# NOT ALL BAD: AN HISTORICAL PERSPECTIVE ON SOFTWARE PATENTS

*Martin Campbell-Kelly*[*]

Cite as: Martin Campbell-Kelly, *Not All Bad: An Historical Perspective on Software Patents*,
11 MICH. TELECOMM. TECH. L. REV. 191 (2005),
*available at* http://www.mttlr.org/voleleven/campbell-kelly.pdf

*This Paper places the current debates about software patents in the historical context of patenting in the information technology industries.*

*The first computer-program products were sold in the mid 1960s when software patents were not generally allowed; as a result, trade secrecy became endemic to the software industry. Software products were also protected by copyright, but in practice this offered little protection against most forms of appropriation by reverse engineering or cloning. By the early 1980s a series of landmark cases led to the acceptance of software patents. It is argued that this development was consistent with the patenting of algorithmic inventions that long predated the invention of the computer. In the 1990s, business method patents were accepted. Again, it is argued that this development was consistent with the "virtualization" of inventions that long predated the Internet. It is shown that patents offer similar benefits to the software industry as for other technological industries, as well as some old and new disadvantages.*

*The Paper draws three main conclusions. First, from an historical viewpoint, software patents are not radically different from those of other technologies; the patent system has adapted to the particular demands of new technologies over time, and the software patent system is already making such adaptations. Second, patents are superior to the alternative IP regimens of trade secrecy and copyright, primarily because of the public benefits of disclosure. Third, patents offer the most economically efficient way of co-ordinating multiple R&D investments in major software technologies.*

[*] Martin Campbell-Kelly is a professor in the Department of Computer Science, Warwick University. He is a historian and computer scientist with a special interest in the history of information processing. His most recent book is FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY (2003).

Software patents are controversial. The current debate on software patents is taking place in two, largely mutually exclusive forums. First there is a lively discussion on the Internet, generally hostile to software patents, which is conducted principally by members of the open-source software community and small and medium-sized enterprises (SMEs). Second, there is a scholarly debate in the peer-reviewed academic literature, in which all shades of opinion are represented, from strongly pro patent to adamantly anti-patent.[1] These two forums represent a "digital divide" somewhat different than the usual meaning of the phrase. Most of the online debate about software patents takes place outside academia by individuals with little access to the academic literature on the subject. As a result the Internet debate is often selective, anecdotal, rhetorical, and rarely meets the standards of rigor that would be required of a peer-reviewed publication.

The principal criticism of software patents is that they are inappropriate because software is a cumulative technology, proceeding by sequential innovation. A software product typically builds on tens or hundreds of previous innovations. Opponents of software patents argue

1. For historical background and bibliography, see Stuart J.H. Graham & David C. Mowery, *Intellectual Property Protection in the U.S. Software Industry*, *in* PATENTS IN THE KNOWLEDGE-BASED ECONOMY 219, 219–58 (Wesley M. Cohen & Stephen A. Merrill eds., 2003).

that patent "thickets" will necessarily impede the flow of new software products. In 1992, Richard Stallman and Simson Garfinkle, two vociferous software patent critics, wrote, "Soon new companies will often be barred from the software arena—most major programs will require licences for dozens of patents, making them infeasible."[2]

Since this prediction, more than a decade has passed and tens of thousands of software patents have been issued.[3] However, the number of software firms—currently at least 35,000—shows no sign of diminishing.[4]

The aim of this Paper is to put software patents, and alarmist predictions, into a historical context. For example, software is far from unique as a sequential technology with strong network effects. The software and computer industries developed from the office machine industries that were established in the last quarter of the 19th century. As discussed below, not only did the office machine industry exhibit sequential innovation and network effects, it also flourished in an environment of aggressive patenting.

Although thickets are the principal concern of critics of software patents, they are not their only anxiety. There is a view that algorithms and business methods are not proper subject matter for patents. History tells us differently. Critics argue that copyright provides sufficient protection for software—again, history tells another story. It is argued that software patents are too obvious, last too long, and are over broad. There is plainly substance to these criticisms, but history suggests the solution may be adjustment rather than abandoning the benefits of patents.

This article is organized as follows. Section 1 frames the debate about software patents in the context of the more general economic arguments about the benefits and costs of patents. Some brief historical case studies show that software technologies, and the businesses built around them, have benefited from the patent system, much as other industries. Moreover, the disadvantages of patents do not appear to be more burdensome for software than for other industries. Section 2 discusses the role of patents in the early office machine industry, the most

2. Richard Stallman & Simson Garfinkle, *Viewpoint: Against Software Patents*, COMM. ACM, Jan. 1992, at 17.

3. Bryan Pfaffenberger, *The Coming Software Patents Crisis: Can Linux Survive?*, LINUX J., Aug. 10, 1999, *available at* http://www.linuxjournal.com/article/5079.

4. There are no reliable estimates of the number of software firms globally. The authors of *Secrets of Software Success* cite two estimates for the total number of firms in the software industry worldwide—one source states 35,000 firms with more than five employees, while another states 150,000 "regardless of their size." DETLEV J. HOCH ET AL., SECRETS OF SOFTWARE SUCCESS: MANAGEMENT INSIGHTS FROM 100 SOFTWARE FIRMS AROUND THE WORLD 38, 276 (1999). For further discussion on this topic, see CAMPBELL-KELLY, *supra* note *, at 12.

direct ancestor of the software industry. It is shown that patents fostered competition and diverse technologies resulting in a rich ecosystem of products from which the market could select. Further, it is shown that office technologies—like software—were sequential and cumulative, but patents did not inhibit innovation. They did, however, inhibit non-innovative makers of clone products. Section 3 discusses the origins of software protection in the 1960s, when the relative benefits and costs of patents, copyright, and trade secrecy were first addressed. Two historical case studies show how one firm opted for trade secrecy, while the other managed to secure a patent. In the first case, society learned nothing about the technology of this important product because it was maintained as a closely held secret for twenty years. In the other case, society benefited from public disclosure that facilitated the development of rival products.

Section 4 gets to the heart of software patent controversy, with a discussion of the patentability of computer algorithms. It is shown that, although algorithms were not regarded as patentable subject matter until the 1980s, in practice algorithmic devices, such as cryptographic machinery, had long enjoyed patent protection. By the 1970s devices with embedded microprocessors and software algorithms were routinely patented. Thus the decision to afford protection to pure software inventions in the 1980s was not so much a radical change as the belated recognition of an established trend. Section 5 discusses the demise of copyright protection for programs and the rise of trade secrecy. IP protection through copyright had generally been adequate for corporate software used on centralized mainframe computers in the 1960s. However, with the rise of personal computers and consumer software in the late 1970s, producers lost trust that copyright would be respected, and increasingly relied on trade secrecy. The example IBM's "object code only" policy instituted in 1983, by which it ceased to distribute source code, is described, along with the use of APIs as a substitute for source code disclosure.

Section 6 argues that a primary benefit of patents is public disclosure of inventions. Two well known patents, for the LZW data compression algorithm and the RSA cryptographic algorithm, illustrate that copyright would have provided these inventions with insufficient protection. Disclosure through patents brought far greater benefits to society than trade secrecy. Section 7 addresses current controversies about business method patents. A frequently voiced concern is that the web implementation of a real world process should not merit a patent. The example of virtual postage meters explored here shows that patents issued for Internet implementations were a logical continuation of a century of patent protection in this important industry. Moreover, patent protection encouraged new entrants into the postage meter industry, against whom the incumbents have had to compete.

Section 8 discusses the value of patents in developing broad software prospects. The examples of software for screen rendering and speech recognition are used to illustrate how these and other technological "grand challenges" are being attacked by multiple firms. Patent protection enables information to be shared among firms, so that duplicate R&D investments can be avoided. The alternative IP protection regime of trade secrecy prevents information sharing, while copyright is irrelevant in this context. Lastly, section 9 address the criticism that software patents constitute a "thicket" that impedes progress. It is shown that, relative to other important industries, the number of software patents issued is not excessive. It is argued that the current concerns about patent thickets are exacerbated by the poor state of prior art searching, the poorly developed software component industry, and the extreme fragmentation of the software industry. Time will mitigate all of these concerns.

The principal conclusions of the article are three-fold. First, from an historical viewpoint, software patents are not radically different from those of other technologies; the patent system has adapted to the particular demands of new technologies over time, and the software patent system is already making such adaptations. Second, patents are superior to the alternative IP regimens of trade secrecy and copyright, primarily because of the benefits of disclosure. Thirdly, patents offer the most economically efficient way of co-ordinating multiple R&D investments in major software technologies.

## I. BROADENING THE DEBATE—THE BENEFITS AND COSTS OF PATENTS

*An Act to promote the progress of useful Arts: The grantee or grantees of each patent shall, at the time of granting the same, deliver to the Secretary of State a specification in writing . . . which specification shall be so particular [as] to enable a workman or other person skilled in the art or manufacture . . . to make, construct, or use the same, to the end that the public may have the full benefit thereof, after the expiration of the patent term.*

—Patent Act of 1790[5]

5. Patent Act of 1790, ch. 7, 1 Stat. 109–112 (1790).

Much of the debate about software patents is essentially one dimensional, focusing on patent thickets, blocking, and the danger of inadvertent infringement. Much less is said about the benefits of patents to society. There is substantial academic literature on the benefits and costs of patents, and the economists Roberto Mazzoleni and Richard Nelson have done great service by pulling this literature together and identifying "four different, broad theories about the principal purposes patents serve":

1. Patents motivate invention
2. Patents induce disclosure and wide use of inventions
3. Patents induce the development and commercialization of inventions
4. Patents enable orderly development of broad prospects[6]

The Mazzoleni and Nelson classification offers a good trade-off between the one-dimensional debate and the highly detailed academic literature. The four theories will be explored explicitly and implicitly in this article, with a preliminary discussion below.

### A. *Patents Motivate Invention*

The motivation-of-invention theory posits that inventors will be encouraged by the temporary monopoly provided by a patent, because it will enhance their chances of profitable exploitation. For example, a temporary monopoly allows the inventors "breathing space" to mobilize resources and to undertake negotiations with parties who might help in the process. By contrast, in the absence of a patent, innovators would be discouraged because their invention, if commercially successful, would immediately be appropriated by imitators.

One of the few well documented examples of patents serving this function in the software industry is provided by Charles Ferguson, the founder of Vermeer Technologies and creator of the FrontPage web development software product.[7] In 1993, Ferguson came up with the idea of a web-authoring program—envisaged as a word-processor for writing

6. Roberto Mazzoleni & Richard R. Nelson, *Economic Theories about the Benefits and Costs of Patents*, 32 J. ECON. ISSUES 1031 (1998)[hereinafter *Economic Theories*]. Mazzoleni and Nelson are not pro patent, and their arguments are not directed toward software patents in particular. *See also* Roberto Mazzoleni & Richard R. Nelson, *The Benefits and Costs of Strong Patent Protection: A Contribution to the Current Debate,* 27 RES. POL'Y. 273 (1998) [hereinafter *Benefits and Costs*].

7. CHARLES H. FERGUSON, HIGH STAKES, NO PRISONERS: A WINNER'S TALE OF GREED AND GLORY IN THE INTERNET WARS (1999).

web pages. His concept had some novel features and he applied for three patents.[8]

Protected by his patents, Ferguson secured $4 million in venture capital and invested in the development of FrontPage.[9] Vermeer Technologies' investors had some collateral in the intellectual property of the patents and the development of FrontPage itself could take place with the security that "we were first by a wide margin and would assert our patents against everybody who followed."[10] After the World Wide Web took off in 1994, Ferguson entered negotiations with Netscape Communications and Microsoft, who were then competing in the "browser wars"[11] and each needed a complementary product for developing web pages. The patents enabled Ferguson to engage in full and open negotiations with both companies, with some security that neither would be able to imitate FrontPage without infringing on Vermeer Technologies' patents. Vermeer's FrontPage was launched in October 1995; Microsoft acquired the company and its product in January 1996 in a reported $130 million stock swap.[12]

Clearly, Vermeer Technologies benefited from the patent system. But what about its competitors? Did the patents, in fact, block other entrants and give Vermeer Technologies immunity from competition by "everybody who followed?" Seemingly not—the second half of the 1990s saw the development of numerous competitors to FrontPage.[13] One reason for the exaggerated fear of blocking is the belief that patents can foreclose an entire software category. Software patents in general (and the FrontPage patents in particular) do not typically occupy a large product space. More usually, a patent protects a unique feature or set of features of the software.[14] The question of the appropriate breadth of software patents has been well explored in the literature.[15]

8. U.S. Patent No. 5,819,092 (issued Oct. 6, 1998)(Online Service Development Tool with Fee Setting Capabilities); U.S. Patent No. 5,793,966 (issued Aug. 11, 1998) (Computer System and Computer-Implemented Process for Creation and Maintenance of Online Services); U.S. Patent No. 5,732,219 (issued Mar. 24, 1998) (Computer System and Computer-Implemented Process for Remote Editing of Computer Files).

9. FERGUSON, *supra* note 7, at 94.

10. *Id.* at 247.

11. MICHAEL A. CUSUMANO & DAVID B. YOFFIE, COMPETING ON INTERNET TIME: LESSONS FROM NETSCAPE AND ITS BATTLE WITH MICROSOFT (1998).

12. Louise Kehoe, *Microsoft Expands in Internet Software with Vermeer Purchase*, FIN. TIMES, Jan. 17, 1996, at 32.

13. Mainstream products competing with FrontPage included: HomePage by Claris, DreamWeaver by Macromedia, HotMetal by XMetaL, and several others.

14. "Most patentable inventions in computer science are not whole software programs but particular ideas or approaches to specific problems." Mark A. Lemley & David W. O'Brien, *Encouraging Software Reuse*, 49 STAN. L. REV. 255, 295 (1997).

15. *See, e.g.*, Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CAL. L. REV. 1 (2001); Robert P. Merges & Richard R. Nelson, *On the*

## B. *Patents Induce Disclosure and Wide Use of Inventions*

In the current debate about software patents, disclosure is the most under-appreciated benefit. To obtain a patent, the applicant must make a disclosure of the invention specific enough that the invention can be reproduced by a person with ordinary skill in the appropriate art (either under license or free of cost when the patent has expired). For historical reasons discussed later in this article, trade secrecy is endemic to the software industry.[16] For-profit software is typically distributed as a binary program, with the intent that users or competitors will not be able to discover its algorithms or data structures. This was not always the case.

For example, in 1957 IBM introduced the first widely adopted programming language, FORTRAN.[17] Because IBM did not at that time assert any intellectual property rights in its software, the source code was distributed to users and a detailed high-level description published in the academic literature.[18] As a result, anyone who wanted to develop a FORTRAN compiler had a clear blueprint of how to do so. Almost every computer manufacturer and many universities developed FORTRAN systems in the next few years. Compiler construction became one of the corner-stones of computer science research and teaching, and developed a substantial academic literature. By contrast, the spreadsheet—an object equally worthy of study—has always been shrouded in secrecy. Spreadsheets have never been a subject of serious scholarly study; there is no textbook literature on spreadsheet program design, nor a significant scientific literature.

Suppose VisiCalc had been patented when it was invented in 1979.[19] It is true that competitors would have had to pay a royalty to Bricklin's

*Complex Economics of Patent Scope*, 90 COLUM. L. REV. 839 (1990); Howard F. Chang, *Patent Scope, Antitrust Policy, and Cumulative Innovation*, 26 RAND J. ECON. 34 (1995).

16. Open-source software is an exception, as source code is fully disclosed. The "open source community" is the generic term for developers, who contribute to software projects such as Linux, giving their (or their employer's) time free and disclosing the code they write. For-profit open-source firms gain their income from customization and other services. In many respects this is an old model of software supply, similar to programming services in the 1950s. *See* CAMPBELL-KELLY, *supra* note *, at 29–55.

