Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?

By Mann, Susan O
Publication: The University of Chicago Law Review
Date: Thursday, January 1 2004
Subject: Commercial law, Intellectual property, History, Trade secrets, Software industry
Location: United States

Over the past few decades, the software industry has emerged as one of the most important sectors of the economy. Performance gains in computer hardware, advances in software functionality, and the growth of the Internet into an established communications and commercial medium have fueled the integration of software into nearly every aspect of modern life. Broad-based investments in information technology (IT) have helped the United States achieve impressive levels of productivity growth and have made the software industry one of the most vibrant segments of the global economy.

Intellectual property (IP) laws have had an important impact on the software industry's success. IP protection has given software developers the incentive to invest in developing and marketing new programs by providing a legal mechanism through which developers can capture at least some of their software's value-whatever that may be-in the marketplace. Without IP protection, second-comers could simply copy the innovation and thereby appropriate at least some portion of its economic value, without having to bear any related development costs. The possibility that third parties might "free ride" on the original inventor's investment in this manner increases the risk that the developer might be unable to earn a competitive return on this investment in the marketplace, thereby diminishing or even eliminating the inventor's incentive to invest in future innovations.[1]

Software developers typically confront two distinct types of free-riding risks. The first risk is that third parties will make wholesale, literal copies of a program, then further copy or distribute these "pirate" copies in a manner that suppresses demand for genuine product from the original developer. The second free-riding risk is that later firms will copy specific elements, features, or technologies embodied in an original software program, but without engaging in the kind of wholesale or literal copying that characterizes piracy. Such "follow-on" copying involves replicating the functionality or appearance of the original program, albeit typically through the use of different or independently produced program code. Both piracy and follow-on copying diminish incentives for innovation because both make it more difficult for the original developer to realize a competitive return on its development costs.

At times, however, acts that may otherwise impinge upon IP rights have been regarded as necessary to promote IT interoperability. The past two decades have seen massive growth in the number and diversity of IT devices, platforms, and applications, which do not always interoperate easily with one another. Given the growing need for interoperability, governments have sometimes concluded that it is necessary, under carefully defined circumstances, to permit certain forms of "reverse engineering" of software programs, even where these acts otherwise would-but for a clear-cut statutory exception infringe on the original developer's IP rights in such programs.

Since at least the 1960s, software developers have relied principally on three distinct IP regimes to protect their programs against third-party appropriation: trade secret, copyright, and patent law. Although the growth of the software industry has corresponded with an ongoing (if gradual) strengthening of all three forms of IP protection, both domestically and internationally, the scope of protection offered by these regimes has varied significantly over time, as has the software industry's reliance on them.

This Essay contends that the history of the software industry can be divided into at least two phases, each characterized by distinct technologies and market structures, which in turn have influenced the significance of the available IP regimes. During the first phase, software's tight integration with hardware and the IT industry's vertical structure led IT firms to rely primarily on trade secret protection and contract law to guard their innovations against appropriation by others. In the second phase, which emerged in the early 1980s and continues in certain respects to this day, software's separation from hardware and the industry's new horizontal structure based in large part on mass-market business models led software developers to rely more heavily on copyright than on other forms of protection.

Recent developments in IP law, together with technological innovations and broader changes in the IT industry, suggest that the software industry may now be entering a third phase. These changes have highlighted copyright's somewhat limited ability to provide appropriate protection against certain forms of copying and have made trade secret law a less attractive option for IT firms and their customers. Developments in the patent area, however, suggest that patent protection may emerge as a critical form of IP protection for software during this new phase.

I. THE FIRST PHASE: 1950S-1970S

A. Technology, Marketplace, and Law

Until at least the mid-1970s, commercial software development rested largely with a handful of hardware manufacturers that catered primarily to large enterprises. The IT systems offered by these manufacturers typically ran custom-built software specifically designed to run on massive mainframe machines that these vendors also supplied and often serviced. Software developers and their customers typically had a direct contractual relationship with one another, and because most software programs were task-specific and customized to the customer's unique needs, vendors had little incentive to make their systems interoperable with those offered by others. This market structure left few opportunities for competing firms to offer complementary products or services. Moreover, because most of the leading firms focused on selling expensive mainframe machines, computing generally remained beyond the reach of small firms, schools, individual consumers, and other segments of society.[2]

Throughout this era, most IT firms viewed secrecy as the best means of protecting their software against unauthorized copying. Thus, the IT industry widely regarded trade secret law (supplemented by contractual restrictions with customers) as the principal legal mechanism for protecting their software against misappropriation.

Trade secret law was not, however, the only legal remedy available to software developers during this period. Already in 1964, the Register of Copyrights announced that the Copyright Office would accept claims to register software.[3] Yet the conditions imposed on such registration-including proof that the program contained sufficient "original authorship" to qualify for protection and had been "published" prior to registration-appear to have lessened the appeal of copyright protection for many software developers. Moreover, the Register noted that software programs might, in certain circumstances, not qualify as a "writing of an author," and that programs in executable-code form might not qualify as "copies" within the meaning of the 1909 Copyright Act then in effect.[4] Given these hurdles, relatively few software developers took advantage of copyright protection for their programs. Between 1964 and January 1, 1977, only 1,205 software programs were registered for protection under the Copyright Act, and over 80 percent of these were registered by two companies, IBM and Burroughs.[5]

Although the Federal Patent Act and state unfair competition law theoretically offered alternative avenues for protecting software during this period, in practice they supplied little real protection. Unfair competition laws normally required proof of misappropriation or "passing off," and as such were generally viewed as ineffective against many forms of third-party copying.[6] While some commentators believed that at least certain elements of software programs should qualify for patent protection, a series of Supreme Court decisions during this period, while never directly addressing the patentability of software, seemed to place substantial barriers to obtaining patents on innovation embodied purely in software.[7]

B. The Software Industry in Transition

1. Technological underpinnings.

Two series of technological innovations propelled a transformation of the software industry in the 1970s and early 1980s. The first can be traced to the invention of the integrated circuit in 1958 by Jack Kilby and Robert Noyce.

Innovations in integrated circuits and "microprocessors" prompted a second series of technological innovations that laid the foundation for the transformation of the software industry. In the early and mid-1970s, software developers began to realize that innovations in computer hardware would drive down costs to the point where computers would soon be affordable to average consumers. As a result, several companies began designing software and IT systems for the small but growing market of individual and small business users.

These changes prompted the emergence of a new generation of independent software developers, many of whom adopted massmarket business models in order to exploit previously untapped economies of scale. These firms often distributed software in packaged form through a wide range of distribution channels, separately from the hardware on which it ran. With the widespread adoption of these mass-market practices, prices fell, competition and innovation increased, and the range and diversity of available products and services grew.[8] As a result, computer use skyrocketed, and the vertically structured and homogenous IT market was replaced by a horizontally structured, diverse software industry comprising thousands of firms serving hundreds of discrete markets.

