commission on Patent Reform
U.S. Patent and Trademark Office
Box 25
Washington, D.C. 20231

make the U.S. more competitive.

The essence of this article is simple: Software intellectual property issues are not inherently different in substance from other technologies; what motivates people is not inherently different; industry life cycle is not inherently different; marketing and business strategies and tactics are not inherently different; the law and policy issues are not inherently different; the technology is not inherently new. Software has been around for 40 years. The issues may be new to those who had no experience with them. But the only difference is that software is a mass market industry for the first time and real money is at stake.

Acknowledgments
I would like to thank Steve Lundberg, John P. Sumner, Susan Nycum, Lewis Gable, George Gates, David Pressman and Tom Hassing for their many useful comments.

References

1. American Bar Association, *Two Hundred Years of English and American Patent, Trademark and Copyright Law.* 1977.
2. Axelrod, R. *The Evolution of Cooperation,* Basic Books. 1985.
3. Brooks, F. No silver bullet: Essence and accidents of software engineering. *Computer* (Apr. 1987).
4. Bruce, R. *Lincoln and the Tools of War.* University of Illinois Press, 1989.
5. Bulkeley, W. Will software patents cramp creativity? *Wall Street J.* (Mar. 14, 1989).
6. Bugbee, B.W. *The Genesis of American Copyright Law.* Public Affairs Press, Wash., D.C., 1967.
7. CBEMA comments on computer-related invention patents. *Comput. Lawyer* (Oct. 1991).
8. Chisum, D. The patentability of algorithms. *U Pitt. L Review 47,* 959,971, (1986).
9. Chisum, D. *A Treatise on the Law of Patentability, Validity and Infringement,* (7 volumes) M. Bender, 1978.
10. Choate, P. *Agents of Influence.* Touchstone, 1990.
11. Clapes, A. Software copyright, and competition. *Quorem* (1989).
12. Cringely, R.X. *Accidental Empires.* Addison Wesley, 1991.
13. Fisher, L.M. Software industry in uproar over recent rush of patents. *New York Times,* May 12, 1989.
14. Foley, et al. *Computer Graphics: Principles and Practice,* Second ed. Addison Wesley, 1990.
15. Garfinkel, S., Stallman, R. and Kapor, M. Why patents are bad for software. *Issues in Science and Tech.* (Fall 1991).
16. Heckel, P. The Wright Brothers and Software Innovation, in *The Elements of Friendly Software Design,* Second ed., Sybex, 1991 (First ed. Warner Books, 1984).
17. Heckel, P. Isolating differences between files. *Commun. ACM* (Apr. 1978).
18. Heckel, P. Software patents and competitiveness. Op Ed, *San Francisco Examiner,* July 8, 1991, p. A-13.
19. Heckel, P. Zoomracks, designing a new software metaphor. *Dr. Dobbs J.* (Nov. 1985).
20. Heckel, P. and Lampson, B. A terminal oriented communications system. *Commun. ACM* (July 1977).
21. Heckel, P. and Schroeppel, R. Software techniques cram functions and data into pocket sized microprocessor applications, *Elect. Des.* (Apr. 12, 1980).
22. Howard, F. *Wilbur and Orville.* Knopf, 1988.
23. *In re Pardo,* 684 F.2d 912 (C.C.P.A. 1982).
24. Jewkes, J., Sawers, D. and Stillerman, R. *The Sources of Invention,* Second ed., Norton, 1969.
25. Judas, J.B. Innovation, A casualty at IBM, *Wall Street Jo.* (Oct. 17, 1991), A23.
26. Kahin, B. The software patent crisis. *Tech. Rev.* (Apr. 1960), 543–58.
27. League for Programming Freedom. Against software patents. *Commun. ACM* (Jan. 1992).
28. League for Programming Freedom. Software patents. *Dr. Dobbs J.* (Nov. 1990).
29. Lewis, T. Empire of the air. *HarperCollins* (1991).
30. Lincoln, A. *Selected Speeches and Writings,* Vintage, 1992.
31. Mansfield, E. Industrial innovation in Japan and the United States. *Science* (Sept. 30, 1988).
32. McCloskey, M. Intuitive physics. *Sci. Am.* (Apr. 1983), 123.
33. Neustadt, R.E. and May, E.R. Thinking in time: The uses of history for decisionmakers. *The Free Press,* 1986.
34. Nycum, S. Legal protection for computer programs. *Comput. Law J. 1,* 1 (1978).
35. Orwell, G. *Politics and the English Language.*
36. Pike, R. Graphics in overlapped bitmap layers. *ACM Trans. Graph. 17,* 3 (July 1983), 331.
37. Ritter, T. The politics of software patents. *Midnight Eng.* (May–June 1991).
38. Root-Bernstein, R. *Discovering.* Harvard University Press, 1989.
39. Samuelson, P. Benson revisited: The case against patent protection for algorithms and other computer program-related inventions, *Emory Law J. 39,* 1025, (1990).
40. Samuelson, P. CONTU revisited: The case against copyright program for computer programs in machine readable form. *Duke Law J. 663* (1984), 705–53.
41. Samuelson, P. Should program algorithms be patented? *Commun. ACM,* (Aug. 1990).
42. Samuelson, P. and Glushko, R. Survey on the look and feel lawsuit, *Commun. ACM* (May 1990).
43. Schon, D. Technology and Change, Delacorte, 1967.
44. Schwarzer, W. Science in the Courtroom. 15th Annual Intellectual Property Law Institute, Intellectual Property Section of the California State Bar, Nov. 1990.
45. Schwartz, E. The coming showdown over software patents. *Bus. Week* (May 13, 1991).
46. Sumner, J. and Lundberg, S. The versatility of software patent protection: From subroutines to look and feel. *Comput. Lawyer* (June 1986).
47. Sumner, J. and Lundberg, S. Software patents: Are they here to stay? *Comput. Lawyer* (Oct. 1991).
48. Slutsker, G. and Churbuck, D. Whose invention is it anyway? *Forbes* magazine (Aug. 19, 1991).

About the Author:
PAUL HECKEL has a small company that develops software. He also consults in the area of user interface design and has been used as an expert on software user interface intellectual property issues by the legal community. **Author's present address:** HyperRacks, Inc., 146 Main St., Suite 404, Los Altos, CA 94022.