17. John Backus, *The History of FORTRAN I, II, and III*, IEEE ANNALS HIST. COMPUTING, July 1979, at 21.

18. J. W. BACKUS ET AL., THE FORTRAN AUTOMATIC CODING SYSTEM, PROC. AFIPS 1964 EASTERN JOINT COMPUTER CONFERENCE 1–5 (1964).

19. In fact Dan Bricklin took legal counsel about securing a patent, but the advice he was given was that an application would be unlikely to be successful. In retrospect, this was poor advice but of course Bricklin and his advisor did not know that the spreadsheet would be one of the cornerstone applications that started the personal computer revolution. For comments from Dan Bricklin and the origins of VisiCalc, see ROBERT SLATER, PORTRAITS IN SILICON 285–94 (1987). *See also* Daniel Bricklin, *Patenting VisiCalc*, Dan Bricklin's website, *at* http://www.bricklin.com/patenting.htm (last visited May 8, 2005).

firm Software Arts, but also, instead of reinventing the wheel each time a new spreadsheet was developed, licensees would have had access to a description of VisiCalc.[20] They would have improved on Bricklin's invention, filing their own patent for each advancement. The academy would also have been able to participate, further advancing the art. If individual entrepreneurs had been unable or unwilling to license the spreadsheet patent, then they could have devised solutions different than the VisiCalc spreadsheet, and who knows what inventions would have emerged. After all, the spreadsheet is just one solution to the problem of a generalized calculating device, but its wide acceptance and ubiquity has driven out alternative solutions.

Because a software patent was not obtained and disclosure was not made, competitors simply cloned the existing product rather than innovating. In the 1980s there were some 75 competing spreadsheets on the market, with few distinguishing characteristics.[21] At different points in time, a different product was dominant: VisiCalc from 1979 to 1983, Lotus 1-2-3 from 1984 to 1990, and Microsoft Excel from 1991 to the present.[22] What caused one product to dominate was not superior technical capabilities, but network effects—the desire of users to share files. When it was not possible to distinguish spreadsheets on their technical merits, consumers chose the next most useful attribute.[23]

### C. *Patents Induce the Development and Commercialization of Inventions*

The inducement-to-commercialize theory argues that patents encourage the refinement, practical application, and deployment of inventions which might otherwise languish for the inventor's lack of

20. It is acknowledged that the requirement for disclosure in software patents is set too low to ensure the simple recreation of an invention. Burke has advocated that there should be a requirement to append source code to a specification. This would certainly have been sufficient to recreate the original spreadsheet invention. Thomas P. Burke, *Software Patent Protection: Debugging the Current System*, 69 NOTRE DAME L. REV. 1115, 1158 (1994).

21. *See generally* ROBERT T. FERTIG, THE SOFTWARE REVOLUTION: TRENDS, PLAYERS, MARKET DYNAMICS IN PERSONAL COMPUTER SOFTWARE 177–91 (1985).

22. Quantitative data on the development of the spreadsheet market appears in STANLEY J. LIEBOWITZ & STEPHEN E. MARGOLIS, WINNERS, LOSERS AND MICROSOFT: COMPETITION AND ANTITRUST IN HIGH TECHNOLOGY 163–80 (1999). *See also* Martin Campbell-Kelly, *The Rise and Rise of the Spreadsheet*, *in* SUMER TO SPREADSHEETS: THE HISTORY OF MATHEMATICAL TABLES 322, 322–47 (Martin Campbell-Kelly et al. eds., 2003).

23. Good sources on network effects in the software industry are Brian W. Arthur, *Competing Technologies, Increasing Returns, and Lock-In by Historical Events*, 99 ECON. J. 116 (1989); Richard N. Langlois, *External Economies and Economic Progress: The Case of the Microcomputer Industry*, 66 BUS. HIST. REV. 1 (1992). More accessible accounts are Brian W. Arthur, *Increasing Returns and the New World of Business*, HAR. BUS. REV. 100 (1996); and Brian W. Arthur, *Positive Feedbacks in the Economy*, SCI. AM. 92 (1990).

funds, entrepreneurial zest, or an external agency. For example, Douglas Engelbart invented the mouse at the Stanford Research Institute (SRI) in the 1960s.[24] The mouse took several years and substantial R&D investments to bring it to market, a role for which SRI was ill-suited. Several manufacturers, including Apple Computer, Logitech, and Microsoft, developed and patented distinct devices based on the SRI foundation patent.[25] The resulting devices were as much an advance over the SRI original as a 1950s automobile was over the Model T.

Interest in the inducement-to-commercialize theory was increased by the Bayh-Dole Act of 1980, which gave universities the patent rights to inventions arising from government-funded research.[26] It was intended that the Act would facilitate the dissemination of innovations by conventional commercial channels rather than simply through academic publications and unpublished reports.[27]

An example of such inducement is the RSA cryptographic algorithm patent, assigned to MIT in 1983.[28] The patent was licensed to RSA Data Security Inc., and for a decade the firm occupied a market niche in secure communications, primarily for the financial services industry. With the explosion of interest in secure communications for e-commerce on the Internet, RSA Data Security gained an economic importance not anticipated when it was formed in 1983.[29] It was well placed to serve the e-commerce market with unique, mature, patent-protected products. (The RSA patent is discussed further in section 6 of this article.)

A related concept is that of the "inventions factory," an enterprise with specialized innovation capabilities but lacking the organizational competencies or mission to exploit them. In the world of software and business method patents, a controversial ideas factory is Walker Digital.[30] Jay Walker is the owner of the contentious reverse-auction patent, which was the concept underlying the PriceLine.com auction website.[31] Walker

24. THIERRY BARDINI, BOOTSTRAPPING: DOUGLAS ENGELBART, COEVOLUTION, AND THE ORIGINS OF PERSONAL COMPUTING (2000). U.S. Patent No. 3,541,541 (issued Nov. 17, 1970).

25. U.S. Patent No. 4,464,652 (issued Aug. 7, 1984) (Apple Computer, Inc.); U.S. Patent No. 4,951,034 (issued Aug. 21, 1990) (Logitech, Inc.); U.S. Patent No. 5,414,445 (issued May 9, 1995) (Microsoft Corporation).

26. The Bayh-Dole Act, 35 U.S.C. §§ 200–212 (1980).

27. *Economic Theories*, *supra* note 6, at 1040.

28. U.S. Patent No. 4,405,829 (issued Sept. 20, 1983).

29. The formation and early history of RSA Data Security Inc. is described in STEVEN LEVY, CRYPTO: HOW THE CODE REBELS BEAT THE GOVERNMENT—SAVING PRIVACY IN THE DIGITAL AGE 130–138 (2002).

30. Dyan Machan, *An Edison for a New Age?*, FORBES, May 17, 1999, at 178.

31. U.S. Patent No. 5,794,207 (issued Aug. 11, 1998). Observing PriceLine.com's weak financial performance since the Internet bubble burst, it is salutary to note that a patent provides a temporary monopoly, not immunity from market forces.

Digital currently claims over 200 software and business method patents.[32] There is a great deal of hostility to Walker's operation, mainly concerning the perceived low quality of business method patents, rather than Walker's concept of an innovation incubator. Inventions factories have a long and illustrious history, extending back to Thomas Edison's Menlo Park Laboratory.[33]

### D. *Patents Enable Orderly Development of Broad Prospects*

Critics of software patents frequently argue that patents block the entry of newcomers into the software field, as if this was inevitably a bad thing. In fact, such blocking can be socially desirable.

Whenever a new technology breaks there is a gold-rush period when an excess number of innovators seek to colonize the new prospect. This was pronounced in the applications software market for personal computers in the 1980s. For example, *Business Week* noted in 1984:

> At the last count, there were 200 or more word processors, 150 spreadsheets, 200 data base programs, and 95 integrated packages that offer at least three functions. Moreover, distributors report that of the 20,000 programs on the market, a mere 20 make up as much as half of their total business.[34]

The vast majority of the marketed software packages were clones of successful products, with negligible originality. The result was a massive shakeout when the gold rush frenzy subsided:

> No one expected the halcyon days of the personal computer software business to pass so quickly. Industry experts had projected that this market would continue to double annually, and 3,000 hopefuls, as a result, had jumped into the fray. But the glut of suppliers, along with the soaring cost of marketing new products and a flood of me-too programs, is changing the picture dramatically.[35]

From society's view point, such "over fishing" is economically wasteful because it means that many skilled innovators are drawn to the new prospect and are thereby removed from more socially desirable activities.[36]

32. For company information on Walker Digital LLC, see its website *at* http://www.walkerdigital.com/OurCompany.html.
33. PAUL ISRAEL, EDISON: A LIFE OF INVENTION (1998).
34. *The Shakeout in Software: It's Already Here*, BUS. WK., Aug. 20, 1984, at 103.
35. *Id.*
36. On the undesirability of over-fishing in the software context, see Merges & Nelson, *supra* note 15, at 869; Rochelle Cooper Dreyfuss, *Are Business Method Patents Bad for*

With the availability of software patents, some of this over-fishing is being eliminated. For example, there is presently a major opportunity for developing software solutions to the email "spam" problem. Although there are scores of entrants into this field, there is a great diversity of approaches and patents are being applied for by both large firms and SMEs.[37] The patent system is thus ensuring an orderly development of this broad prospect. Rather than merely cloning a successful product, an innovator must devise an original solution that does not infringe on another's patent.[38] This generates variety, and the Darwinian selection of the marketplace will ensure the survival of the fittest.[39] One counter argument sometimes asserted is that differentiating innovation in a patent application from infringement of a prior work is unreasonably daunting. This may be true, but a solution lies in the form of an improved prior art database, not the abandonment of software patents.

## II. Before There Was Software: Patents in the Early IT Industry

*In those days, it was not enough for an inventor to have a promising idea and sell it or assign it to a large company that could help him exploit it or leave him free to pursue other ideas. Ideas, even good ones, were plentiful commodities, as much so as the ribbon and broad goods that the merchant sold. The inventor had to convince someone else that his idea was practical; he had to figure out how much it would cost to prove it; how his invention could be built and financed; who would try it; and, finally, how he could repay the money needed to get it off the ground.*

—Geoffrey D. Austrian[40]

*Business?*, 16 SANTA CLARA COMPUTER & HIGH TECH. L.J. 263, 274 (2000); Jared Earl Grusd, *Internet Business Methods: What Role Does and Should Patent Law Play?*, 4 VA. J.L. & TECH. 9, 53 (1999), *available at* http://www.vjolt.net/vol4/issue/v4i2a9-grusd.html.

37. A web source lists 70 U.S. anti-spam patents granted, with many more applications in progress. Bob Wyman, *US Spam Patents: Partial List*, *at* http://www1.ietf.org/mail-archive/web/asrg/current/msg05356.html (last visited May 8, 2005). Large firms granted or applying for patents include AT&T, Microsoft, and IBM. SMEs granted or applying for patents include Spam Arrest LLC, BrightMail Inc., and Mailblocks Inc.

38. On me-too products and innovation failure, see Kenneth W. Dam, *Some Economic Considerations in the Intellectual Property Protection of Software*, 24 J. LEGAL STUD. 321, 358 (1995).

39. *See generally* RICHARD R. NELSON & SIDNEY G. WINTER, AN EVOLUTIONARY THEORY OF ECONOMIC CHANGE (1982).

40. GEOFFREY D. AUSTRIAN, HERMAN HOLLERITH: FORGOTTEN GIANT OF INFORMATION PROCESSING 20 (1982).

The patenting practices of today's old-line computer firms—such as IBM, Unisys and NCR—were established in the last quarter of the 19th century. Patents were as crucial to starting a technology-based business as finance and manufacture. When the business was established, patents protected R&D investments against appropriation by free-riders and enabled innovations to be traded with competitors. The early IT industry exhibited similar features to the software industry of today. Many of the products were developed sequentially, with significant network effects. Patenting was an instrumental part of the innovation process, rewarding innovators for their investments while promoting competition.

Today IBM is the world's most prolific investor in patents, for both computer hardware and software.[41] The company was founded in 1896 as the Tabulating Machine Company of New York by Herman Hollerith, an engineer and patent agent.[42] A graduate of the Columbia School of Mines, Hollerith's short career had included a period as an instructor at MIT, a few months in the Census Office, and a spell as a patent examiner in the U.S. Patent and Trademark Office (USPTO), Washington DC. In 1884 Hollerith "hung out his own shingle nearby as an 'Expert and Solicitor of Patents'."[43] For Hollerith, being a patent attorney was not his preferred career path, but rather a way to move into the world of invention and something to fall back on if things didn't work out. He set out to make his fortune by devising solutions to some of the critical technological problems of the era. He was doing exactly what the U.S. Patent Act of 1790 had intended. If his inventions were successful, he would secure a temporary monopoly in exchange for full disclosure.[44]

Hollerith decided to devise a system to mechanize the census.[45] In 1885, the 1880 population census was still being processed (it would take 7 years in total).[46] His idea was to use machinery to automate the then-manual counting, sorting and tabulation of census returns. He filed a patent for his census machine in 1887,[47] and in 1889 won the contract for processing the 1890 census. His invention did the job in a third of the time.[48]

41. U.S. PAT. & TRADEMARK OFF., PATENTING BY ORGANIZATIONS 2001 (2002), *available at* http://www.uspto.gov/web/offices/ac/ido/oeip/taf/topo_01.pdf.

42. A good source on the early history of the Tabulating Machine Company is SAUL ENGELBOURG, INTERNATIONAL BUSINESS MACHINES: A BUSINESS HISTORY (1976).

43. AUSTRIAN, *supra* note 40, at 20.

44. Patent Act of 1790, *supra* note 5.

45. The most comprehensive sources on the U.S. census are MARGO J. ANDERSON, THE AMERICAN CENSUS: A SOCIAL HISTORY (1988) and LEON E. TRUESDELL, THE DEVELOPMENT OF PUNCH CARD TABULATION IN THE BUREAU OF THE CENSUS, 1890–1940 (1965).

46. MARTIN CAMPBELL-KELLY, ICL: A BUSINESS AND TECHNICAL HISTORY 8 (1989).

47. U.S. Patent No. 395,781(issued Jan. 8, 1889).

48. *See* CAMPBELL-KELLY, *supra* note 46, at 8-13.

Hollerith was a consummate operator of and believer in the patent system. His patent for the census machine protected many further inventions and improvements. Many new machines were produced for the commercial market in the early 1900s, all based on the original census machine. Hollerith's patents did not in practice give him a monopoly on punched card machinery, but rather his success encouraged others to go into the punched-card business using methods that did not infringe his patents. James Powers, another inventor with experience of the U.S. census, designed a machine that was entirely mechanical rather than using Hollerith's electrical sensing and switching technologies.[49] Another successful competitor, Royden Pierce, made purpose-built punched-card machinery for corporations such as insurance companies.[50]

These competing products illustrate a benefit of patent disclosure. Because Hollerith rented his equipment it would have been difficult for a competitor to gain access to machinery and disassemble it in order to reverse engineer a competing product. Instead, competitors could read the Hollerith patents and obtain information required to invent around the original patent, creating products that were more varied and with more functions than Hollerith could have conceived alone.[51]

When Hollerith retired from The Tabulating Machine Company in 1911, it was moderately successful.[52] Under his successor, Thomas Watson Sr., it became a global powerhouse. The name was changed to the International Business Machines Corporation in 1924.[53] One of Watson's first acts on taking charge of the company was to establish an Inventions Department. He had spent his early career at National Cash Register, where Boss Kettering, arguably America's greatest inventor after Edison, had transformed the once-temperamental cash register into a mainstay of American retailing through patented innovation.[54] Watson planned to do the same for the punched card machine. IBM patented its own inventions, and acquired others it thought would be useful. For example, in 1922 IBM acquired the Royden Pierce patents (and Pierce joined IBM's Inventions Department).[55] In 1926 Watson set up a Patent Development Department under his chief inventor James W. Bryce.[56]

49. U.S. Patent No. 1,245,504 (issued Nov. 6, 1917).

50. EMERSON W. PUGH, BUILDING IBM: SHAPING AN INDUSTRY AND ITS TECHNOLOGY 44 (1995).

51. For example, Powers introduced printing tabulators and alphabetic codes long before Hollerith. CAMPBELL KELLY, *supra* note 46, at 34–37, 48–49.