This transformation, however, also raised a series of technological and legal challenges for software developers, including the problem of unauthorized copying. As hardware became more powerful and less expensive, people found it increasingly profitable to copy software, either as a means to avoid purchasing authorized copies, or to sell such unauthorized copies to third parties. At the same time, the emergence of mass-market distribution models for packaged software led some commentators to question whether trade secret law remained a viable mechanism for combating such third-party copying, at least in certain circumstances.[9]

2. Legal response-CONTU.

Concerns regarding IP protection for software came to a head as Congress set about revising the Copyright Act, a process that culminated in the enactment of the Copyright Act of 1976.[10] Lawmakers, unable to agree on language regarding the scope of protection for computer programs, established the National Commission on New Technological Uses of Copyrighted Works (CONTU) to study the issue and make recommendations.[11] After commissioning several studies and hearing testimony from dozens of witnesses, CONTU recommended that copyright protection extend to computer software, including software in object-code form.[12]

CONTU's recommendation was not without its critics. Commissioner John Hersey, for instance, issued a pointed dissent in which he argued that no compelling evidence had been offered that copyright protection would promote innovation in the software industry, or that the existing framework of legal and technological protections for software was inadequate.[13] Noting that the hardware and IT systems markets were dominated by four companies - IBM, Burroughs, Honeywell, and Sperry-Univac - Commissioner Hersey predicted that the likely effect of extending copyright protection to software would be to "strengthen the position of the large firms, to reinforce the oligopoly of these dominant companies, and to inhibit competition from and among small independents."[14] Commissioner Hersey further worried that extending copyright protection - traditionally dedicated to protecting expression by and for humans - to object code, which could be interpreted only by a computer, would signal an "equivalence[] of human beings and machines" in the eyes of the law, a result that would invariably "impoverish" society in the long run.[15]

In hindsight, these objections seem misguided. Copyright protection has given significant impetus to the growth of a vibrant software industry, and software firms of all sizes routinely rely on copyright law to prevent unauthorized reproduction and distribution of their programs. Rather than entrenching the positions of the leading IT firms of the day, copyright protection provided the foundation for a new generation of software providers that greatly expanded the range and diversity of cost-effective software options available to consumers. Far from diminishing the value of human creativity, the growing range of software programs to which copyright protection provided an impetus vastly improved the means through which people could create, distribute, and enjoy creative works of all types.

Nevertheless, CONTU's recommendations left unresolved three critical issues, and ongoing attempts to resolve these issues in the courts have had a significant impact on IP protection for software. First, despite its best efforts, CONTU did not articulate clearly the point at which the "expressive" (and therefore protectable) elements of a computer

program end and the unprotectable elements-such as ideas, methods of operation, and so forth-begin.[16] Acknowledging that "[t]o attempt to establish such a line in this report written in 1978 would be futile,"[17] CONTU instead left this task to "the institution designed to make fine distinctions - the federal judiciary."[18]

Second, CONTU failed to address the key interoperability challenges that were beginning to confront the emerging mass-market software industry. Although CONTU recognized that the goal of achieving IT interoperability might conflict at times with copyright protection for software, it failed to anticipate the specific types of interoperability challenges that would confront mass-market hardware manufacturers and software developers.[19]

Third, CONTU did not resolve the issue of whether extending copyright protection to software might impact the availability of protection under other legal regimes, particularly trade secret and patent law. With respect to patents, CONTU appeared to acknowledge that certain elements of software might, at least in theory, fall within the scope of patentable subject matter.[20] At the same time, while recognizing that the availability of copyright protection would not preempt the availability, as a legal matter, of trade secret protection, CONTU hinted that copyright might one day come to supplant trade secret law as the preferred method of protecting software, particularly in the emerging mass-market software industry.[21]

Despite these unresolved issues, Congress adopted CONTU's recommendations in 1980, thereby expressly bringing software within the statutory scope of copyrightable subject matter. Congress left to the courts the task of demarcating the line between the copyrightable and uncopyrightable elements of software and determining how best to accommodate the goal of interoperability with copyright protection, as well as clarifying the relative scope of patent, copyright, and trade secret protection in software. As the courts began to tackle these issues, it became increasingly apparent that they were, to some degree, intertwined.

II. THE SECOND PHASE: 1980s-1990s

In many respects, Congress's decision to bring software expressly within the reach of copyright law just as the software industry was maturing into its second phase served the IT industry well. Copyright protection enabled software developers to distribute their programs to an unlimited number of customers and through a wide range of distribution channels without jeopardizing their rights in such programs. The fact that most developers distributed their programs only in object code form often enabled them to rely on trade secret protection for the inner workings of the program expressed in source code, while copyright protected the object code itself.[22] Copyright also mitigated the problems that otherwise might have arisen from the absence of a direct contractual relationship between developers and end-users (though the prevalent use of end-user license agreements for software provided a partial substitute for such direct contractual relationships). Copyright varied significantly, however, in its capacity to resolve the challenges of piracy, follow-on copying, and interoperability in a way that adequately protected developers against third-party appropriation.

A. Copyright and Piracy

Although piracy - the wholesale, literal copying of a computer program - emerged as one of the principal business challenges facing software developers during the 1980s and 1990s, U.S. courts had little difficulty concluding that copyright law prohibited most forms of piracy. In the seminal case of Apple Computer, Inc v Franklin Computer Corp,[23] for instance, the Third Circuit held that computer programs, whether in source or object code form, qualified as "literary works" under the Copyright Act and that a competitor's wholesale copying of software infringed on the original developer's copyrights in such programs.[24] In doing so, the court rejected the argument that software programs, and operating systems in particular, constituted "processes," "systems," or "methods of operation" that placed them beyond the scope of copyright protection.[25] Since Apple Computer, no U.S. court has disputed the proposition that the wholesale, literal copying of a protected software program is proscribed under the Copyright Act.

Over the course of the 1980s and 1990s, the recognition that software was entitled to protection as a literary work under copyright law was increasingly accepted outside the United States. In 1978, the World Intellectual Property Organization (WIPO) issued model law provisions in accordance with this view.[26] In 1991, this view was further solidified with the adoption of the European Community's Directive on the Legal Protection of Computer Programs, which expressly directed the Community's Member States to amend their copyright laws as necessary to protect software as a literary work.[27] Finally, the 1994 WTO Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS Agreement) expressly provided that computer programs, whether in source or object code, are entitled to copyright protection as literary works.[28]

In sum, over the course of the 1980s and 1990s, national laws and international norms came to reflect the strong consensus that copyright protection extended to software, and that the wholesale, literal copying of a computer program, whether in source or object code, infringed on the rights of the copyright owner.