52. ENGELBOURG, *supra* note 42, at 54–58.

53. *Id.* at 117.

54. For a discussion of Boss Kettering, see generally STUART W. LESLIE, BOSS KETTERING (1983).

55. PUGH, *supra* note 50, at 42–44.

56. CHARLES J. BASHE ET AL., IBM'S EARLY COMPUTERS 35 (1986).

Competition between IBM and Powers Accounting Machines intensified when the latter was acquired in a series of mergers that led to the formation of Remington Rand (the forerunner of Unisys) in 1927.[57] During the 1930s, the Hollerith and Powers lines of development constantly leapfrogged one another vying for technological superiority through patented innovations.[58] Arguably, the punched-card machine art developed much more rapidly in this competitive environment than if either firm had been able to rest on its laurels. It was the fact that inventions were patented that forced the competitors to devise original solutions rather than simply appropriate the competitor's innovations.

It is likely that many patents would have appeared "obvious" to an individual experienced in the art, and could have run into similar objections that software and business method patents encounter today. However, it is often only in hindsight that patents seem obvious. A good example of the difficulty of retrospectively assessing obviousness is illustrated by Hollerith's stop-card patent of 1914.[59] A problem with the tabulating machine at that time was that it would plow through a deck of cards, producing a total only when the entire deck had been consumed. However, it was often necessary to get subtotals at intermediate points in the deck, for example when a customer account number changed. Hollerith's patented solution was to insert "stop cards" by hand into the deck. These were ordinary, blank cards with a notch that caused the machine to stop, enabling the operator to record subtotals before restarting the machine. Trivial as this invention might seem, the related patent was the source of a significant patent dispute in Europe.[60] It reinforces the inventor's adage that the most difficult part of invention is seeing the problem; after that, devising a solution is usually relatively easy.

By the start of World War 2, IBM owned about 1400 patents in the electric accounting machine field.[61] The company employed some of America's foremost inventors, such as Clair D. Lake, James Bryce, Royden Pierce and Frederick L. Fuller.[62] In 1936, Bryce was honored by the USPTO during its centennial celebrations as one of the "ten 'greatest living inventors' ".[63] Academic journals then, as now, did not afford much of an outlet for incremental mechanical innovation. For inventors, a patent was an important recognition of their professional standing, in what

57. MARTIN CAMPBELL-KELLY & WILLIAM ASPRAY, COMPUTER: A HISTORY OF THE INFORMATION MACHINE 36 (1996).

58. Arthur L. Norberg, *High Technology Calculation in the Early 20th Century: Punched Card Machinery in Business and Government*, 31 TECH. & CULTURE 753 (1990).

59. U.S. Patent No. 1,087,061 (issued Feb. 10, 1914).

60. CAMPBELL-KELLY, *supra* note 46, at 87–90.

61. ENGELBOURG, *supra* note 42, at 136.

62. PUGH, *supra* note 50, at 77–81.

63. *Id.* at 78.

was otherwise an anonymous calling. When he was in his late seventies, Fuller published an autobiographical account of his life as an inventor, including 170 pages of descriptions and drawings from his patents.[64] Software patents serve a similar "ego-boo" role for programmers whose inventions might not merit publication in peer-reviewed academic journals.[65]

*

Typewriter and word processor innovation is a compelling example of the patent system working at its best: it fostered innovation and competition, leading to stupendous improvements in a relatively short period of time. The early typewriter industry was considerably more competitive than punched card machine manufacturing, because the market was much bigger. Some 140 U.S. firms, and 400 worldwide, fought for market share around the turn of the 19th century.[66] The Remington Typewriter Company was the first mover in the typewriter industry.

The classic QWERTY typewriter was patented by its inventor Christopher Latham Sholes in 1868.[67] Incidentally, the patent did not protect the QWERTY keyboard arrangement. The persistence of the QWERTY keyboard is perhaps the most cited example in network economics.[68] Had this arrangement, providing the user interface for Sholes's typewriter, been protected by a copyright regime analogous to that which protected software user interfaces in the 1980s, universal adoption of the QWERTY interface may not have been possible (see section 5 *infra*).

Mechanical writing was a keenly contested area. Sholes was the 52nd individual to file a typewriter patent, but the first to invent a ma-

64. FREDERICK L. FULLER, MY HALF CENTURY AS AN INVENTOR (1938).

65. "Ego-boo" has not yet made the conventional dictionaries. Its sense is conveyed in the following usage, "Hackers operate in a gift economy in which giant-size egos compete with one another for attention and reputation on the Net. If you do something cool, like reduce the length of a subroutine by 50 percent, you score major egoboo." Mark Frauenfelder, *Man Against the FUD*, L.A. WKLY, May 21, 1999.

66. *See* WILFRED A. BEECHING, CENTURY OF THE TYPEWRITER (1974); BRUCE BLIVEN JR., THE WONDERFUL WRITING MACHINE (1954); G. TILGHMAN RICHARDS, THE HISTORY AND DEVELOPMENT OF TYPEWRITERS (1964); George Nichols Engler, The Typewriter Industry: The Impact of a Significant Technological Revolution (1969) (unpublished Ph.D. dissertation, University of California at Los Angeles) (available from UMI Dissertation Publishing).

67. U.S. Patent No. 79,265 (issued Jun. 23, 1868).

68. *See, e.g.*, Paul A. David, *Understanding the Economics of QWERTY: The Necessity of History*, *in* ECONOMIC HISTORY AND THE MODERN ECONOMIST 30, 30–49 (William N. Parker ed., 1986); JAMES M. UTTERBACK, MASTERING THE DYNAMICS OF INNOVATION 5–7 (1994); Stanley J. Liebowitz & Stephen E. Margolis, *The Fable of the Keys,* 33 J.L. & ECON. 1 (1990).

chine the world actually wanted.[69] When Sholes needed money for improvements, he secured what we would now call "angel funding" from James Densmore, a retired and well-to-do printer, in exchange for a quarter share of the patent.[70] Sholes would take another five years and build 50 models before the machine was ready for manufacture.[71] The value of the patent to Sholes was thus to turn his invention into property that could be exchanged for funding, and the temporary monopoly provided "breathing space."[72]

The capital requirements for manufacture were far beyond Densmore's pocket, so Densmore approached Philo Remington, a manufacturer of small arms. The Remington Company was struggling since the end of the Civil War, and the typewriter offered a chance to put its idle plant to work. The first Remington typewriter went on sale in 1874,[73] but it was a slow seller. The worst of many imperfections was that the typist could not see what he or she had typed because the type-bars printed on the underside of the carriage. This defect was eliminated in one of the most important typewriter patents of all time, Frank X. Wagner's front-strike patent granted in 1893 and assigned to the Underwood Corporation.[74] With the Underwood patent, the typewriter took its modern form. The Underwood No. 5 "visible" typewriter went on sale in 1899. Protected by Wagner's and subsequent patents "by 1920 Underwood's sales of Model 5 were equal in quantity to all of the other firms in the typewriter industry combined".[75] Underwood's competitors responded with their own patented improvements. Besides variants of visible typing, innovations included such items as "noiseless" operation, tab settings, multicolor ribbons, and a lever operated paper feed.

We do not know much detail of the interaction between the typewriter firms at this distance in time. We do know, however, that a great deal of cross-licensing of patents took place, generally resulting in a common look and feel for all typewriters. More than a hundred firms came and went in the U.S. during the formative years of the industry, leaving only a handful of major firms that included Underwood, Remington, L.C. Smith, and Royal.[76] To be a member of the typewriter elite, a firm needed to have patents to trade with the others; creating patented

69. BLIVEN, *supra* note 66, at 42; VICTOR M. LINOFF, TYPEWRITER TOPICS, THE TYPEWRITER: AN ILLUSTRATED HISTORY, at v (2000).
70. BLIVEN, *supra* note 66, at 48.
71. *Id.* at 49.
72. On breathing space, see Merges & Nelson, *supra* note 15, at 871.
73. RICHARDS, *supra* note 66, at 24.
74. U.S. Patent No. 559,345 (issued Apr. 28, 1896).
75. Engler, *supra* note 66, at 30.
76. The shakeout in the typewriter is quantitatively described in UTTERBACK, *supra* note 68, at 33–34.

innovations was how they stayed in the game. Rather like software, every typewriter model contained dozens of patented innovations that could potentially hold up its production, but there is no evidence that such blocking took place.[77] The firms that were shaken out included imitators and free-riders that had no innovations to trade,[78] as well as those that failed to develop effective production, sales, and service organizations.[79]

By 1910 (when the typewriter was not much older than the personal computer is today) there were 2600 patents in the typewriter class, and it was a mature reliable product that had transformed the American office.[80] There was relatively little product innovation after this date; instead manufacturers competed on cost, reliability, quality, service operations, and customer training.[81] For example, Remington excelled in service operations while Royal was noted for the quality of its perfectly aligned type and elegant typefaces. In the 1930s the typewriter firms were a prosperous, stable oligopoly. They were huge vertically integrated operations employing tens of thousands of workers. The existence of this oligopoly did not exclude innovative small firms from participation. The success of the industry spawned a strong aftermarket for complementary products, as simple as carbon paper and as complex as adding and totalizing attachments.[82]

The industry changed significantly in 1933, when IBM paid $1 million to acquire the Electromatic Typewriter Company of Rochester, New York.[83] IBM refined the technology—patenting its many improvements—and brought the first successful electric typewriter to market in 1935.[84] IBM enjoyed significant success with this typewriter when of-

77. The situation is analogous to that of patent cross-licensing in the semiconductor and software industries today. *See* JAMES BESSEN & ERIC MASKIN, SEQUENTIAL INNOVATION, PATENTS AND IMITATION (MIT Department of Econ., Working Paper No. 00-01, January 2000); CARL SHAPIRO, NAVIGATING THE PATENT THICKET: CROSS LICENSES, PATENT POOLS, AND STANDARD-SETTING (Competition Policy Center, Paper CPC00'011, 2000).

78. A typical free-riding firm, which did not last long, was the Manhattan Typewriter Company of New York, which made a clone of the Remington No. 2 in 1898 for which "[e]xpired patents were chiefly utilized." LINOFF, *supra* note 69, at 43.

79. ALFRED D. CHANDLER, THE VISIBLE HAND: THE MANAGERIAL REVOLUTION IN AMERICAN BUSINESS 277–78, 308 (1977).

80. BLIVEN, *supra* note 66, at 102.

81. *See generally* JAMES W. CORTADA, BEFORE THE COMPUTER: IBM, NCR, BURROUGHS, AND REMINGTON RAND AND THE INDUSTRY THEY CREATED, 1865–1956 (1993).

82. For an impression of the "incredible range" of office equipment available in the 1920s, see JoAnne Yates, *Business Use of Information and Technology during the Industrial Age*, *in* A NATION TRANSFORMED BY INFORMATION 107, 125-26 (Alfred W. Chandler Jr. and James W. Cortada eds., 2000).

83. ENGELBOURG, *supra* note 42, at 252–53; H.S. Beattie & R.A. Rahenkamp, *IBM Typewriter Innovation*, 25 IBM J. RES. & DEV. 729 (1981).

84. BEECHING, *supra* note 66, at 123.

fices re-equipped following World War 2. This led to another cycle of product innovation as the incumbent manufacturers brought out electric models to compete with IBM. But none of the firms made a successful transition; the old industry was decimated and IBM came to dominate the typewriter market as surely as Underwood had done half a century earlier.

In the 1960s, another wave of innovation in machine writing started with the word processor. IBM, which invented the term "word processing," was the dominant player in the field, although its products were pedestrian evolutionary developments of its standard Selectric typewriter.[85] In 1976 a newcomer to word processing, Wang Laboratories, introduced a CRT-based system—"a patented design that would win much praise"[86]—which enabled users to edit and view documents on a screen before printing them.[87] The Wang Word Processing System (WPS) was hugely successful, primarily because of its software-driven, user-friendly interface. It soon enjoyed a dominant market share. Of course, IBM and other word processing firms soon invented around one another's patents producing a rich ecology of competing systems.[88] By the mid-1970s, the cumulative total of typewriter and word processing patents exceeded 17,000.[89]

Our own generation's wave of machine-writing innovation began with the rise of the personal computer and word processing software around 1980.[90] There was no significant patenting activity associated with word processing software; this, combined with the low entry barriers, encouraged several hundred firms into the market. In 1985 there were 300 packages for the IBM-compatible PC alone;[91] almost all were "me-too" clones of existing products. This was a classic case of overfishing and within a decade the number of firms had been reduced to perhaps a score.[92]

85. F.T. May, *IBM Word Processing Developments*, 25 IBM J. RES. & DEV. 741 (1981).

86. CHARLES C. KENNEY, RIDING THE RUNAWAY HORSE: THE RISE AND DECLINE OF WANG LABORATORIES 69 (1992).

87. For histories of the Wang WPS word processing system, see KENNEY, *supra* note 86 at 63–77; and AN WANG & EUGENE LINDEN, LESSONS: AN AUTOBIOGRAPHY 171–187 (1986). The WPS patent is U.S. Patent No. 4,145,739 (issued Mar. 20, 1979). The patent is a typical process-and-apparatus software-related patent of the type discussed in section 4.

88. Amy D. Wohl, *What's Happening in Word Processing*, DATAMATION, Apr. 1977, at 65.

89. Search conducted by the author on the USPTO website (http://www.uspto.gov/patft/index.html) for Class 400 Typewriting Machines, 1790–1975.

90. PAUL FREIBERGER & MICHAEL SWAINE, FIRE IN THE VALLEY: THE MAKING OF THE PERSONAL COMPUTER 147–48, 152–53 (1984).

91. FERTIG, *supra* note 21, at 164.

92. On over-fishing, see Merges & Nelson, *supra* note 15, at 869; Grusd, *supra* note 36, at 53; Dreyfuss, *supra* note 36, at 274.

## III. THE FIRST SOFTWARE PRODUCTS

*"Productization" of software posed a completely new set of considerations and problems for Informatics. No one had sold a software product before. How much should the product be sold for? . . . Should the software be sold outright allowing purchasers to do whatever they wanted with their purchase copy, except resell it? How can unauthorized duplication of the system and transfer of it to others be prevented?*

*While such questions have since been answered numerous times by many software companies, in 1967 these were totally new and unanswered questions for the embryonic industry.*

—Richard L. Forman[93]

IBM introduced its first electronic computer, the model 701, in 1953.[94] The term software had not yet been invented (it came into use about 1960)[95] and the programs that IBM supplied for the 701 consisted of only a few hundred lines of code—a tiny fraction of the amount one would get with a domestic PC today.[96]

Neither IBM nor any other computer manufacturer took any steps to protect the intellectual property of its programs. IBM made a policy decision that computer programs and procedures were not patentable.[97] Nor did IBM assert copyright in its programs because it was unclear that such an assertion would have any validity.[98] The lack of concern for intellectual property in software may seem surprising, but as late as 1970 manufacturer-supplied programs accounted for only about 3 percent of the cost of a computer.[99] There was little economic incentive to press for an appropriate IP regime for software protection.[100]

93. RICHARD L. FORMAN, FULFILLING THE COMPUTER'S PROMISE: THE HISTORY OF INFORMATICS 1962–1968, at 9/18 (1985).

94. BASHE ET AL., *supra* note 56, at 163.

95. Ivars Peterson, *Software's Origin*, SCI. NEWS ONLINE, Jul. 29, 2000, *at* http://www.sciencenews.org/articles/20000729/mathtrek.asp.

96. BASHE ET AL., *supra* note 56, at 332.

97. PUGH, *supra* note 50, at 226.

98. *See* ROBERT V. HEAD, A GUIDE TO PACKAGED SYSTEMS 123 (1971).

99. Watts S. Humphrey, *Reflections on a Software Life*, *in* IN THE BEGINNING: PERSONAL RECOLLECTIONS OF SOFTWARE PIONEERS 29–53 (Robert L. Glass ed., 1998).

100. In 1965 James Birkenstock, vice-president of IBM, asserted the company's continuing lack of interest in software patents at the presidential commission on the working of the patent system. *See* Pamela Samuelson, *Benson Revisited: The Case against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 EMORY L.J. 1025, 1038 (1990).

In the mid-1950s, about a dozen firms entered the programming services industry, writing programs to order for clients.[101] Here again, software protection was not a significant issue because programs written under contract for an organization were so particular that they would have had little value to another organization. Moreover, in an era when computers cost several hundred thousand dollars a year to rent, the high programming cost was buried under the overall cost of computer ownership.[102]

The situation changed in the mid 1960s with the arrival of the IBM System/360 computer. The System/360 was extremely successful, creating for the first time an industry standard platform.[103] At the same time, the computer population had begun to explode—from 5,500 worldwide in 1960 to 29,600 by 1965, an annual growth rate of 30 percent.[104] Falling hardware prices had created a new sector of the computer market of corporations paying annual rentals of as little as fifty thousand dollars.[105] For these new owners, custom-written programs were not economically justifiable. Fortunately, the new generation of computers had much greater speeds and larger memories than previous models, allowing the inefficiencies of generalized software "products" to be tolerated rather than requiring development of more efficient, custom-written programs. For software firms, the large customer base of System/360 users made writing such generalized programs a viable business proposition.

Hence, the second half of the 1960s saw the first program products from independent software vendors.[106] Software products needed two kinds of intellectual property protection, for which the existing mechanisms of trade secret, copyright and patent each offered a degree of cover, although none was wholly satisfactory. First, the functional aspects and source code of a program needed to be protected from appropriation by a competitor that might plagiarize them to create a competing product. Second, packaged programs needed to be protected from unauthorized copying by a user organization that wanted to make

101. Two good business histories of programming services are CLAUDE BAUM, THE SYSTEM BUILDERS: THE STORY OF SDC (1981) and FORMAN, *supra* note 93. See also CAMPBELL-KELLY, *supra* note *, at chs. 2 & 3 for a history of software contractors.

102. Frederick P. Brooks Jr., *No Silver Bullet: Essence and Accidents of Software Engineering*, COMPUTER, Apr. 1987, at 10.

103. *See generally* EMERSON W. PUGH ET AL., IBM'S 360 AND EARLY 370 SYSTEMS (1991).

104. MONTGOMERY PHISTER JR., DATA PROCESSING: TECHNOLOGY AND ECONOMICS 251 (2nd ed. 1979) (1978).

105. FRANKLIN M. FISHER ET AL., IBM AND THE U.S. DATA PROCESSING INDUSTRY: AN ECONOMIC HISTORY 105 (1983).

106. Luanne James Johnson, *A View from the 1960s: How the Software Industry Began*, IEEE ANNALS HIST. COMPUTING, Jan.–Mar. 1998, at 36. The early history of the software products industry is described in CAMPBELL-KELLY, *supra* note *, at 89–120.

use of the program but did not wish to pay for it. This second copy protection is aimed at what is now known as piracy.

Two early software products provide examples of the differing ways in which companies used intellectual property to secure their property rights. Furthermore, these examples suggest that patents helped create rather than diminish sequential innovation. Mark IV, a file management system, utilized the protection regimes of trade secrecy and contract law; Applied Data Research Inc. (ADR) obtained a patent for its flowcharting package, Autoflow. Using contract and trade secrecy meant that all information about Mark IV was kept out of the public sphere—competitors were forced to start from scratch. With Autoflow, securing a patent meant that information about the inner-workings of Autoflow was available to be built upon and used by competitors.

The Mark IV file management system was one of the most important products in the early history of the software industry.[107] Because of its early introduction, it shaped many of the practices of the industry—not least its approach to protecting intellectual property. The program was created by the Informatics Corporation in 1967.[108] A "file management system" was an early form of database, supplemented by a complementary suite of programs for file maintenance and report generation. Mark IV was a very important product because for the first time it provided users with a packaged system which they could use to run a substantial part of their business. Early users included firms such as Prudential and Sun Oil.[109] Following its launch in 1967 it was the world's top-selling program for 15 years.[110] When it peaked in 1983 it had generated cumulative revenues exceeding $100 million and had several thousand users worldwide.[111]

However, back in 1965, when Mark IV was still on the drawing board, software was seen as a free good. A major cultural shift was needed for users to accept and respect the concept of software as intellectual property. To protect the anticipated $500,000 development costs of Mark IV[112], Informatics' president Walter F. Bauer considered the three options: patents, copyright, and trade secrecy.

First, Informatics applied for a patent. However, in 1965 it was generally held that programs could not be patented because the U.S. patent

107. Walter F. Bauer, *Informatics: An Early Software Company*, IEEE ANNALS HIST. COMPUTING, Apr.–Jun. 1996, at 70; John A. Postley, *Mark IV: Evolution of a Software Product, a Memoir*, IEEE ANNALS HIST. COMPUTING Jan.–Mar. 1998, at 43.

108. FORMAN, *supra* note 93, at 9/20.

109. *Id.*

110. Bauer, *supra* note 107, at 74–75.

111. Mark IV revenues are summarized in CAMPBELL-KELLY, *supra* note *, at 117.

112. FORMAN, *supra* note 93, at 9/15.

system explicitly excluded mathematical laws (and hence computer algorithms) as patentable subject matter.[113]

The second option was copyright protection. In 1964, the U.S. Copyright Office had accepted that programs could be afforded copyright protection under a "rule of doubt," provided that a human-readable copy of the program source code was deposited in the Office.[114] However, this would have enabled a competitor to inspect the code of the program and reverse engineer a clone. In this context, reverse engineering would mean to study the source code in order to understand how it worked, transcribe its data structures and formats, and then write a program with the same behaviors.[115] This would have been completely legal. However, writing the source code represented only a fraction of the $500,000 development cost of Mark IV. Most of the money went to defining the system and its data formats, testing trial versions with users, reversing product decisions that did not work out, and creating a demand for the product through sales and marketing efforts. A competitor developing a product through reverse engineering of Mark IV would only face the cost of writing the source code; they could produce a functional replacement of the product for a fraction of Informatics' development cost. For Bauer in 1965, this risk was unacceptable.[116]

Thirdly, and largely as a last resort, Informatics decided to use trade secrecy and contract law to protect Mark IV. Customers were not sold the program, but rather were granted a license to use it, which incorporated a non-disclosure agreement. The program and all the associated documentation remained the property of Informatics, and customers were forbidden to make any disclosures of the program or documentation to a third party. Further, Informatics' own workers were bound by an employment contract that forbade them from disclosing Mark IV knowledge to third parties, or transferring know-how or code to another employer.[117]

113. Mark IV did, however, subsequently gain patent protection in Canada and the UK. *Id.* at 9/19.

114. Milton R. Wessel, *Legal Protection of Computer Programs*, HARV. BUS. REV., Mar.–Apr. 1965, at 97, 103, citing *Copyright Registration for Computer Programs, May 19, 1964*, 11 BULL. CR. SOC. 361 (1964).

115. For an analysis of the many techniques of reverse engineering, see Andrew Johnson-Laird, *Software Reverse Engineering in the Real World*, 19 U. DAYTON L. REV. 843 (1994).

116. For an extended discussion on the inadequacies of copyright protection for software, see Pamela Samuelson et al., *Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994).

117. FORMAN, *supra* note 93, at 9/19.

ADR took a very different approach to protecting its product, Autoflow.[118] In the 1960s, almost all programming shops required programmers to document programs with a flowchart—a graphical representation of the logical flow of the program. Flowcharting was often the last, and most irksome, task of a programmer before moving on to the fresh field of a new assignment. Consequently, flowcharts often did not get drawn, and maintenance costs increased. Martin Goetz, cofounder of ADR, designed Autoflow to produce flowcharts effortlessly by reading through a user's source program and from it automatically generating and printing a neatly formatted flowchart. It was a tour de force of programming that even today is an impressive piece of coding. The system cost about $10,000 to develop, much more to promote; it paid off, however, as it went on to sell several thousand copies.[119]

In 1968, Goetz applied for a patent for the program; in 1970, it was one of the earliest software product patents granted.[120] Goetz had been able to take advantage of an August 1966 advisory by the U.S. Patent and Trade Mark Office (USPTO) that "a patent could be granted to a program if it could meet the requirements of either a 'process' or an 'apparatus'."[121] Accordingly, the Autoflow program was presented as a machine for achieving a particular kind of information transformation. Just as a pin-making machine would have transformed steel wire into pins, so the Autoflow "machine" transformed source card decks into printed charts.

In accordance with society's patent bargain, in exchange for full disclosure of the invention, ADR obtained a 17-year monopoly for Autoflow after which anyone was free to make use of it without permission or charge. ADR's patent attorney took disclosure seriously. The patent specification included a complete 50-page listing of the program. Furthermore, Autoflow was used to produce a complete flowchart of itself—an example of recursion guaranteed to warm the heart of any computer scientist. Because of the thorough disclosure required by patent law, contemporary competitors were able to study Autoflow, understand its algorithms, improve or design around them, and produce competing products which the market could accept or reject.

Critics of software patents often claim the "competition through improvement" argument is spurious because there is usually only one best

118. For a short history of ADR, see Don Leavitt, *A Silver Anniversary in a 15-Year-Old World,* SOFTWARE NEWS, Jul. 1, 1984, at 38.

119. *Id.*

120. U.S. Patent No. 3,533,086 (issued Oct. 6, 1970).

121. Robert B. Bigelow, *Legal Aspects of Proprietary Software*, DATAMATION, Oct. 1968, at 32–39. For a brief history of "tricks of the trade" for process-and-apparatus patents, see Burke, *supra* note 20, at 1151–54.

way to write an algorithm and a patent forecloses it to others.[122] However, the Autoflow patent did not create a barrier preventing the entry of competitors. By 1972, there were at least four competing products.[123] It is not known whether or not any of the products infringed the ADR patent. Because none of the products was as successful as Autoflow, the question probably never arose.

The Autoflow example shows the patent system working as it was intended. In exchange for a temporary monopoly, ADR disclosed its invention allowing the knowledge to diffuse into the software writing community. Competitors were free to do their best to improve on the invention. Contrast this with Mark IV, protected vigorously by trade secret. Assuming no infringement took place, each of the half-dozen Mark IV competitors had to build from the ground up, learning little from the others or Informatics' prior experience.[124] It was like six desert island companies.[125]

## IV. ALGORITHMS, SOFTWARE PATENTS AND THE VIRTUALIZATION OF INVENTIONS

*The players move from space to space in accordance with the throw of the dice and hence become subject to rental charges if they land upon property held by other players. If unable to meet a rental charge and the player concerned is unable to raise money by disposing of any property which he holds, to the Bank or other players in accordance with the rules of the game, he is*

122. For example, one software-patent critic states, "functional claiming, combined with the virtual nature of software, is an absolute bar to the development of any functionally equivalent product." Russell Moy, *A Case against Software Patents*, 17 SANTA CLARA COMPUTER & HIGH TECH. L.J. 67, 99 (2000). In contrast, John Swinson, a legally trained computer scientist, states, "a given result can generally be reached by more than one program." John Swinson, *Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection,* 5 HARV. J.L. & TECH. 145, 149 (1991).

123. The competing products were Dynachart by Applications Programming Co., Autodoc by Data Instrument Co., Quick-Draw by National Computer Analysts, and FCP by World Computer Corp. ADR's Autoflow cost the most at $7,370, while the others cost $1,500 upward. The IBM Program Application Library also included a free flowcharting program, Autodoc-V. *See* Ned Chapin, *Flowchart Packages and the ANSI Standard*, DATAMATION, Sept. 1972, at 48–49.

124. Donald B. Steig, *File Management Systems*, DATAMATION, Oct. 1972, at 48–51; Larry Welke, *A Review of File Management Systems*, DATAMATION, Oct. 1972, at 52–54.

125. It should be noted, however, that the level of disclosure in the ADR Autoflow patent—which included a full source code listing—was much greater than that of most current patents. Although current disclosure requirements are set too low, the reform of the disclosure rules is a better solution than driving software firms back into trade secrecy. *See supra* note 20.

*considered bankrupt and must quit the game, taking up his symbol or token from the board.*

—Monopoly Patent, 1935[126]

The history of software patents is bracketed by two landmark decisions: *Gottschalk v. Benson* in 1972 and *Diamond v. Diehr* in 1981.[127] Both addressed the issue of whether or not a computer algorithm constituted patentable subject matter. After *Gottschalk*, the patent environment was unfavorable to software patents; after *Diamond* it became broadly favorable.[128] However, between these two decisions, there were major technical and business developments—the invention of the microprocessor, the widespread use of embedded microprocessors, and the rise of the personal computer. Domestic sales of U.S. software products grew from $440 million to over $4 billion.[129]

*Gottschalk v. Benson* concerned a patent application by Bell Telephone Laboratories, the research arm of AT&T.[130] Bell Labs was one of the first companies to advocate software patenting. Bell Labs had already been granted several patents when an application by two of its researchers, Gary Benson and Arthur Tabbot, was rejected in 1968.[131] The Benson-Tabbot invention, filed in 1963, was a means of converting Binary Coded Decimal (BCD) numbers to ordinary binary numbers.[132] Although most of the patent was expressed in process-and-apparatus form, one of the claims related to the algorithm itself. The algorithm could have been executed by any general purpose computer or even using pencil and paper. Moreover, the algorithm could have been

126. U.S. Patent No. 2,026,082 (issued Dec. 31, 1935).

127. Gottschalk v. Benson, 409 U.S. 63 (1972); Diamond v. Diehr, 450 U.S. 175 (1981). The histories of *Gottschalk v. Benson* and *Diamond v. Diehr*, and the many cases in between, have been the subject of many articles. A good bibliography is given in Burke, *supra* note 20.

128. Good histories of this period in software patent history which convey something of the political and economic as well as the legal environment are Burke, *supra* note 20; and Samuelson, *supra* note 100.

129. CAMPBELL-KELLY, *supra* note *, at 14.

130. Concise technical and legal histories of *Gottschalk v. Benson* appear in HOWARD W. BROCKMAN, INTELLECTUAL PROPERTY LAW FOR ENGINEERS AND SCIENTISTS 215–16 (2004); H. W. A. M. HANNEMAN, THE PATENTABILITY OF COMPUTER SOFTWARE: AN INTERNATIONAL GUIDE TO THE PROTECTION OF COMPUTER-RELATED INVENTIONS 51–54 (1985); GREGORY A. STOBBS, SOFTWARE PATENTS 286–90 (1995).

131. *In re* Benson, 441 F.2d 682 (C.C.P.A. 1971).