B. Copyright and Interoperability

Whereas copyright's capacity to proscribe wholesale copying was relatively straightforward, balancing developers' need for protection, on the one hand, against acts of unauthorized copying that promote interoperability, on the other, proved to be a greater challenge, particularly due to a process of reverse engineering software known as "decompilation."

Software programs are typically written as a series of formal instructions known as source code. Access to a program's source code can promote interoperability by revealing a program's interface specifications, which allows subsequent developers to ensure that their own programs can share data with the original program. Where a developer does not have access to these specifications, the developer may seek to "decompile" a program by translating the object code into a human-legible form that resembles the source code. Decompilation can also make it simpler for subsequent developers to imitate the program and develop close substitutes that compete directly with the original program. Most forms of decompilation result in the creation of copies of the decompiled program-either exact copies or, more commonly, derivative, "intermediate" copies of the original program.

Despite initial doubts that copyright law permitted such copies, U.S. courts generally came to endorse the view that, to the extent necessary to promote interoperability, unauthorized decompilation normally does not violate copyright. For instance, in Atari Games Corp v Nintendo of America Inc,[29] the Federal Circuit held that Atari's decompilation of software embedded in Nintendo's hardware console, for the purpose of ensuring that Atari's video games could run on the console, was excused under the Copyright Act's fair use provisions.[30] The court, however, emphasized the narrowness of its holding by noting that "[f]air use to discern a work's ideas . . . does not justify extensive efforts to profit from replicating protected expression. . . . Any reproduction of protectable expression must be strictly necessary to ascertain the bounds of protected information within the work."[31] The court was also clearly troubled that a holding in favor of Nintendo would effectively expand copyright doctrine into an area traditionally considered to be within the purview of patent law.[32] Over the years, several courts have agreed that decompiling software for the sole purpose of revealing information necessary to achieve interoperability may be excused as fair use under the Copyright Act.[33]

Just as courts in the United States were confronting potential conflicts between copyright protection and decompilation, a similar debate was raging across the Atlantic in the context of the proposed EC Software Directive. As ultimately enacted, Article 6 of the Directive permits lawful users of a software program to decompile the program solely for the purpose of achieving interoperability with other programs.[34] Moreover, decompilation is excused only if "the information necessary to achieve interoperability has not previously been readily available" to such users and "these acts [of decompilation] are confined to the parts of the original program which are necessary to achieve interoperability."[35] Finally, the Directive provides that information obtained through decompilation may not be used for purposes other than to achieve interoperability.[36] Thus, the Software Directive sought to avoid shifting the balance of copyright protection for software in a manner that would allow later firms to appropriate the creative expression of the original software developer in order to develop imitative products.[37]

C. Copyright and Follow-On Copying

Although courts readily held that literal, wholesale copying of software could violate copyright law, cases involving non-literal or piecemeal copying in order to replicate a particular element, feature, or technology met with a less uniform judicial response. Taken together, the decisions reflect a general reluctance on the part of courts to extend copyright protection to elements such as a program's structure, sequence, or organization-features that do not fall squarely within more traditional conceptions of copyrightable expression.

Initially, however, several courts seemed to suggest that copyright protection for software should be construed broadly to protect against, not only literal copying, but also copying a program's more abstract elements as well. In Whelan Associates, Inc v Jaslow Dental Laboratory, Inc,[38] for instance, the court rejected the claim that, because the defendant had not literally copied the plaintiffs software program, it could not be found liable for copyright infringement.[39] Instead, the court ruled that the purpose or function of the program constituted the "idea" of the program, "and everything that is not necessary to that purpose or function would be part of the expression of the idea" and therefore entitled to protection.[40]

Six years after Whelan, the Second Circuit offered a decidedly narrower interpretation of the scope of copyright protection for software in Computer Associates International, Inc v Altai, Inc.[41] As in Atari Games, the court in Computer Associates also noted that its decision was motivated at least in part by its concern not to extend copyright protection into the domain traditionally protected by patents:

[I]t may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes. This results from the hybrid nature of a computer program, which, while it is literary expression, is also a highly functional, utilitarian component in the larger process of computing. Generally, we think that copyright registration-with its indiscriminating availability-is not ideally suited to deal with the highly dynamic technology of computer science. Thus far, many of the decisions in this area reflect the courts' attempt to fit the proverbial square peg in a round hole. The district court and at least one commentator have suggested that patent registration, with its exacting, up-front novelty and non-obviousness requirements, might be the more appropriate rubric of protection for intellectual property of this kind.[42]

The divergence of approach between the holdings in Whelan and Computer Associates was manifested in two key decisions involving Lotus Corporation's popular spreadsheet program, Lotus 1-2-3.[43] In Lotus Development Corp v Paperback Software International,[44] Lotus claimed that Paperback's competing spreadsheet program, VP-Planner, infringed Lotus 1-2-3 by copying its menu command structure, including specific literal and non-literal elements such as the menu's command terms, the structure and order of those terms, and their presentation on the screen.[45] The district court, relying on Whelan, agreed.[46]

In the meantime, Borland International introduced Quattro Pro, a spreadsheet program that offered its own menu command structure, but also included an "emulator" that allowed users to operate the program using the traditional Lotus 1-2-3 command structure, as well as compatibility with Lotus 1-2-3 macros. Lotus brought suit for copyright infringement, alleging that the emulator's literal copying of Lotus 1-2-3's menu command structure, as well as Quattro Pro's compatibility with Lotus 1-2-3 macros, infringed on Lotus's copyright in Lotus 1-2-3.

On appeal as Lotus Development Corp v Borland International,[47] the First Circuit held that Lotus's menu command hierarchy constituted a "method of operation" and, as such, was expressly excluded from copyright protection under 102 of the Copyright Act.[48] In a separate concurrence, Judge Boudin echoed the concern expressed in Atari Games and Computer Associates that extending copyright protection to a program's menu commands would "have some of the consequences of patent protection in limiting other people's ability to perform a task in the most efficient manner."[49] He added: "It is no accident that patent protection has preconditions that copyright protection does not - notably, the requirements of novelty and nonobviousness - and that patents are granted for a shorter period than copyrights."[50]

Whereas Computer Associates adopted a relatively narrow view of copyright protection with respect to non-literal copying, Borland marked a significant narrowing of protection even in cases of limited literal copying. In the period since these decisions, most courts have followed the reasoning of Computer Associates and Borland and have largely

abandoned the more expansive view of protection articulated in Whelan and its progeny.[51]