132. For an authoritative history of BCD numbers, see DONALD E. KNUTH, THE ART OF COMPUTER PROGRAMMING, VOL. 2: SEMI-NUMERICAL ALGORITHMS 169 (1969). However, it is not necessary to understand or know the notational details of BCD and binary numbers, or the conversion algorithm itself, to follow the legal argument.

considered a "law of nature".[133] The patent was rejected by the examiner because the algorithm constituted "non-statutory subject matter."[134] Bell Labs took the case to the Court of Customs and Patent Appeals, which reversed the decision on the grounds that the process "had no practical use other than the more effective operation and utilization of a machine known as a digital computer" and the Court saw "no sound reason why the claims in this case should be held non-statutory."[135] Finally, the Commissioner of the Patent Office, Leonard Gottschalk, appealed to the Supreme Court for a writ of certiorari. The Supreme Court reversed again, stating that the claim was "so abstract and sweeping as to cover both known and unknown uses" and "would wholly pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself."[136] In effect, this decision ruled out "pure" software patents but left the door open for software enabled inventions that produced a new, useful, and non-obvious technical effect.[137]

From a historical perspective, the patenting of algorithms has never been a black-and-white issue. For example, a patented board game such as Monopoly embodies a set of rules and procedures that a computer scientist would consider to be program-like.[138] Indeed, board games are sometimes used as a pedagogical device to get across procedural programming concepts.[139] Today, Monopoly is sold as a computer game—a pure software artifact.[140]

A cryptographic apparatus is another example of an embodied algorithm. Landmark inventions include the Vernam cryptographic machine (1919), the Swedish Enigma (1923), the U.S. Sigaba (1930), and the British Typex (1935).[141] The Enigma used a scrambling unit that converted

133. That is, if BCD numbers could be said to exist in nature. This would depend on whether one subscribed to the doctrine that "God made the integers; all else is the work of man." Leopold Kronecker, quoted in E.T. BELL, MEN OF MATHEMATICS 477 (1986).

134. *In re* Benson, *supra* note 131, at 1137.

135. Cited in STOBBS, *supra* note 130, at 289.

136. *Id.*

137. The term "technical effect" is predominantly used within the European Community; the United States does not appear to have such a neat encapsulation. *See* REINIER BAKELS & P. BERNT HUGENHOLTZ, THE PATENTABILITY OF COMPUTER PROGRAMS: DISCUSSION OF EUROPEAN-LEVEL LEGISLATION IN THE FIELD OF PATENTS FOR SOFTWARE (Eur. Parl. Directorate-Gen. for Research, Working Paper No. JURI 107 EN, 2002).

138. U.S. Patent No. 2,026,082, *supra* note 126.

139. *See, e.g.*, M.A. Storey et al., *How Do Program Understanding Tools Affect How Programmers Understand Programs?* 36 J. SCI. COMPUTER PROGRAMMING 183 (2000).

140. Parker Bros., the original owners of Monopoly, is a subsidiary of Hasboro, Inc. The Monopoly computer game is manufactured by Hasboro Interactive.

141. The best accessible technical account of non-electronic cryptographic apparatus is DAVID KAHN, THE CODE-BREAKERS: THE STORY OF SECRET WRITING 394–434 (1967). Kahn makes good use of the patent literature. A more recent treatment is the historical overview in DONALD W. DAVIES & WYNN L. PRICE, SECURITY FOR COMPUTER NETWORKS: AN

plain text keyed by the operator into an encrypted message. When the encrypted message was keyed into a second machine using the same "key", the process was reversed, revealing the original text. The information transformation performed by the Enigma was algorithmic, even though it was achieved by a set of rotating wheels and electrical connections. Indeed, the Enigma was broken by the Allies in World War 2 when they recognized that it was an algorithmic device, and used mathematics and proto-computers to attack it.[142] Today, software Enigma simulators have been written by computer hobbyists and amateur cryptographers.[143]

There are two reasons why patents on such algorithmic devices were allowed. First, the algorithms and the hardware implementations were so commingled that the machinery itself passed the tests of novelty and non-obviousness. Second, patent examiners were not familiar with the concept of an algorithm as a static description of a dynamic process, and had no vocabulary for it. Although patent examiners lacked the conceptual background to recognize algorithms, the Patent Office was granting algorithmic patents.

The effect of the *Gottschalk v. Benson* result was to discourage applications for pure software inventions—inventions that could form the basis for a software product, independent of a particular hardware configuration—in the 1970s.[144] Many patents were granted for physical inventions that incorporated algorithms and programs. This was especially common when inventors replaced electro-mechanical components in existing products with embedded microprocessors to improve the functionality of the manufactured artifact.

For example, the 1970s saw electro-mechanical carburetors in automobiles replaced by microprocessor-controlled fuel injection systems. Electro-mechanical pin-ball machines were augmented with programmed microprocessors. Controllers for elevators, once directed by electro-mechanical relays, were computerized. Automatic braking systems, electronic typewriters, microwave ovens, computing scales, videogames—the list of software-related inventions patented between

INTRODUCTION TO TELEPROCESSING AND ELECTRONIC FUNDS TRANSFER 30–39 (1989). U.S. Patents for the cited machines are as follows. The Vernam machine: U.S. Patent No. 1,310,719 (issued Jul. 22, 1919); the Enigma: U.S. Patent No. 1,657,411 (issued Jan. 24, 1928); the Sigaba: U.S. Patent No., 2,028,772 (issued Jan. 28, 1936)—subsequent patents were not granted until recent times due to secrecy.

142. ANDREW HODGES, ALAN TURING: THE ENIGMA 160–241 (1983).

143. For a listing of several Enigma and other simulators, see Enigma simulator index page, *at* http://frode.home.cern.ch/frode/crypto/simula/index.html (last visited May 8, 2005).

144. For example, as late as 1985 ADAPSO's manual on software protection commented on software patent applications "If you *still* want to go ahead, your first step will bring a hostile response." ERNEST E. KEET, PREVENTING PIRACY: A BUSINESS GUIDE TO SOFTWARE PROTECTION 44 (1985) (emphasis added).

*Gottschalk v. Benson* and *Diamond v. Dhier* is as long as it is idiosyncratic. In many cases the patent specifications included complete program listings; it can hardly be disputed that they incorporated algorithms.[145]

These inventions—and many others like them—had a century or more of patent history. The incorporation of embedded microprocessors was so widely accepted as an incremental product improvement that these innovations seemingly slipped unchallenged under the Patent Office door. There was an acceptance of creeping intangibility within patents, but it was difficult to draw the line where a patent would be rejected on the grounds of non-statuary subject matter.

An example of this virtualization is the postage meter. Arthur Pitney patented his first postage meter in 1901 and, with a partner Walter Bowes, had developed a significant business by the 1920s.[146] The original machine printed a fixed denomination on an envelope and incremented a tamper-proof counter.[147] Periodically, a Post Office account representative would read the meter and reset it after collecting the amount due. Subsequent models in the 1920s and 1930s allowed variable postage amounts to be printed and incorporated detachable meters so that the user could take the meter to the Post Office.[148] Next, electrically operated models were developed, and in the 1960s discrete electronics were incorporated. In 1976 a patent was granted for a "Microcomputerized Electronic Postage Meter System" that included a 20–page program listing.[149] In 1978, a "Remote Postage Meter Charging System" enabled postage costs to be settled electronically instead of physically taking the meter to the Post Office; again, the specification included many pages of computer code.[150] In 1981, another patent for a "Microcomputerized

145. The following patents are typical of those in the 1970s that included detailed program listings, flowcharts or algorithms in their specifications. U.S. Patent No. 4,255,789 (issued Mar. 10, 1981) (assigned to Bendix Corp.); U.S. Patent No. 4,208,717 (issued Jun. 17, 1980) (assigned to Westinghouse Electric Corp.); U.S. Patent No. 4,198,051 (issued Apr. 15, 1980) (assigned to Bally Manufacturing Corp.); U.S. Patent No. 4,154,855 (issued May 15, 1979) (assigned to Litton Systems, Inc.); U.S. Patent No. 4,138,719 (issued Feb. 6, 1979) (assigned to Xerox Corp.); U.S. Patent No. 4,089,524 (issued May 16, 1978) (assigned to Gremlin Industries, Inc.); U.S. Patent No. 4,063,620 (issued Dec. 20, 1977) (assigned to Westinghouse Electric Corp.); U.S. Patent No. 4,012,725 (issued Mar. 15, 1977) (assigned to Hewlett-Packard Company); U.S. Patent No. 3,962,569 (issued Jun. 8, 1976) (assigned to Reliance Electric Company);.

146. *See generally* WILLIAM CAHN, THE STORY OF PITNEY-BOWES (1961).

147. U.S. Patent No. 710,997 (issued Oct. 14, 1902).

148. The Pitney-Bowes business took off in 1920 with the detachable-meter model M. The invention is celebrated in PITNEY BOWES MODEL M POSTAGE METER 1920: AN INTERNATIONAL HISTORIC MECHANICAL ENGINEERING LANDMARK (American Society of Mechanical Engineers), 1986.

149. U.S. Patent No. 3,978,457 (issued Aug. 31, 1976).

150. U.S. Patent No. 4,097,923 (issued Jun. 27, 1978).

Electronic Postage Meter System" allowed the postage meter to be integrated with a personal computer and printer so that, for example, prepaid postage labels could be printed.[151] The innovations to the postage meter contained increasing levels of algorithmic components; none were rejected as non-statutory subject matter. This increasing virtualization demonstrated that software components could be patented within tangible inventions, while pure software was still largely unprotected.

One major software product did benefit from patent protection in the 1970s, a program called SyncSort.[152] The program was developed by Duane Whitlow, the proprietor of a small software contracting firm Whitlow Computer Services of Englewood Cliffs, N.J. (The company was later renamed SyncSort Inc., and it is still a significant industry player.) SyncSort improved the speed of the existing IBM sorting programs by at least a factor of two.[153] The program used a complex and very particular algorithm that had little application outside of the context of the IBM System/360 computer for which it was designed. The invention also made use of a novel and previously unknown machine instruction.[154] It may have been the particularity of the algorithm, or the fact that it made use of a newly discovered hardware innovation, that rewarded this pure software product with rare patent protection.

There was no shortage of potential test cases that a patent commissioner might have chosen to explore the patentability of software inventions in the courts. Hence, it is surprising that the case Commissioner Sidney Diamond chose for a writ of certiorari was one that seems to have been marginal and likely to shed little light on the subject.

In 1975, the Federal Mogul Corporation, a manufacturer of automotive parts, filed a patent for an improved rubber curing process developed by its engineers James Diehr and Theodore Lutton.[155] The manufacturing process involved the fabrication of rubber components that had to be removed from a heated mold when the rubber had cured. Prior to the Diehr-Lutton invention, the curing time was determined by an operator using rule of thumb and years of experience. The invention replaced the human agent with a set of temperature sensing devices and an algorithm, executed by a programmed computer, which automatically opened the mold after the computed cure time. The patent examiner rejected the

151. U.S. Patent No. 4,271,481 (issued Jun. 2, 1981).

152. U.S. Patent No. 4,210,961 (issued Jul. 1, 1980).

153. *See id.* at col. 23.

154. Surprisingly, the instruction was not documented and was unknown to IBM—it simply "fell out of the wiring." Quoting Duane Whitlow, cited in CAMPBELL-KELLY, *supra* note *, at 102.

155. Concise technical and legal histories of *Diamond v. Diehr* appear in BROCKMAN, *supra* note 130, at 216–17; STOBBS, *supra* note 130, at 300–06; and HANNEMAN, *supra* note 130, at 85–92.

patent on the grounds that the algorithm was non-statutory subject matter. The decision was reversed in the Court of Customs and Patent Appeals and affirmed by the Supreme Court on the grounds that the invention was so particular it did not rule out the use of the algorithm in other contexts. A patent was granted in 1982.[156]

After *Diamond v. Diehr*, the Patent Office was broadly favorable to pure software patents; however, the decision left many issues unresolved. Some commentators believed the decision to be a "green light for the unlimited patenting of software techniques"; the light was dim and the software industry did not engage gear.[157] It would be several years before inventors began regularly seeking pure software patents.[158] Meanwhile, copyright was king. Of the dozen or so guides to software protection published in the 1980s, all but one were devoted to copyright.[159] Even the software-protection guide published by the industry trade association ADAPSO focused on the use of copyright for protecting intellectual property in software.[160]

## V. THE LIMITS OF COPYRIGHT PROTECTION AND THE RISE OF TRADE SECRECY

*When programs cannot be protected by statute, they must be protected by secrecy. And when programs are not freely disclosed, they are not as likely to be improved by use. Secrecy does not*

156. U.S. Patent No. 4,344,142 (issued Aug. 10, 1982).

157. Stallman & Garfinkle, *supra* note 2, at 22.

158. Burke offers a "realpolitik explanation" that software patents were discouraged partly on the logistical grounds that the USPTO lack trained examiners. Burke, *supra* note 20, at 1146.

159. The exception was HANNEMAN, *supra* note 130. Typical books of the 1980s aimed at software makers and focusing on copyright protection included FREDERICK L. COOPER III, LAW AND THE SOFTWARE MARKETEER (1988); ANTHONY L. CLAPES, SOFTWARE COPYRIGHT, AND COMPETITION: THE "LOOK AND FEEL" OF THE LAW (1989); KEET, *supra* note 144; G. GERVAISE DAVIS III, SOFTWARE PROTECTION: PRACTICAL AND LEGAL STEPS TO PROTECT AND MARKET COMPUTER PROGRAMS (1985); THORNE D. HARRIS III, COMPUTER SOFTWARE PROTECTION: A PRACTICAL HANDBOOK ON COPYRIGHTS, TRADEMARKS, PUBLISHING AND TRADE SECRETS (1985); FREDERIC WILLIAM NEITZKE, A SOFTWARE LAW PRIMER (1984); THOMAS J. SMEDINGHOFF, THE LEGAL GUIDE TO DEVELOPING, PROTECTING AND MARKETING SOFTWARE: DEALING WITH PROBLEMS RAISED BY CUSTOMERS, COMPETITORS AND EMPLOYEES (1986).

160. KEET, *supra* note 144. For a history of ADAPSO, see Jerome L. Dreyer, *The ADAPSO Story*, DATAMATION, Mar. 1970, at 55. The ADAPSO software protection committee was formed in the mid-1960s. There are no known records of the committee, although former officers of the committee gave a retrospective account at the Intellectual Property Issues Workshop at a recent ADAPSO Reunion. SOFTWARE HISTORY CENTER, ADAPSO REUNION TRANSCRIPT, MAY 2–4, 2002 (2003).

*follow the Constitutional plan for intellectual property. It does not "promote the Progress of Science and Useful Arts."*

—Robert B. Bigelow[161]

In the late 1970s the personal computer revolution changed the market dynamics for software products. Whereas in the 1970s a successful mainframe program might have cost upward of $10,000 and sold only a few hundred copies, a successful personal computer software product typically cost $500 and sold several hundred thousand copies. For example, by late 1983 it was estimated that VisiCalc, the pioneering spreadsheet, had sold 700,000 copies at $250, and WordStar, the leading word processing package, had sold 650,000 copies at $495.[162]

A central IP concern for mass market software was piracy. Because mainframe packages had been directly purchased and controlled by corporate information systems departments, piracy was virtually unknown.[163] However, personal computer packages were often directly acquired by user departments or by individuals. These classes of users were generally less concerned about piracy than a corporate IS department. Piracy was rife in personal computer software, and as late as 1994 there was estimated to be one pirated copy for every legitimate copy of a software product.[164] Copyright law was fairly well positioned to protect against this sort of unauthorized copying of an actual software program. However, copyright law was expected to do double duty by also protecting against theft of the intellectual content of a software product by a competitor. To facilitate this protection regime, in 1970 the Copyright Office extended copyright protection to a machine readable version of a program and no longer required an inventor to deposit the source code. Much of the contemporary debate about software protection was muddied by use of copyright to attempt to protect both the code and functions of a program against appropriation by competitors.

Copyright protection was designed to protect only the *expression* of work, not its function. In the 1980s, this limitation of copyright protection appeared to legitimize the efforts of competitors of best-selling products to reverse engineer the program and create a competing clone product. Reverse engineering did not require the direct copying of code, but rather created a functional replica by the observation of the pro-

161. Bigelow, *supra* note 121, at 37.

162. *See* EFREM SIGEL, COMMUNICATIONS TRENDS, THE BUSINESS/PROFESSIONAL MICROCOMPUTER SOFTWARE MARKET, 1984–86, at 38 tbl.3.3 (1984).