D. The Evolution of Patent Protection for Software

Even as courts were grappling with delineating the reach of copyright protection for software, the possibility that patent law might afford an alternative means of protecting software-based innovations was beginning to intrigue many legal practitioners - and some software developers as well. Although a series of Supreme Court decisions over the course of the 1970s led some in the U.S. legal community to conclude that software would rarely qualify for protection under the Patent Act, a signal that the tide might be turning under U.S. law came with the Supreme Court's decision in Diamond v Diehr.[52]

In Diehr, the Supreme Court held that, although a mathematical formula embodied in a software program might not qualify for patent protection on its own, application of the formula to perform a useful process-in Diehr, a process for curing synthetic rubber-did qualify for patent protection.[53] Although patent applicants after Diehr were somewhat more confident that their claims for software-based inventions fell within the scope of the Patent Act, they were nevertheless careful to draft their applications as claiming machines or processes performing specific, useful tasks.[54] For the next decade, the Patent and Trademark Office (PTO) examination of "software" applications revolved around the existence and significance of a "mathematical algorithm" with a claim that defined the applicant's invention, using the so-called Freeman-Walter-Abele test.[55]

The holding in Diehr was significantly broadened in 1994 with the Federal Circuit's decision in In re Alappat,[56] which held that Diehr's useful function requirement could be satisfied by drafting the relevant claim to include the software running on a general purpose computer.[57] Thus, after Alappat, patent applicants needed only to define their claims in terms of a computer program implemented in a machine in order to bring their claims within the scope of patentable subject matter.[58]

Alappat's machine-implementation requirement itself fell only one year later when IBM, in In re Beauregard,[59] appealed the PTO's rejection of a claim to a computer program embodied in a floppy diskette.[60] Rather than contest IBM's appeal, the PTO announced that it would not oppose the claim and, soon thereafter, issued new examining guidelines indicating that the PTO would accept claims for software-based inventions regardless of whether such inventions were implemented in hardware.[61]

Just as the PTO and the courts were gradually opening the door to embrace "pure" software inventions, such patents were increasingly making their mark in the IT marketplace. Over the course of the 1990s, several companies succeeded in licensing patented, software based technologies, often under terms that generated substantial royalty revenue.[62] At the same time, cross-licensing of software patent portfolios among IT companies became a common and accepted industry practice. In these and other ways, commercial relationships within the IT industry demonstrated that, whatever the legal issues surrounding the validity and scope of software patents, such patents had significant real-world value.

E. IP Protection for Software at the New Millennium: Observations

As noted above, software developers' growing reliance on copyright during the 1980s rested in part on their belief that copyright would allow them to prevent not only wholesale, literal copying of their programs, but follow-on copying as well. Moreover, while many developers continued to rely, at least in part, on trade secret law to guard against such copying, some wondered whether trade secret protection might require developers to distribute only object-code versions of their programs-a result that arguably stood in some tension with the growing market demand for IT interoperability and product transparency.[63]

The limited ability of copyright to protect software developers against certain forms of follow-on copying, combined with trade secret's tension with market demand for product transparency and interoperability, has led many software developers to explore additional avenues for protecting their programs against third-party appropriation. These developments suggest that the software industry might be entering a third phase in its evolution, one in which software patents play a more central role.

III. ENTERING A THIRD PHASE? THE ROLE OF SOFTWARE PATENTS

Software developers have responded to the demand for greater interoperability and product transparency in a variety of ways. First, many developers document and disclose essential interfaces and protocols so that independent developers and hardware manufacturers can easily write interoperable programs and hardware device drivers. As previously noted, some developers also provide access to their programs' underlying source code and combine these with opportunities for licensing so that users and others can take advantage of these disclosures for their own purposes. In addition, firms across the IT industry regularly contribute to developing voluntary, industry-wide technology standards.

If the common theme that unites these trends is to facilitate "openness" in a manner that also provides incentives for innovation, there are good reasons to believe that patent law may offer a superior regime to both copyright and trade secret law for protecting at least certain elements of software programs. First, in contrast to copyrights and trade secrets, a prerequisite for patent protection is that the inventor must disclose a clear and precise description of his or her invention, thereby promoting the goals of technological disclosure and IT product transparency.[64]

Second, whereas copyrights protect only the author's original expression of an idea, patents protect the actual invention, not just a single implementation of it. Thus, patent protection enables software developers to share key technologies with partners, customers, and others (even competitors) without significantly diminishing the developer's ability to prevent second comers from slavishly copying those aspects of a software program that are truly novel and innovative. In this manner, patent protection may be better suited than either trade secret law or copyright for enabling software developers to maintain the integrity and value of their IP assets in ways that are consistent with promoting interoperability and product transparency.[65]

Third, as many courts have recognized, patent law offers a distinct form of protection - and serves different policy goals - than does copyright law. Patents seek to promote technological progress by giving exclusive rights in discrete inventions in exchange for early public disclosure of the invention. Exclusivity gives the innovator control over the

patented invention. This, in turn, enables the patent owner to realize economic benefits, either through sales or licensing. Exclusivity provides both an economic incentive for the initial invention and its commercial development, as well as a stimulus for the development of new, noninfringing technology through other independent inventions or design-arounds.[66]

The patent examination process is designed to ensure that legal protection will extend only to technologies that are truly novel, useful, and non-obvious. Whether in terms of the European requirement of an "inventive step,"[67] or in terms of a non-obvious advance over existing technology,[68] issued patents must embody something truly new and innovative. Copyright law, by contrast, promotes creativity by protecting any original work of authorship fixed in a tangible medium of expression; any work that does not copy the expressive content of another protected work will be entitled to copyright protection regardless of whether it is new, useful, or constitutes an advance over preexisting works. This substantially lower threshold of protection for copyrights as compared to patents suggests that patent protection may provide a greater incentive than copyright for software developers to focus their efforts on achieving truly innovative advances in technolgy.[69]

Finally, patent laws typically grant innovators twenty years or less of protection from the time the patent application is filed.[70] This period of exclusivity is decades shorter than the fifty-plus years of protection generally afforded by national copyright regimes and the theoretically unlimited term of protection available under trade secret law. Thus, patented innovations are likely to enter the public domain more rapidly than works or know-how protected by either copyright or trade secret.

In these respects, patent protection for software provides a desirable form of protection for many forms of software innovation and may offer a more effective mechanism than either copyright or trade secret law for balancing incentives for innovation against the goals of interoperability and transparency. Nevertheless, software patents have been the subject of a fair amount of criticism, including the following:

Lack of qualified patent examiners. A frequently expressed concern is that the PTO has insufficient staff to review software applications and that existing examiners lack expertise in current software technology. Such concerns were expressed vociferously during the public hearings held by the PTO in 1992 and 1994, and again in 1998, following the Federal Circuit's landmark decision in State Street Bank & Trust Co v Signature Financial Group, Inc,[71] which held that business methods implemented in software may satisfy the subject matter and utility requirements of the Patent Act.[72]

Inadequate database of prior art. Some have argued that PTO examiners do not have access to sufficiently comprehensive databases of non-patent prior art. This deficiency, it is argued, has led PTO to issue patents on software technologies that were obvious in light of the existing art.[73]

Impact on small firms and individual developers. Some contend that software patents will benefit primarily large firms and will provide few if any advantages for smaller firms and individual software developers. These critics often point to the time and expense involved in prosecuting patent applications and contend that small firms might lack the resources to acquire software patents.

Impediment to innovation. Finally, some critics contend that software patents may impede innovation in ways that copyright and trade secret protection do not.[74] Some of these concerns rest upon the perception that software patents tend to be granted too broadly and, accordingly, are largely a function of the concerns regarding qualified patent examiners and prior art databases already mentioned.

One important response to these criticisms is that they rely on a stark definitional distinction - specifically, between software-based and non-software-based inventions - that often cannot withstand scrutiny in light of modern technology. Increasingly, discrete IT innovations can be implemented in either hardware or software. Furthermore, the explosive growth of "embedded" software in recent years means that many products and devices we normally consider hardware actually rely on software in order to function properly. Ongoing advances in hardware power and software sophistication are likely to further blur distinctions drawn between hardware and software. In such cases, the innovation that is the subject matter of the patent application may not be easily divisible into software and non-software components.[75]

Thus, attempts to single out certain innovations for special (discriminatory) treatment under patent law based on the fact that such innovations can be implemented in software will become increasingly untenable on either policy or pragmatic grounds. Moreover, legal rules designed to limit or deny patent protection specifically to inventions that are or can be implemented in software are likely to constitute a step backward to the days when inventors and patent attorneys expended great effort in describing their inventions in ways that minimized the role of software.[76]

These arguments suggest that neither policy nor technology offers a compelling justification for limiting or denying patent protection to software-based innovations. At the same time, the IT industry, legal practitioners, and policymakers must take seriously and respond to the criticisms set out above. Although recent empirical studies of business method and software patents suggest that some of these initial criticisms might in fact have been misplaced, efforts to respond to these criticisms have already made important headway.

A. Adequate Training and Funding

In the United States, several steps have been taken to address concerns over examiner training and patent office funding. The PTO recently provided additional staffing and training in several disciplines involving software-related technologies-particularly for applications in Class 705, which embraces most business method patents and which includes a relatively high proportion of software patents.[77] Also, in March 2000, the PTO announced that it would institute an extra level of examiner review for business method patent applications.[78] In the first year after this reform went into effect (PTO fiscal year 2001), the allowance rate of business method patents fell from 56 percent to 36 percent.[79] Further, the PTO has expanded its use of the "second pair of eyes" procedure to other technologies, including software patents primarily classified in classes other than Class 705,[80] thus ensuring greater quality assurance for other software-related patents.

With respect to funding, Congress is currently considering legislation that would authorize an increase in the fees PTO charges for various review-related services.[81] The

PTO contends that enactment of this legislation will enable the Office to invest in the personnel and technology resources it needs to improve its operations. Moreover, as of the time of this writing, industry and policymakers are exploring ways to ensure that increased revenues are made available to the PTO in the year they are collected.

Similar reform efforts are taking place outside the United States as well. For instance, Japan's patent office recently published a proposal to improve the pace and quality of patent examinations by expanding its staff of patent examiners and by working with private-sector entities to review certain cutting-edge technologies.[82] In a related move, a government body recently proposed that Japan create a high court specializing in patent disputes and related intellectual property matters.[83]

These initiatives constitute important steps in improving the quality and efficiency for the review of software patent applications. More importantly, they demonstrate that patent offices often will have the ability to respond to questions over patent and examination quality. Early indications in the United States suggest that these initial reform efforts have already resulted in improvements in processing time as well as the quality of software patents.

B. Improved Prior Art Databases

Although critics of software patents often contend that prior art is particularly deficient with respect to software technologies, recent research suggests that the quality and quantity of prior art references in patents for software-based inventions may in fact be equal-and in some cases superior-to such references in patent applications generally.[84] At the same time, efforts are underway to improve examiner access to prior art. For instance, the Scientific & Technical Information Center-Electronic Information Center (STIC-EIC), which assists PTO examiners in conducting prior art searches, is currently working on an initiative to collate examining resources in a web-based tool for Class 705 examiners, which will collect databases, web sites, and electronic and print literature resources of Class 705 topics.[85]

C. Software Patents and Smaller Developers

Whether the availability of patent protection for software will ultimately provide more advantages to smaller firms or to larger firms is an unanswered question. Yet there are reasons to believe that software patents may serve to level the playing field between small and large firms.

First, because patent law prohibits the unauthorized copying of discrete patented technologies, patent protection may enable smaller firms more effectively to prevent their larger competitors from capitalizing on their patented innovations. Furthermore, to the extent patent protection provides a more robust form of protection against third-party copying than copyright or trade secret law, start-up firms that secure patent protection for their software innovations may be more likely to attract critical venture capital funding than those that rely solely on copyright. As Robert Merges has noted, "the connection between patents and venture capital financing is a well-accepted part of Silicon Valley practice."[86]

One important way in which the costs of patent protection can be reduced is to improve the international harmonization of patent laws. The past thirty-five years have seen some progress on this front. The Patent Cooperation Treaty of 1970, administered by WIPO, has made it significantly easier for inventors to secure patent rights in multiple jurisdictions by providing for the filing of a single international patent application, which has the same effect as filing a national patent application in each of the designated countries.[87] More recently, ratification of the WTO TRIPS Agreement created the world's first enforceable substantive and procedural standards for patent protection.

Nevertheless, more can be done to strengthen the international treaty framework and ensure a consistent level of patent-rights recognition and enforcement throughout the world. Negotiations are currently underway between the Member States of WIPO to draft a Substantive Patent Law Treaty (SPLT). The goal of the SPLT is to resolve discrepancies between the substantive rules of various national patent regimes-in the words of WIPO, "to simplify, streamline and achieve greater convergence among national law and practice in the examination and grant of patents."[88] The SPLT and similar efforts to harmonize national patent rules could significantly reduce the costs for inventors of obtaining patent protection in multiple jurisdictions.