163. SOFTWARE HISTORY CENTER, *supra* note 160 at 123.

164. BUS. SOFTWARE ALLIANCE & SOFTWARE AND INFO. INDUS. ASS'N, 1998 GLOBAL SOFTWARE PIRACY REPORT, *at* http://www.bsa.org/usa/press/newsreleases/loader.cfm?url=/commonspot/security/getfile.cfm&PageID=21740 (May 1999).

gram's black-box behavior.[165] Copyright law had already been shown to extend beyond literal copying. Cloning of computer programs was likened to the misappropriation of a plot in a play, something the courts found manifestly unfair.[166]

The most important reverse engineering cases were those between the Lotus Development Corp. and three firms that cloned its 1-2-3 spreadsheet.[167] *Lotus v. Paperback Software* was the first to come to court.[168]

Lotus introduced its 1-2-3 spreadsheet in January 1983 for the IBM PC.[169] During the next two years, the IBM-compatible PC became an industry standard and sales of 1-2-3 soared.[170] Paperback Software introduced its VP-planner spreadsheet in 1987.[171] It was self-described as being "keystroke for keystroke" compatible with 1-2-3, and relied on its low price of $99 to generate sales.[172] There was no claim of theft of 1-2-3's source code; VP-planner had been written from the ground-up to emulate the behavior of Lotus 1-2-3. Lotus was extremely vulnerable to competition of this kind. Lotus had gained its market position by early entry, huge investments in publicity,[173] and product refinement based on

165. Indeed, cautious cloners sometimes adopted "clean room" techniques to ensure that it could be proved no actual copying of code had taken place.

166. Dennis S. Karjala, *The Relative Roles of Patent and Copyright Protection of Computer Programs*, 17 J. MARSHALL J. COMPUTER & INFO. L. 41, 53 (1998).

167. Competitors of Lotus 1-2-3 included: Paperback Software's VP-planner, Mosaic Software's The Twin, and Borland International's Quattro. Discussions of the Lotus look-and-feel lawsuits are given in BERNARD A. GALLER, SOFTWARE AND INTELLECTUAL PROPERTY PROTECTION 91–104 (1995); LAWRENCE D. GRAHAM, LEGAL BATTLES THAT SHAPED THE COMPUTER INDUSTRY 56–58, 62–68 (1999); Pamela Samuelson, *Legally Speaking: How to Interpret the Lotus Decision (And How Not To)*, COMM. ACM. Nov. 1990, at 27. Lengthier discussions include Dennis S. Karjala & Peter S. Menell, *Brief Amicus Curiae: Applying Fundamental Copyright Principles to Lotus Development Corp. v. Borland International, Inc.*, 10 HIGH TECH. L.J. 177(1995); Pamela Samuelson, *Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of Lotus v. Paperback,* 6 HIGH TECH. L.J. 209 (1991).

168. The court decisions of Lotus against it's competitors are: Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995); Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203 (D. Mass. 1992); Lotus Dev. Corp. v. Borland Int'l, Inc., 788 F. Supp. 78 (D. Mass. 1992); Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990).

169. *See* Campbell-Kelly, *supra* note 22, at 332.

170. *See Lotus's One Millionth Software Package*, LOTUS, Jun. 1985, at 7 (reporting that 1-2-3 was priced at $495 and had sold a million copies by mid-1985).

171. For a comprehensive history of spreadsheets, see Campbell-Kelly, *supra* note 22, at 322–347.

172. The phrase "keystroke for keystroke" used in the judgment appeared in Paperback Software's own user manual PAPERBACK SOFTWARE INTERNATIONAL, VP-PLANNER, at xi (1987).

173. The development and launch costs of Lotus 1-2-3 are detailed in Peter Petre, *The Man Who Keeps the Bloom on Lotus*, FORTUNE, Jun. 10, 1985, at 136–46.

user feedback. A cloning competitor could free ride on these investments.

In January 1987 Lotus filed suit against Paperback Software. Because Paperback Software had not infringed Lotus copyright by direct copying, Lotus sought to apply the obscure concept of look-and-feel infringement.[174]

In 1990, the District Court of Boston issued a controversial decision, finding that Paperback Software had infringed Lotus' copyright by copying the menu structure of 1–2–3.[175] The decision implied that user interfaces would have to be significantly different to avoid infringement, whereas users of competing products benefited by having a common user interface across applications. As a result of this decision, VP-planner was withdrawn from the market and Mosaic Software also pulled its product before its case came to court.

The suit against Borland had a different outcome. Borland had designed its product with two interfaces that the user could choose between. One interface was Borland's own, while the other emulated that of 1-2-3. Like Paperback Software, Borland lost in the lower court. But Borland won on appeal and the decision was upheld by the Supreme court against challenge by Lotus.[176] The software industry interpreted this decision to mean that it was acceptable to emulate an interface or menus to assist in product switching, but that the cloned interface should not be the primary interface of a program.

Copyright protection of software was an uncertain business. Much of the reason for the debate over software protection is that programs contain two distinct kinds of intellectual investments. The first is the source code. Source code is entirely utilitarian: it is prosaic, workmanlike, unimaginative grunt work[177] that has always been fantastically expensive to write and debug.[178] Copyright protects source code from appropriation in principle, but not in practice, and for this reason very few for-profit

174. This concept had had a number of previously successful outings. For example, in 1970 in *Roth Greetings Cards v. United Card Co.* 429 F.2d 1106 (9th Cir. 1970), the plaintiff succeeded in a copyright action against a competitor that had published "substantially similar" greetings cards to its own. The court applied the "total concept and feel" test to find infringement. Another widely publicized suit was *Sid & Marty Krofft Television Productions Inc. v. McDonald's Corporation*, in which characters in the McDonaldland TV commercials were found to have infringed characters in the children's TV program *H.R. Pufnstuf.* Sid & Marty Krofft Television Prod. Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977). On the origins of look-and-feel, see GRAHAM, *supra* note 167, at 54–55.

175. Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990).

176. Lotus Dev. Corp. v. Borland Int'l, Inc., 516 U.S. 233 (1996).

177. Programmers often talk about the elegance of their programming efforts. However, such elegance has little economic value. It has the same kind of value that an elegantly worded memo or email message has over a purely utilitarian one.

178. *See supra* note 102 and accompanying text.

mass-market software makers have disclosed their source code.[179] The second intellectual investment is in the functional attributes of the program, such as algorithms, data structures, user interfaces and menu structures. Frequently, even if a publisher only disclosed the binary version of the program, a clause was included in the licensing agreement to expressly forbid code examination for the purpose of reverse engineering.[180] This was an attempt to incorporate trade secrecy as a supplement to the insufficient copyright regime.

Although source code was almost never disclosed for mass-market software, disclosure persisted with enterprise software into the 1980s. Indeed, an entire industry had developed producing complementary products for IBM's operating systems, database systems and teleprocessing products. The industry only existed by virtue of access to IBM's source code. However, in March 1983, IBM announced its Object Code Only (OCO) policy, by which it ceased to supply the source code of its software products.[181] IBM stated at the time of the OCO announcement that it would provide "application program interfaces"—now known as APIs.[182] However, at that time the phrase was a neologism, and IBM's competitors argued that IBM was incapable of defining satisfactory interfaces to allow seamless integration with IBM's programs. The IBM user group SHARE and the software industry trade association ADAPSO lobbied without success to have the OCO policy reversed; gradually the protests faded away.[183]

In 1987 IBM announced its System Application Architecture (SAA), a program by which it formally published APIs for all its software products.[184] This made the publication of APIs much more formal and systematic. Indeed, trade secrecy and the evolution of published APIs

179. Although the Netscape Corporation has recently disclosed the source code of its Communicator web browser, this appears to be a tactical ploy in its longstanding battle with Microsoft over web browser supremacy.

180. For example, Microsoft's standard license agreement of the late 1980s contained the clause, "You may not modify, adapt, translate, reverse engineer, decompile, disassemble, or create derivative works based on the [software]."

181. Ralph Emmett Carlyle & Jeff Moad, *IBM and the Control of Information*, DATAMATION, Jan. 1, 1988, at 34, 38, 41, 44. Martin Goetz & Peter Schneider, *Object Code Only: Is IBM Playing Fair?* COMPUTER WORLD, Feb. 8, 1988, at 55, 58–59, 62, 66.

182. *See* Carlyle & Moad, *supra* note 181, at 44.

183. The ADAPSO petition was withdrawn after IBM agreed to give independent software companies assistance with problems related to source code access. Willie Schatz, *ADAPSO Withdraws Opposition to IBM's Object Code Stance*, DATAMATION, Oct. 1, 1988, at 17.

184. *See generally* MICHAEL KILLEN, IBM: THE MAKING OF THE COMMON VIEW (1988); L. ROBERT LIBUTTI, SYSTEMS APPLICATION ARCHITECTURE: THE IBM SAA STRATEGY (1990).

can be viewed as a technological solution to the failure of copyright to adequately protect source code from appropriation by competitors.[185]

Under the regime of trade secrecy, algorithms and data structures remained secret and consequently quite complex techniques (spelling checkers in word processors, say) have been endlessly reinvented by other developers for other programs needing the same functionality. Provided they meet the novelty requirements of patent law, algorithms, data structures, user interfaces, and menu structures are all patentable subject matter. Such patents have the major benefit of disclosing techniques otherwise locked up in binary programs.

## VI. Patents and Disclosure

*Software from UK company Pixology is claimed to be the first technology for correcting the red eyes that come up in photographs taken with flash cameras. Its Iriss software removes red-eye, which is caused by the reflection of the flash light from the thin capillaries at the back of the eye, intensified by the lens in front of the eye. Pixology's technical director, Mike Stroud, says "It's a secret how we do it. All I will say is that Iriss does not look for the eyes in a photo."*

—Computer Bulletin, November 2003, p. 21

The U.S. Patent Act of 1790 was introduced to prevent exactly the kind of trade secrecy suggested by the above quotation. The firm Pixology is based in Europe, where pure software innovations are not currently patentable. Although copyright protection for software exists in Europe, this would not protect Pixology against a competitor imitating its technology. Hence Pixology's reliance on trade secrecy.

A controversial and well known software patent for the LZW data compression algorithm clarifies a difference between copyright and patent protection.[186] The LZW algorithm is named for its inventors Abraham Lempel, Jacob Ziv, and Terry Welch. A data compression algorithm constitutes a technique for reducing in size a quantity of binary data. Most computer data is highly redundant, and compression of 90 percent is not uncommon. The particular advantage of the LZW algorithm is its sim-

185. Of course, APIs have many other functions besides concealing source code. They are analogous to the standard interfaces used throughout the engineering industry.

186. For competing views of two computer scientists on the legitimacy of the LZW patent, see Stallman & Garfinkle, *supra* note 2; and Paul Heckel, *Debunking the Software Patent Myths*, COMM. ACM, June 1992, at 121, 133.

plicity and speed. Using only modest computer power, it enables images to be compressed and decompressed almost instantaneously.

Terry Welch applied for a patent for the algorithm in June 1983, which was subsequently assigned to his employer Unisys. A year later a description of the compression method was published in *IEEE Computer*, a widely read technical magazine.[187] The disclosures through the patent and article provided an exemplary description of the invention, both in terms of an "apparatus" (a set of electronic registers and control signals), and as an algorithm in the form of a set of four FORTRAN subroutines consisting of approximately 110 lines of code. Anyone wanting to use the LZW algorithm could purchase a license from Unisys and copy the FORTRAN routines (or transliterated routines) into their own program. The patent expired in June 2002; since then, anyone has been free to use this method without a license. It is difficult to think of a more conventional or appropriate use of the patent system.

If instead of patenting LZW, only copyright applied, copyright would protect the FORTRAN exposition of the algorithm, but it would not protect the algorithm itself. Anyone would be free to use the algorithm, perhaps writing it in another programming language such as C or Java, or even in FORTRAN provided the implementation was sufficiently distinct. Copyright would effectively confer no protection at all. As discussed above, without software patents, this lack of protection in the copyright regime leads to greater use of trade secrecy and less public disclosure.

The LZW patent did not have the effect of stifling competition; it encouraged it. Inventors had access to a very precise description of the LZW code and were free to improve it or to circumvent it in a non-infringing manner.[188] However, LZW remains a very good compression tool, although ever-improving computer speeds have reduced its speed advantage.

Much of the controversy over the LZW patent concerned its exploitation by Unisys. Welch's publication in *IEEE Computer* in 1984 did not state that a patent had been applied for. This article was widely read by the software community, and the technique was presumed to be in the public domain. As a result, the algorithm quickly diffused into many

187. U.S. Patent No. 4,558,302 (issued Dec. 10, 1985). Terry A. Welch, *A Technique for High Performance Data Compression,* COMPUTER, Jun. 1984, at 8. COMPUTER is the flagship magazine published by IEEE for the diffusion of knowledge among its computer and electronic-engineering professional membership. Unlike patents and the academic literature, COMPUTER is widely read by practitioners.

188. In fact, at the time of writing there are 185 patents in the USPTO database citing LZW as prior art, and several hundred patents have been granted for data compression techniques since LZW was issued. Author's estimates based on USPTO website searches, Apr. 4, 2005.

computer technologies requiring data compression, including imaging systems, modems, and laser printers. Most significantly, it was used in the Graphics Interchange Format (GIF), a popular image compression technique developed for the CompuServe consumer network in 1987. At that time, the bandwidth of consumer networks was so limited that the use of images would have been infeasible without compression. Unaware of the existence of the LZW patent, CompuServe made GIF's available "for use in computer software without royalties, or licensing restrictions."[189] With the rise of the World Wide Web in the early 1990s, GIFs probably accounted for more network traffic than any other file type. The particular advantage of the LZW technique was that it enabled images to be decompressed and rendered on a computer screen almost instantaneously using modest PC hardware.

In 1990, Unisys asserted its intellectual property rights and demanded royalties from users of the patent. The software community was outraged, provoking many to abandon GIFs in favor of public domain JPEG images. Infringers who were locked into using GIFs or LZW compression reluctantly bought licenses from Unisys. In a 2003 press release, Unisys stated that almost 3000 license agreements had been issued worldwide by the time the U.S. patent expired.[190]

What affronted much of the software-writing community was what amounted to deception. Software developers had been led to believe the technique was in the public domain, only to have a patent enforced when users reached the point of no return. Although the LZW technique was deserving of a patent, data compression techniques are not difficult to substitute. Had the need to pay royalties for LZW been known, adequate royalty-free substitutes would likely have been developed by the open source community and incorporated into derivative products in lieu of the LZW algorithm.[191] However, it is a poor argument to cite this malpractice to condemn software patents in general.

As the examples of the punched-card machines and typewriters discussed earlier in this article illustrate, patents have played a powerful role in the cumulative improvement of inventions. Public-key encryption provides a compelling example of cumulative improvement in the do-

189. Graphics Interchange Format (GIF) Specification, *at* http://www.w3.org/Graphics/GIF/spec-gif87.txt (Jun. 15, 1987).

190. *See* "LZW Patent and Software Information: License Information on GIF and Other LZW-based Technologies," Unisys Corporation website, *at* http://www.unisys.com/about__unisys/lzw (visited Jul. 11, 2003). The press release is no longer available on Unisys' website, but a copy has been archived on an open-source website *at* http://www.sslug.dk/patent/lzwunisys.html

191. *Supra* note 16.

main of software.[192] This encryption technique has been described by a historian of cryptography, Simon Singh, as the most important development in cryptography in 2000 years.[193]

Prior to public-key encryption, the most widely used technique was the data encryption standard (DES), patented by IBM in 1976.[194] At that time, secure computer communications were primarily used in the civilian sector for the transmission of high value funds between financial institutions. In DES, users employed a common encryption-decryption algorithm, relying on a unique key shared between parties to ensure communications could not be deciphered. To guarantee security, the keys had to be distributed by a courier at considerable expense. As a result, DES encryption was practical only for high value transactions.[195]

The invention of public-key cryptography is one of the most fascinating scientific success stories of the twentieth century.[196] Whereas in private-key encryption, the keys have to be kept secure, in public-key ciphers this is not necessary. As a result, the cost of key distribution is eliminated.[197] The feasibility of public-key encryption was discovered by two Americans, Martin Hellman and Whitfield Diffie, working from Stanford University. Diffie and Hellman filed a patent for public-key cryptography, the technology that today underpins secure Internet commerce, in September 1977 and assigned the patent to Stanford University.[198]

One more piece of the puzzle remained. Diffie and Hellman had proven the feasibility of public-key encryption; they had not yet identified a fast, practical algorithm. This was achieved a few months later with the RSA algorithm, named for its inventors Ronald Rivest, Adi Shamir, and Leonard Adleman all of whom were MIT faculty. Rivest, Shamir, and Adleman filed for a patent in December 1977, assigning rights to their employer, MIT.[199] In 1983, MIT licensed the technology to

192. Good histories of public-key cryptography include SIMON SINGH, THE CODE BOOK: THE SCIENCE OF SECRECY FROM ANCIENT EGYPT TO QUANTUM CRYPTOGRAPHY (1999); LEVY, *supra* note 29.