D. Software Patents and Innovation

A further criticism occasionally leveled at software patents is that the very existence of such patents will impede innovation. Concerns such as these, of course, are not uncommon in the face of proposals to extend IP protection to new subject matter.[89]

Patent protection invariably involves a trade-off between providing incentives for innovation today, in exchange for temporary limits on subsequent attempts to build upon such innovation. Yet there is considerable consensus that appropriately balanced patent regimes are likely to spur rather than inhibit innovation. The prospect of obtaining an exclusive right to practice and/or license an invention gives inventors important financial incentives to invest time and resources into conceiving of, developing, and marketing new technologies and products. An invention's disclosure to the public, in turn, expands the body of scientific and technological knowledge that future inventors may draw upon. And competitors who are unable to practice a patented invention will often search for new ways to improve their products or solve a problem, and this search itself can result in a further technological advance.

There is no reason to think that this phenomenon will not continue to hold true with respect to software-based technologies. Moreover, ensuring that the private sector has adequate financial incentives to invest in software research is particularly important today as government-funded research-which has traditionally been significant in the software area-becomes an increasingly smaller relative portion of national spending on R&D. Fortunately, the software industry has been relatively successful not only in inventing new technologies, but also in developing and marketing these technologies in ways that provide the resources for further rounds of innovation. Given that the U.S. PTO alone has already issued as many as one hundred thousand software patents, this success seems to be at least in part because of, rather than in spite of, the availability of patent protection for software.

CONCLUSION

Although IP laws have had an important influence on the software industry, technological advances by software developers have likewise had a strong influence on IP laws. This process of mutual influence illustrates IP laws' ability to evolve in the face of change, but has also forced software developers constantly to reevaluate how best to protect their programs against third-party appropriation. Whereas the first phase of the software industry relied primarily on trade secret and contract law, the industry's second phase has been characterized by a greater reliance on copyright law. Recent developments in the software industry, however, combined with legal developments that have clarified the reach of copyright protection with respect to piracy, follow-on copying, and decompilation, have led the IT industry to examine whether patent law might provide an important complement to copyright and trade secret law for spurring software innovation.

Several signs suggest that software patents may indeed play a positive role in promoting technological innovation. Yet as software patents continue to present examiners with relatively new technology, it is crucial that current examination practices be monitored closely, using factually supported information to assist in making needed adaptations. In this vein, many patent offices will likely need some time-and additional resources-to implement reforms and best practices, and thus to be able routinely to provide high-quality examinations for software-based innovations to the same extent such examinations are available for non-software-based inventions. In addition, all of the relevant stakeholders-including inventors, examiners, and users-should be encouraged to engage in an ongoing dialogue over patent quality to ensure that such patents balance the goals of technology innovation with the need for IT transparency and interoperability.

Initial challenges in the examination process for software patents are being addressed in the United States, and there are good reasons to believe that such challenges are surmountable elsewhere. Efforts to improve examiner training and increase patent office funding have met with some success. Contrary to conventional wisdom, references to prior art in software patents appear to be equal or superior to the norm, while ongoing efforts to further harmonize patent law and procedures hold promise in the effort to bring global patent protection within the financial reach of smaller firms.

The software industry, like the IT industry generally, has been an engine of innovation and economic growth over the past two decades. Patent laws and practices that extend equal protection to softwarebased inventions offer the best hope for keeping this engine of innovation going while promoting the broader goals of IT interoperability and technology transparency.

FOOTNOTE

[1] See, for example, Kenneth W. Dam, Some Economic Considerations in the Intellectual Property Protection of Software, 24 J Legal Stud 321, 333-38 (1995).

[2] A more complete examination of these developments is set forth in Bradford L. Smith, The Third Industrial Revolution: Law and Policy for the Internet, 282 Recueil des Cours 229, 241-46 (Martinus Nijhofi 2000). For further analyses of the software industry's development, see Dam, 24 J Legal Stud at 326-32 (cited in note 1); Stuart J.H. Graham

and David C. Mowery, Intellectual Property Protection in the US Software industry, in Wesley M. Cohen and Stephen A. Merrill, eds, Patents in the Knowledge-Based Economy 219, 220-23 (National Academies 2003).

[3] See Copyright Office Circular 31D (Jan 1965).

[4] See id. Accordingly, registration was made contingent upon the presence of authorship and deposit with the Copyright Office of a human-readable version of the program. Id.

[5] See National Commission on New Technological Uses of Copyrighted Works (CONTU), Final Report 34 (Library of Congress 1979) (dissenting opinion of Commissioner John Hersey).

[6] See id at 18 (noting that, although the doctrine of unfair competition may offer relief ancillary to copyright in certain situations, its scope is too narrow to provide sufficient protection on its own to computer software).

[7] See generally Gottschalk v Benson, 409 US 63 (1972) (rejecting a patent application for a software program on the ground that granting the patent would extend protection to an algorithm embodied in the program). See also Parker v Hook, 437 US 584, 594 (1978) (holding that the mere discovery of a natural phenomenon or mathematical formula is not patentable without some other inventive concept in its application); Dann v Johnston, 425 US 219, 220 (1976) (concluding that a "machine system for automatic record-keeping of bank checks and deposits" is "unpatentable on grounds of obviousness").

[8] See Federal Reserve Bank of Dallas 1999 Annual Report, The New Paradigm 9, 20 (2000) (describing the growth of software companies and the benefits of utilizing economies of scale).

[9] See, for example, Peter S. Mencll, Envisioning Copyright Law's Digital Future, 46 NY L Sch L Rev 63, 74 (2002) (noting concerns about the viability of trade secret protection, but recognizing a series of cases "affirming that trade secrecy protection remained viable so long as the product was distributed in a form (such as object code) that made it difficult for others to decipher its secrets").

[10] Copyright Act of 1976, Pub L No 94-553, 90 Stat 2541, codified at 17 USC 101 et seq (2000).

[11] See Act of December 31, 1974 201(b), Pub L No 93-573, 88 Stat 1873, 1874.

[12] See CONTU, Final Report at 1 (cited in note 5).

[13] See id at 30-31 (dissenting opinion of Commissioner Hersey).

[14] Id at 36.

[15] Id at 37.

[16] See id at 18-23.

[17] Id at 22.

[18] Id at 23.

[19] See id at 12-14. CONTU recommended, and Congress ultimately enacted, a limited right of "adaptation" in Section 117 of the Copyright Act in order to ensure that specified end-user modifications of software programs did not result in infringement liability. See 17 USC 117. CONTU, however, was quite clear in its intent to draw a narrow exception. In CONTU's reading of this exception, "[t]he adaptor could not vend the adapted program," and "this right of adaptation could not be conveyed to others along with the licensed or owned program without the express authorization of the owner of the copyright in the original work." CONTU, Final Report at 13 (cited in note 5).

[20] CONTU, Final Report at 16-17 (cited in note 5).

[21] Id at 17-18.