193. SINGH, *supra* note 192, at 123.

194. U.S. Patent No. 3,798,359 (issued Mar. 19, 1974).

195. For an authoritative technical and historical account of DES, see DAVIES & PRICE, *supra* note 141, at 47–78. Besides being an inventor of packet switching used on the Internet, the late Donald Davies was an accomplished amateur historian of cryptography. *See also* William E. Burr, *Data Encryption Standard*, *in* A CENTURY OF EXCELLENCE IN MEASUREMENTS, STANDARDS, AND TECHNOLOGY, at 250–253 (David R. Lide ed., 2001).

196. The following historical account is based on SINGH, *supra* note 192, at 120–130.

197. The public-key encryption technique, which sounds impossible at first encounter, is based on the use of "one-way" mathematical functions. An authoritative technical description is given in DAVIES & PRICE, *supra* note 141, at 209–251.

198. U.S. Patent No. 4,200,770 (issued Apr. 29, 1980).

199. U.S. Patent No. 4,405,829 (issued Sept. 20, 1983).

RSA Data Security Inc., for which the inventors served as consultants.[200] With the rise of the Internet, the RSA algorithm provided a low-cost mechanism for secure, low-value financial transactions. The technology was licensed *inter alia* for Microsoft's Internet Explorer and Netscape's Communicator web browsers, where it now underpins virtually all small consumer transactions. The company was sold for $200 million in 1996. Today, RSA Security has nearly 14,000 customers, technology partners for over 1000 products and annual sales of $260 million.[201] Although the original RSA patent expired in 2000, RSA Security remains well placed, with more than thirty subsequent patents.[202] Indeed, the RSA patent was the foundation of a wave of innovation: no less than 350 cryptography patents cite RSA as prior art. Moreover, because there is no trade secrecy, university research and teaching of encryption technology is thriving.[203]

The downside of patenting is that it can provide rough justice for some. For example, Michael Williamson and Clifford Cocks of the British government's code-breaking operation GCHQ (Government Communications Headquarters) independently invented public-key encryption before Diffie and Hellman, but never disclosed or patented the invention.[204] Because of the first-to-file rule, Britain was obliged to pay to use the RSA algorithm, just like anyone else. Patents can also exclude newcomers. In the late 1980s, while RSA Data Security Inc. catered to the military and business markets, a computer programmer and political activist Phil Zimmerman wanted to bring public-key encryption to the ordinary PC user.[205] He called his system PGP, for Pretty Good Privacy.[206] Inevitably PGP infringed the RSA patent, and Zimmerman was further embroiled in a government prosecution for exporting military technology without a license. Ultimately, PGP leaked out and the RSA patent was close enough to expiration not to be worth pursuing. The world got its Pretty Good Privacy.

200. LEVY, *supra* note 29.
201. RSA SECURITY INC., ANNUAL REPORT, at 5, 6, 12 (2003).
202. *Id.* at 8.
203. There are several peer-reviewed journals in computer security and cryptography and dozens of textbooks and practical guides for university courses.
204. SINGH, *supra* note 192, at 123.
205. For a good history of PGP, see generally SIMSON L. GARFINKEL, PGP: PRETTY GOOD PRIVACY (1994).
206. "Pretty Good Privacy" was an homage to Ralph's Pretty Good Groceries in Garrison Keillor's *Prairie Home Companion*. LEVY, *supra* note 29, at 195.

## VII. BUSINESS METHOD PATENTS

*Supposedly, in order to be patented, something has to be new, useful and unobvious. . . . Of course, when the patent office gets into the game, they start interpreting new and unobvious. New turns out to mean we don't have it in our files and unobvious tends to mean unobvious to someone with an IQ of 50.*

—Richard Stallman[207]

The above quotation from Richard Stallman is typical of the antagonism toward software patents, and especially business method patents, that appears on the World Wide Web. There is unquestionably a problem concerning the threshold of obviousness, but the rhetorical assertion that "unobvious tends to mean unobvious to someone with an IQ of 50" does not get anywhere near the heart of the problem.[208]

The amazon.com "1-click" patent attracts more attention than any other for its apparent obviousness.[209] The standard method of purchasing from a website is to place the goods in a virtual shopping cart, then to provide shipping information at checkout and to authorize payment by credit card. Amazon.com replaces this often-tedious process with "1-click" ordering. Once 1-click has been activated, the user simply clicks on items and all the minutia of payment and delivery are completed on the basis of information already stored in Amazon's database.

The patent may look trivial and obvious, but this is a judgment made in hindsight, and the popular debate is often uninformed by the nature of invention or the mechanisms of disclosure. Examination of the 1-click

207. Richard Stallman, Software Patents—Obstacles to Software Development, Lecture at the Computer Laboratory, University of Cambridge (Mar. 25, 2002), *at* http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html.

208. Good discussions on the obviousness of business method patents include ADAM B. JAFFE & JOSH LERNER, INNOVATION AND ITS DISCONTENTS (2004); Ron Laurie & Robert Beyers, *The Patentability of Internet Business Methods: A Systematic Approach to Evaluating Obviousness*, *in* COPY FIGHTS: THE FUTURE OF INTELLECTUAL PROPERTY IN THE INFORMATION AGE 237–71 (Adam Thierer et al. eds., 2002); Robert Merges, *As Many as Six Impossible Patents Before Breakfast: Property Rights for Business Concepts and Patent System Reform* 14 BERKELEY TECH. L.J. 577 (1999).

209. The patent is examined in many scholarly discussions of business method patents, both pro and con. BRONWYN H. HALL, BUSINESS METHOD PATENTS, INNOVATION, AND POLICY (NBER Working Paper No. 9717, 2003); STARLING DAVID HUNTER, HAVE BUSINESS METHOD PATENTS GOTTEN A BUM RAP? SOME EMPIRICAL EVIDENCE (MIT Sloan School of Management, Working Paper No. 182, 2003); Keith Maskus & Eina Wong, *Searching for Economic Balance in Business Method Patents*, 8 WASH U.L.J. & POL'Y 289, 292, 299 (2002); Dreyfuss, *supra* note 36; Peter Wayner, *How Can They Patent That?, in* COPY FIGHTS: THE FUTURE OF INTELLECTUAL PROPERTY IN THE INFORMATION AGE 221–28 (Adam Thierer et al. eds., 2002).

patent shows that the technology involved is not trivial by the standard of inventions in general. The specification consists of 6 pages of text and 11 pages of drawings.[210] The invention involves the deployment of web and database technologies, and has the added complication that multiple 1-click orders have to be integrated into a single shipment, with minimal shipping cost, and a single charge deducted from the customer's credit card. The 1-click invention is not out of line with the threshold of patentability in other fields. The patent examiner granted the patent, not because she was incompetent, but because the invention met the conventional thresholds of novelty, obviousness and inventiveness.

One reason that critics consider the 1-click patent to be obvious, is that the level of skill needed to implement it is not particularly high. (This criticism is often made by programmers who have an exceptional level of skill, such as Richard Stallman. It is rather like a concert pianist assuming every saloon-piano player shares his or her level of skill.) In fact, implementing the 1-click process on a website would demand the skills typically found in a computer science graduate or a substantially experienced journeyman programmer.

However, patents cannot be judged on the basis that they do not look unduly complicated or difficult to implement. Many, perhaps most, patents would fail by that test; Hollerith's stop card patent was certainly not difficult, but few would argue it was unpatentable. Patents that seem obvious are often only obvious in hindsight, because a patent discloses the invention but not the process of discovery. It is rather like a mathematical theorem or a poem—if it is done well, the pain of creation is not perceptible. In the case of the 1-click patent, we do know something of the discovery process.[211] Amazon.com's founder Jim Bezos has long held a mission "to make Amazon.com 'the most customer-centric' company in history."[212] The company is considered to be the first Internet retailer to use a "wizard" approach to order processing—taking the customer through a series of steps to complete a purchase, with the option of advancing or going back a step at each stage. In this context, 1-click can be seen as the next stage in an evolutionary inventive process. Bezos claims that developers spent "thousands of hours" perfecting the process. For example:

> When we did focus groups and tested this new feature before launching it, the biggest problem was that people didn't think they had really finished the order. . . . So we had to change the text surrounding this thing to not only say "Thank you for your

210. U.S. Patent No. 5,960,411 (issued Sept. 28, 1999).

211. *See generally* ROBERT SPECTOR, AMAZON.COM: GET BIG FAST (2000).

212. *Id.* at 126.

> order", but in parenthesis it now says "Yes, you really did place an order."[213]

The devil is in the details, and the 26 claims of the patent are full of such details.

No one had used 1-click ordering on the web before. After Amazon.com began using the system in late 1997, it was imitated by several on-line merchants including Amazon.com's principal competitor Barnes and Noble. Amazon.com filed an infringement suit against Barnes and Noble in October 1999. This led to an outcry from the Internet community, and the Free Software Foundation launched a "Boycott Amazon!" campaign.[214] The suit was eventually settled out of court, although the terms have not been disclosed.[215]

It is not the purpose of this article to advocate the merits of the 1-click patent. That has been amply and inconclusively discussed elsewhere. Rather, this article does attempt to encourage recognition that business method patents can be genuinely innovative, involve costly invention, and meet the normal criteria of patentability. Moreover, in the same way that some early patents can now be seen to embody algorithms (as discussed in section 4 of this article), there are early patents that can retrospectively be classed as business method patents.[216]

There is a common perception that patents have been granted for taking any real world invention and putting it on the Internet. This is a distortion; it is true only to the extent that an Internet-based invention passes the normal thresholds of usefulness, novelty and non-obviousness. To disallow inventions solely on the grounds that they are implemented through the Internet would be to fail to recognize the virtualization of inventions that has gone on since World War 2.

An instructive example of the virtualization of inventions that is currently taking place is the "virtual postage meter," by which firms and individuals can purchase postage stamps and prepaid mailing labels through the Internet.[217] The virtualization of postage meters was introduced in section 4. Virtual postage meters have now been in existence

213. *Id*. at 153.

214. Steven Levy, *The Great Amazon Patent Debate*, NEWSWEEK, Mar. 13, 2000, at 74.

215. Richard S. Grunner, *Everything Old is New Again: Obviousness Limitations on Patenting Computer Updates of Old Designs*, 9 B.U. J. SCI. & TECH. L. 209, 237 (2003).

216. For example, the following two patents for insurance quotations, dated 1919 and 1999, incorporate essentially the same business methods. U.S. Patent No. 1,314,146 (issued Aug. 26, 1919) (illustrating projections using a cardboard calculator); and U.S. Patent No. 5,956,691 (issued Sept. 21, 1999) (illustrating projections using a computer screen).

217. Damon Darlin, *Innovate or Die*, FORBES, Feb. 24, 1997, at 108; Stephen Manes, *Going Postal—Digitally*, FORBES, Sept. 6, 1999, at 228.

for ten years and have an appreciable business history.[218] They are a lens through which we can further examine the role of patents in Internet-based inventions.

There have been five significant players in virtual postage metering—three entering the postage industry strictly as Internet-providers, E-Stamp, Stamps.com, and Endicia; and two that were in the postage market prior to their Internet implementations, Pitney Bowes's ClickStamp, and Neopost's Simply Postage. Of the new entrants, the first mover was E-Stamp, whose founder Salim Kara applied for a patent in 1994, issued in 1996[219] E-Stamp was followed by Stamp.com (with no patent) and Endicia (with a patent issued to its CEO Harry Whitehouse in 1999).[220] The firms began trading in 1999-2000 when US Post Office approval had been granted. The incumbents Pitney Bowes and Neopost entered somewhat later, again with their own patents.[221]

In 1999 Pitney Bowes filed an infringement suit against E-Stamp and Stamp.com, subsequently settled with a cross-licensing agreement with "no material financial payment."[222] It is tempting to characterize the new entrants as David and the incumbents as Goliath. It is as if the newcomers got to the party first and then the incumbents muscled in only when the former had established the viability of the business. The history is more complex than this. The incumbents were caught in a classic situation characterized as a disruptive technology.[223] Unlike the new entrants with no history, the incumbents were major firms that had to radically re-engineer their business model in a way that sustained their current operations while migrating to the new world; the transition is still in progress. It is not unlike the painful transition that IBM had to make to adapt to the personal computer. In both cases, to portray the incumbents as technological dinosaurs only waiting in the wings ready to stomp on the newcomers is too simplistic.

There are some empirical observations one can make from the virtual postage meter story. First, there is no monopoly; the market has

218. COMPETITION COMMISSION (UK), NEOPOST SA AND ASCOM HOLDING AG: A REPORT ON THE PROPOSED MERGER 61-64(2002), *at* http://www.competition-commission.org.uk/rep_pub/reports/2002/465neopost.htm (last visited May 8, 2005).

219. U.S. Patent No. 5,510,992 (issued Apr. 23 1996). Several patents were granted to Kara subsequently.

220. U.S. Patent No. 6,005,945 (issued Dec. 21, 1999), and subsequent patents.

221. Pitney Bowes: U.S. Patent No. 6,064,993 (issued May 16, 2000), and subsequent patents. Neopost: U.S. Patent No. 6,424,954 (issued Jul., 23, 2002), and subsequent patents.

222. Press Release, Stamps.com, Pitney Bowes and Stamps.com Settle Patent Infringement Litigation (Dec., 22 2003), *at* http://www.stamps.com/company/news/20031222a (last visited May 8, 2005).

223. Joseph L. Bower & Clayton M. Christensen, *Disruptive Technologies: Catching the Wave*, HARV. BUS. REV., Jan.–Feb. 1995, at 43.

sustained both newcomers and incumbents. The existence of patents has not allowed a first mover to enjoy a monopoly, but it has discouraged non-innovating imitators. Because the newcomers had strong patent positions, they were not immediately out-gunned by the powerful incumbents. Second, the incumbents have been tested by truly powerful competitive forces, probably more dramatic than at any time in their history. One should note that Pitney Bowes is a very innovative firm, when measured by patent activity; it has a portfolio of 3000 patents, and makes 200–300 new applications a year.[224] If Pitney Bowes had been non-innovating, it would not have survived the transition to virtual postage meters. Third, a lot more is required to succeed than having a sound patent position. There has been shake-out and consolidation in the industry, particularly following the dot-com bust. Finally, although a virtual postage meter is a totally intangible invention, the virtual meter can infringe on existing postage meter patents, themselves the fruit of costly R&D investments. It is difficult to draw a line. If patents were wholly disallowed for Internet-based inventions, then any present-day artifact capable of virtualization would be vulnerable to emulation regardless of prior R&D investments. In this regard, virtual postage meters are in the vanguard of a change that may sweep across manufacturing to a degree we cannot yet predict.