[22] "Object code" is computer code expressed as a series of 1s and 0s that can be directly executed by a computer's central processing unit. Object code is "compiled" or "assembled" from source code, which consists of human-readable program statements. Because it is nearly impossible to understand a complex software program merely by reviewing the object code, distribution of a program in object code form arguably does not disclose trade secrets embodied in the program.

[23] 714 F2d 1240 (3d Cir 1983).

[24] Id at 1249.

[25] Id at 1250-51.

[26] See International Bureau of the World Intellectual Property Organization, Model Provisions on the Protection of Computer Software 5 (WIPO 1978).

[27] Council Directive 91/250 of 14 May 1991 on the Legal Protection of Computer Programs, Art 1, 34 Off J Eur Communities (L 122) 42, 44 (May 17, 1991).

[28] See Agreement on Trade-Related Aspects of Intellectual Property Rights (Apr 15, 1994), Art 10(1), reprinted in The Legal Texts: Results of the Uruguay Round of Multilateral Trade Negotiations Annex 1C at 325 (Cambridge 1994).

[29] 975 F2d 832 (Fed Cir 1992).

[30] Id at 843-44.

[31] Id at 843.

[32] Id at 842 (internal citations omitted):

Under the [Copyright] Act, society is free to exploit facts, ideas, processes, or methods of operation in a copyrighted work. To protect processes or methods of operation, a creator must look to patent laws. An author cannot acquire patent-like protection by putting an idea, process, or method of operation in an unintelligible format and asserting copyright infringement against those who try to understand that idea, process, or method of operation.

[33] See, for example, Sony Computer Entertainment, Inc v Connectix Corp, 203 F3d 596, 602 (9th Cir 2000); DSC Communications Corp v DGI Technologies, Inc, 81 F3d 597, 601 (5th Cir 1996); Bateman v Mnemonics, Inc, 79 F3d 1532, 1539 n 18 (11th Cir 1995); Sega Enterprises Ltd v Accolade, Inc, 977 F2d 1510, 1520 (9th Cir 1992).

[34] See Council Directive 91/250, Art 6, 34 Off J Eur Communities (L 122) at 45 (cited in [note] 27).

[35] Id Art 6(1)(b)-(c).

[36] Id Art 6(2)(c) (noting that decompilation may not "be used for the development, production or marketing of a computer program substantially similar in its expression [to the decompiled program], or for any other act which infringes copyright").

[37] See Bridget Czarnota and Robert J. Hart, Legal Protection of Computer Programs in Europe-A Guide to the EC Directive 83 (Butterworths 1991) (explaining that a program developed via decompilation of other programs may only compete with those other programs if the decompilation was done for the purpose of interoperability).

[38] 797 F2d 1222 (3d Cir 1986).

[39] Id at 1233-40.

[40] Id at 1236.

[41] 982 F2d 693 (2d Cir 1992).

[42] Id at 712. See also Apple Computer, Inc v Microsoft Corp, 35 F3d 1435, 1443 (9th Cir 1994) (noting that software should not be granted patent-like protection under copyright).

[43] See Menell, 46 NY L Sch L Rev at 90-94 (cited in note 9).

[44] 740 F Supp 37 (D Mass 1990), revd, Lotus Development Corp v Borland International, 49 F3d 807 (1st Cir 1995), affd without opinion, 516 US 233 (1996).

[45] See Lotus, 740 F Supp at 63.

[46] See id at 67-70.

[47] 49 F3d 807 (1st Cir 1995), affd without opinion, 516 US 233 (1996).

[48] See id at 815-19.

[49] Id at 819 (Boudin concurring).

[50] Id.

[51] See, for example, Apple Computer, 35 F3d at 1435; Gates Rubber Co v Bando Chemical Industries, Ltd, 9 F3d 823, 834 (10th Cir 1993). See also Menell, 46 NY L Sch L Rev at 84 (cited in note 9).

[52] 450 US 175 (1981).

[53] Id at 191-93.

[54] See, for example, Julie E. Cohen and Mark A. Lemley. Patent Scope and Innovation in the Software Industry, 89 Cal L Rev 1, 8-9 (2001) (noting that Diehr and its progeny created a "doctrine of magic words" that allowed software to be patented as long as applicants called it something other than software).

[55] See In re Freeman, 573 F2d 1237, 1245 (CCPA 1978); In re Waller, 618 F2d 758, 767 (CCPA 1980); In re Abele, 684 F2d 902, 905-07 (CCPA 1982). See generally Lee E. Barrett, Patentable Subject Matter: Mathematical Algorithms and Computer Programs, 1106 Off Gaz PTO 5 (Sept 5, 1989), reprinted in 38 Pat Trademark & Copyright J (BNA) No 948 (Sept 21, 1989).

[56] 33 F3d 1526 (Fed Cir 1994) (en banc).

[57] Id at 1543 (noting that the appropriate inquiry is "whether the claim as a whole is directed to statutory subject matter").

[58] See Cohen and Lemley, 89 Cal L Rev at 10 (cited in note 54).

[59] 53 F3d 1583 (Fed Cir 1995).

[60] Id at 1584.

[61] See generally PTO, Examination Guidelines for Computer-Implemented Inventions, 61 Fed Reg 7478 (1996).

[62] For instance, since 1993, San Diego-based Thomson and Munich-based Fraunhofer Gesellschaft have together owned and licensed a series of software patents covering the mp3 digital audio compression format. See Thomson, The History of MP3, online at http:// www.mp3licensing.com/mp3/history.html (visited Dec 16, 2003). IBM has obtained more than 7,500 patents for software-related technologies since 1993. See IBM, Intellectual Property & Licensing: Patents, online at http://www.ibm.com/ibm/licensing/patents/software.shtml (visited Jan 16, 2004).

[63] Market demand for greater interoperability and product transparency has already led some software developers to provide source code access to customers and partners. See, for example, Microsoft Corp, Shared Source Licensing Programs, online at http://www.microsoft.com/ resources/sharedsource/Licensing/default.mspx (visited Dec

16, 2003) (describing Microsoft's Shared Source suite of licensing programs, which allow customers and partners to access the source code of several Microsoft operating system programs); Apple Computer, Inc, Darwin, online at http://developer.apple.com/darwin (visited Dec 16, 2003) (describing the availability of elements of Apple's OS X operating system program under the Apple Public Source license). Interest in source code access has likewise prompted a recent surge in interest in "open source" software, which is typically licensed under terms that allow users to view, and often also to modify and redistribute, the program's source code.

[64] See 35 USC 112 (2000).

[65] Moreover, the public disclosure of a patented invention, which is the quid pro quo for the patent grant, often spurs further innovation, not only through independent inventions and design-arounds, but also through licensing and complementary innovation. See Kewanee Oil Co v Bicron Corp, 416 US 470, 484 (1974). Thus, patents may offer incentives for innovation beyond those that would normally exist when software is protected only through copyright or trade secret.

[66] Considered in this way, patent exclusivity stimulates progress in the "useful arts" consistent with the constitutional mandate in US Const Art I, 8, cl 8.

[67] See Kurt Haertel, ed, European Patent Convention: Convention on the Grant of European Patents Art 52(1) (Carl Heymanns 1973) (Volker Vossius, trans).

[68] See 35 USC 103(a).

[69] Although critics of software patents sometimes characterize patent protection for software as duplicative of copyright protection, the vastly different requirements, scope of protection, and policy goals of patent and copyright law suggest that these regimes are in fact complementary. Moreover, such complementary protection is not unique to software. For instance, advances in semiconductor technology are entitled to patent protection, while the actual layout of a semiconductor is entitled to copyright-like protection as a "mask work" under the Semiconductor Chip Protection Act of 1984, Pub L No 98-620, 98 Stat 3335, 3347-55, codified at 17 USC 901-14 (2000).

[70] See, for example, 35 USC 154(a)(2); Hartel, European Patent Convention Art 63(1) (cited in note 67).

[71] 149 F3d 1368 (Fed Cir 1998).

[72] Id at 1375. Throughout the 1990s, and particularly after State Street was decided, the PTO experienced a significant increase in applications directed to software and business method patents, although more recently the number of applications for such patents has declined. Although the PTO does not maintain aggregated information regarding "software patents" (which span a wide variety of PTO classifications), it does maintain figures related to business method applications, which generally fall within Class 705. In the PTO fiscal years 2000, 2001, and 2002, respectively, 7,800, 8,700, and 6,782 Class 705 applications were filed, demonstrating that the "bubble" of applications peaked and then decreased dramatically after the fiscal year ending September 30, 2001. See PTO,

Class 705 Application Filing and Patents Issued Data, online at http://www.uspto.gov/web/menu/pbmethod/applicationfiling.htm (visited Dec 16, 2003).

[73] At least one commentator has suggested that this lack of non-patent prior art references is a result of the fact that, at least until the mid-1990s, many if not most software-based innovations were protected by trade secrets, making published references to such innovations rarer than in other fields. See Radhika Tandon, Moving Forward: Patentability of Software and Business Method Patents, 6 Intel Prop L Bull 1, 3 (2001). This premise is rejected in a more recent article, possibly indicating that the problem is ameliorating. See John R. Allison and Emerson H. Tiller, The Business Method Patent Myth, 18 Berkeley Tech L J 987 (2003).

[74] See generally Cohen and Lemley, 89 Cal L Rev at 3 (cited in note 54) (examining the implications of patent law for innovation in the software industry and arguing for refinement of traditional patent doctrine).

[75] In addition, many patents for innovations that involve software are not awarded to software developers, but rather to firms across the economy that are developing new software-based manufacturing and other methods to become more efficient and competitive. If such innovations otherwise satisfy the criteria for patent protection, the mere fact that such innovations are implemented in software does not, without more, justify denying them the benefits of patent protection. See, for example, Graham and Mowery, Intellectual Property Protection in the US Software Industry at 234 (cited in note 2) (noting that, in 1997, the one hundred largest U.S. packaged software firms accounted for less than 3.25 percent of all U.S. software patents).

[76] See Allison and Tiller, 18 Berkeley Tech L J 987 (cited in note 73).

[77] As of April 2001, the PTO had increased the number of examiners assigned to software-related applications within Class 705 to seventy-seven (from seventeen in late 1997). See Business Method Patent Improvement Act of 2001, Hearings on HR 1332 before the Subcommittee on Courts, the Internet, and Intellectual Property of the House Committee on the Judiciary, 107th Cong, 1st Sess 25 (2001) (statement of Nicholas P. Godici).

[78] See PTO, A USPTO White Paper: Automated Financial or Management Data Processing Methods (Business Methods) 21 (July 19, 2000), online at http://www.uspto.gov/web/menu/ busmethp/whitepaper.doc (visited Dec 16, 2003).

[79] Linda E. Alcorn, Pursuing Business Method Patents in the US Patent and Trademark Office, 20 Computer & Internet Law 27, 30 (Mar 2003); Hearings on HR 1332 at 52 (cited in note 77) (statement of Ronald E. Myrick).

[80] See PTO, Action Paper re Patent Quality Improvement: Expansion of the Second-Pair-of-Eyes Review, online at http://www.uspto.gov/web/offices/com/strat21/action/q3p17a.htm (visited Dec 16, 2003).

[81] United States Patent and Trademark Fee Modernization Act of 2003, HR 1561, 108th Cong, 1st Sess (Apr 2, 2003), in 149 Cong Rec H 7781 (July 25, 2003).

[82] See Patent Office Unveils Plan to Speed Up Patent Examinations, Nihon Keizai Shimbun (June 18, 2003).

[83] See Govt to Create Intellectual Property High Court by '06, Nikkei Report (Jan 15, 2003).

[84] For instance, John Allison and Emerson Tiller recently analyzed more than 1,000 Internet business method patents issued between mid-1996 and mid-1998 and compared them to a random sample of patents from fourteen discrete technology areas issued during the same period. Their research revealed that Internet business method patents had significantly more patent references, non-patent references, and total references to prior art than patents in general. See Allison and Tiller, 18 Berkeley Tech L J at 1040 (cited in note 73). The authors also found that Internet business method patents appear to have cited non-patent prior art of a similar quality to that in the average patent. Id at 1052.

[85] See USPTO White Paper at 11, 15 (cited in note 78).

[86] Robert Merges, The Uninvited Guest: Patents on Wail Street, Remarks to the 2003 Financial Markets Conference of the Federal Reserve Bank of Atlanta 27 (Apr 4, 2003), online at http://repositories.cdlib.org/cgi/viewcontent.cgi?article=1087&context=blewp (visited Dec 16, 2003).

[87] See Patent Cooperation Treaty Art 3, 28 UST 7645, TIAS No 8733 (1970).

[88] WIPO, Press Release, Member States Review Provisions on Patent Law Harmonization (May 22, 2003), online at www.wipo.int/pressroom/en/updates/2003/upd194.htm (visited Dec 16, 2003).

[89] See, for example, Tandon, 6 Intel Prop L Bull at 2 (cited in note 73).

AUTHOR_AFFILIATION

Bradford L. Smith†

Susan O. Mann‡

AUTHOR_AFFILIATION

† Senior Vice President, General Counsel, and Secretary at Microsoft Corporation.

‡ Senior Policy Counsel at Microsoft Corporation.