## VIII. BROAD PROSPECTS AND REVERSE SALIENTS

*A salient is a protrusion in a geometric figure, a line of battle, or an expanding weather front. As technological systems expand, reverse salients develop. Reverse salients are components in the system that have fallen behind or are out of phase with the others.*

—Thomas P. Hughes[225]

For the last 20 years, the history of technology has been dominated by a theory of technological advance that is frequently referred to as the Social Construction of Technology (or "SCOT"). A leading protagonist of this theory, Thomas Hughes, has used the military metaphor of the "reverse salient" to illustrate the theory. It is as if technology advances along a broad front, but from time to time, part of the front fails to keep

224. *Pitney Bowes Ranks in Top 200 Companies Receiving U.S. Patents for 14th Consecutive Year*, BUS. WIRE, Apr. 13, 2000, *at* http://www.findarticles.com/p/articles/mi_m0EIN/is_2000_April_13/ai_61475928.

225. Thomas P. Hughes, *The Evolution of Large Technological Systems, in* THE SOCIAL CONSTRUCTION OF TECHNOLOGICAL SYSTEMS: NEW DIRECTIONS IN THE SOCIOLOGY AND HISTORY OF TECHNOLOGY 51, 73 (W.E. Bjiker et al. eds., 1987).

up with the general advance. At that point additional technological warriors move to the reverse salient and seek to push it forward. These technological warriors are not typically lone inventors operating in trade secrecy, but individuals and firms that cooperate through information disclosure in order to solve the critical problem.[226]

The history of information processing affords many compelling examples that validate the SCOT perspective. For example in the late 1940s a critical problem in computing was that random access memory was slow and unreliable, and this led to the development of magnetic core memory.[227] In 1949 a new magnetic ceramic Deltamax hit the market. In the words of a contemporary memory researcher, "It was completely obvious that you could make a memory with this material."[228] Numerous players entered the field, including IBM, RCA, several lesser firms such as Wang Laboratories, and university research laboratories including MIT and the University of Illinois. The first two related patents were filed by RCA and MIT's Jay Forrester in 1950–51.[229]

It was at about this time that IBM decided to enter the computer business. Its first computers used the existing memory technologies but they proved too unreliable for a high-volume, commercial product.[230] For its model 704 computer, to be delivered in 1954, IBM decided to use core memory. As there were no commercially available products, IBM launched its own development program.[231] IBM had a patent sharing agreement with RCA, started a negotiation with MIT, and set to work.[232] The core memory for the 704 was a technological improvement over the existing technologies, but was prohibitively expensive. The 4096 word[233] memory for the 704 rented for $6,100 a *month* (equivalent to a manufacturing cost of $1.31 per bit).[234] IBM, however, worked on improving the technology; it acquired several more patents and developed its own patented innovations. Protected by its patents, IBM was able to subcontract manufacture to specialists instead of investing in costly fabrication plants. However, as the market for computers boomed, IBM went into

226. *Id.*

227. *See generally* EMERSON W. PUGH, MEMORIES THAT SHAPED AN INDUSTRY: DECISIONS LEADING TO IBM SYSTEM/360 (1984).

228. *Id.* at 82.

229. U.S. Patent No. 2,736,880 (issued Feb. 28, 1956) (filed 1951, assigned to MIT); U.S. Patent No. 3,164,813 (issued Jan. 5, 1965) (filed 1950, assigned to RCA). Both patents describe small but scaleable memory systems based on magnetic cores. The patents are discussed in detail in PUGH, *supra* note 227, at 81–87.

230. PUGH, *supra* note 227, at 138.

231. *Id.*

232. BASHE ET AL., *supra* note 56, at 269.

233. Here, a "word" is equivalent to 32 bits, or 4 bytes.

234. PUGH, *supra* note 227, at 138.

full scale production in its own right. IBM improved not only in the design of the memory, but also the manufacturing processes, producing cheaper, smaller, faster cores. In the fifteen years from 1955 to 1970, the cost of core memories fell by a factor of 200.[235] In 1968, however, IBM abandoned its research into core memory. The new technology of microelectronics promised semiconductor memories (the kind we use today) and another cycle of innovation started.

The way in which the patent system facilitated the development of core memory was not universally beneficial. The primary benefit was that it enabled information sharing. Instead of several laboratories operating in trade secrecy, their inventions were disclosed and became available to all, subject to the intellectual property rights of the owners. The downside was that the owner of one crucial patent could hold the others to ransom. In this case, MIT's Research Corporation was the hold-out, initially seeking a 2 cents per bit royalty. IBM argued that if all the 13 patents involved were licensed on the same basis "the cost per bit would be twenty-six cents, making core storage economically infeasible."[236] IBM also believed that one day core memories would account for billions of bits of storage. IBM refused to take out a license with the MIT Research Corporation, which responded with an infringement suit. IBM ultimately conceded to pay $13,000,000, said to be the largest patent settlement ever made up to that date.[237] However, IBM believed that it was only the $26 million invested by IBM in core memory development that made the Forrester patent so valuable.[238] IBM was so affronted, it cut off its long-standing diplomatic relationship with MIT, and IBM's President Thomas Watson Jr. resigned from the board of trustees of MIT.[239] Between royalties and the cost of litigation, the core memory patents did not come cheap. Still, IBM has not lost faith in the patent system; indeed, today it takes out more patents than any other firm in the world.[240] Although software patents occasionally seem unjust, it has to be kept in mind that such unfairness is not unique to software; it is an unfortunate by-product of the patent system that affects all industries.

235. *Id.* at 245.

236. *Id.* at 209.

237. BASHE ET AL., *supra* note 56, at 270. It should be noted that IBM cut a good deal. By 1963, it was making a billion cores a year; by 1970, 20 billion a year. *See* PUGH, *supra* note 227, at 245.

238. PUGH, *supra* note 227, at 212.

239. James W. Birkenstock, "Pioneering: On the Frontier of Electronic Data Processing, a Personal Memoir," IEEE ANNALS HIST. COMPUTING Jan.–Mar. 2000, at 4, 29.

240. IBM was granted 3411 U.S. patents in 2001. The top-ten patenting firms were NEC, Canon, Micron Technology, Samsung, Matsushi, Sony, Hitachi, Mitsubishi, and Fujitsu. PATENTING BY ORGANIZATIONS 2001, *supra* note 41, at B1-1.

Computer memory is no longer a critical problem for the computer industry. Today, many of the reverse salients are on the software front. These are major technological barriers that will not likely be solved by solitary inventors, and probably not by the research laboratories of individual firms, but through the combined efforts of industry and university researchers. Two topical examples of such critical problems are screen rendering and speech recognition.

For several years the market for electronic books has failed to take off. One reason for this failure is that reading from today's computer screens is impractical and uncomfortable compared with reading from a printed page. It is impractical because today's LCD screens are clunky, power hungry, and have poor contrast.[241] Screen technology is improving, and many firms are developing next-generation technologies, such as e-paper, that will eventually fix some of the ergonomic and portability issues.[242] These firms are aggressively patenting their hardware innovations, with little or no media attention. But screen technologies are advancing not just by hardware improvements but also by software innovations such as Microsoft's ClearType and Adobe Systems' Cool Type.

Today, the resolution of a lap-top screen is about 100 dots or *pixels* per inch. This does not begin to compare with the minimum 1000 dots per inch of a printed book. Because resolution is perceived in dots per square inch, the perceived resolution difference between a computer screen and the printed page is a factor of a hundred.

In May 2001, Microsoft obtained its first patent for ClearType technology.[243] The details of ClearType are complex, though the principle is simple enough. The technology makes use of the fact that on a color screen each pixel consists of three sub-pixels, one for each of the three primary colors. Normally, when displaying black characters on a white background, every sub-pixel is either fully on or off with no gradations in between. ClearType technology exploits the fact that each sub-pixel can be individually set to an intermediate brightness. When ClearType characters are examined under a magnifying glass, one can see that individual sub-pixels have been used to smooth out the blocky, jagged edges of characters and enhance their serifs. A usability guru, Jakob Nielsen, estimates it improves reading speed by 10 to 15 percent.[244]

241. For a user perspective on e-books, see Stephen H. Wildstrom, *A New Chapter for E-Books*, BUS. WK., Mar. 27, 2000, at 8.

242. Steve Ditlea, *The Electronic Paper Chase*, SCI. AM., Nov. 2001, at 38–43.

243. Doon Barker, *Pixel Perfect: Silencing Critics . . . Microsoft Receives E-Book Patent*, TECH. REV., Jun. 2001, at 29; U.S. Patent No. 6,239,783 (issued May 29, 2001).

244. Jakob Nielsen, *Avoiding Commodity Status*, ALERTBOX, *at* http://www.useit.com/alertbox/20020203.html (Feb. 3, 2002).

This research is costly. According to Microsoft, its researchers spent more than two years sifting through a large amount of research related to both typography and the psychology of reading before even getting started.[245] Nor does Microsoft have the field to itself. Adobe Systems, the firm behind the ubiquitous PDF portable document format, has its own clear reading technology, Cool Type.[246]

Right now, ClearType technology is a nice feature in the Windows XP operating system, and Adobe Systems' latest products are similarly improved. But they are unlikely to be decisive in persuading users to invest in e-books. It will be perhaps ten years before all the technologies are in place to make e-books an attractive proposition for the average consumer. Screen technology is advancing on two fronts, hardware and software; both are equally important. If these essential software inventions are to take place, the innovating firms need to be able to share information in the same way and with the same kind of protection afforded to hardware makers.

An even more distant prospect is speech recognition technology.[247] Anyone who has used dictation software or conversed with an interactive voice response system will appreciate how immature speech recognition technology is today. Having just written a 380 page book using IBM's ViaVoice dictation software, I can assert it was just about worth the hassle.[248] When I first tried the software in 1995, I gave up after a few hours, but with each release the product got better and better. Much of this improvement is due to IBM's speech recognition research, for which it has obtained 170 software patents in the last 10 years.[249] IBM has been working on speech recognition for 25 years. Perhaps in another 25 years we will have computers without keyboards and voice response systems that really do listen, understand and speak. But this is by no means certain—speech recognition is a long term investment of uncertain rewards.[250]

245. *Microsoft Reader Uses ClearType Technology and Traditional Typography to Enhance On-Screen Reading*, Microsoft PressPass, *at* http://www.microsoft.com/presspass/features/1999/08-30seybold.asp (Aug. 30, 1999).

246. Paul Hilts, *Adobe Shows CoolType*, PUBLISHERS WKLY., Mar. 20, 2000, at 12; U.S. Patent No. 5,929,866 (issued Jul. 27, 1999).

247. For an accessible history of speech recognition, see Alan A. Andolsen, *Can You Understand Me Now?*, 38 INFO. MGMT. J. 52 (2004).

248. For another user perspective on ViaVoice, see Diane Brady, *The Single-Handed Reporter*, BUS. WK., May 12, 2003, at 90.

249. *See* IBM, *Human Language Technologies*, *at* http://www.research.ibm.com/hlt/html/patents.html (last visited May 8, 2005).

250. For a less than sanguine view on the current state of speech recognition, see Stephen Manes, *Good Enough Isn't*, FORBES, Oct. 28, 2002, at 310.

IBM is just one player in this hugely important field with many competitors, large and small.[251] Thousands of patents have been issued in the last 25 years: recognizers for continuous speech; recognition in the presence of noise; systems for high accuracy with a limited vocabulary; systems for acceptable accuracy with a wide vocabulary; language models and error correction techniques; and semantic analyzers to extract the meaning of utterances.[252] Dictation software that retails at fifty to a hundred dollars builds on these innovations. Hundreds of other speech recognition applications will eventually emerge.

## IX. THE SOFTWARE PATENT THICKET

> *Software patents, for instance, can protect a single line of code that tells a computer to do a specific task. This might include telling one computer program to activate another program.*
>
> *Such narrow slicing of software development can hinder invention of fully formed technologies, which often are built on the work of others, critics argue.*
>
> *. . . Small firms have an increasingly difficult time breaking through patent "thickets" amassed by large firms. International Business Machines Corp., the world's patent leader, received 22,357 from 1993 to 2002 and earned roughly $10 billion in licensing fees from them.*
>
> —Jonathan Krim[253]

There is widespread concern in SMEs and the open-source community of inadvertently infringing on a patent, thereby rendering oneself open to a ruinous lawsuit. The fear is somewhat exaggerated. I know of no patent that protects "a single line of code," cited in the above quotation, but perhaps it is a theoretical possibility. Additionally, while IBM does earn significant royalties on its patents, royalties are earned on both hardware and software patents, predominantly the former.[254]

251. Competitors in the speech recognition market include AT&T, Hitachi, Microsoft, Philips, Siemens, Xerox, and dozens of others, large and small. *See* Savitha Srinivasan & Eric Brown, *Is Speech Recognition Becoming Mainstream?* COMPUTER, Apr. 2002, at 38.

252. *See generally id.*

253. Jonathan Krim, *Patenting Air or Protecting Property? Information Age Invents a New Problem*, WASH. POST, Dec. 11, 2003, at E1.

254. Press Release, IBM Corp., IBM Tops U.S. Patent List for 8th Consecutive Year (Jan. 10, 2001), *at* http://www-1.ibm.com/press/PressServletForm.wss?MenuChoice=pressreleases&TemplateName=ShowPressReleaseTemplate&SelectString=t1.docunid=1433&

The LZW and RSA patents discussed in section 6 were atypical in that they were major innovations well known to the great majority of software practitioners working in the relevant field. However, as with other technologies, in practice most patents are granted for incremental innovations that do not in themselves amount to very much. In the aggregate, however, they can transform a product. The patented improvements for spelling and grammar checking in Microsoft Word are a typical example.

Consider what happens when I type the following (admittedly somewhat contrived) sentence:

"its goin to be gorilla warfare in teh pyrenees" said henry.

As Word struggles to make sense of this sentence, several patented innovations are called into play.[255] Automatically, the lower case *i* in *its* and the lower case *p* in *pyrenees* are capitalized; *teh* is corrected to *the*; and the quotation marks are converted to "smart" quotes. In addition, a wavy red line appears under the words *goin* and *henry* and a wavy green line under *gorilla*. All this occurred in the "background" without me having to invoke proofing tools. The "autocorrect" and smart quotes features are protected by patent 5,787,451 and background checking is protected by patents 5,787,451 (the '451 patent) and 5,761,689. Right clicking over the word *goin* gives a menu of five correct spellings to choose from, of which *going* is the first—I select it. Likewise I change *henry* to *Henry*. This innovation, which places candidate corrections in order of likelihood based on the user's typing history, is protected by patent number 6,377,965. Having done this a wavy green line now appears under *its*; a right click makes the correction to *it's*. The wavy green line under *gorilla* indicates a grammar or semantic error; right clicking offers the alternative *guerrilla*. This is an impressive innovation. The word *gorilla* is found to be incorrect because of its surrounding context; this feature is protected by patent 5,940,847, "System and method for automatically correcting multiword data entry errors."

At first sight, this looks an impressive achievement, and in many ways it is. But it is also revealing of the state of the art and how many more patented innovations will be needed to do a really good job. For example, although Word picked up the incorrect capitalizations of *pyrenees* and *henry*, it would not have corrected *china* or *martin*, in the same places, because they are nouns in their own right; nor would it have

TableName=DataheadApplicationClass&SESSIONKEY=any&WindowTitle=Press+Release&STATUS=publish (last visited May 8, 2005).

255. U.S. Patent No. 6,377,965 (issued Apr. 23, 2002); U.S. Patent No. 5,940,847 (issued Aug. 17, 1999); U.S. Patent No. 5,787,451 (issued Jul. 28, 1998); U.S. Patent No. 5,761,689 (issued Jun. 2, 1998).

objected to *farther* in place of Henry, when *father* might have been intended. It detects *gorilla warfare* but not *running saw*. Word's semantic checking has a long way to go.

These word-processor innovations are consonant with the kind of patented improvements that transformed the typewriter in the period 1890–1910. Indeed, it seems that Word may be rather like the typewriter, in that it will not be perfected in one fell swoop, but by the accumulation of hundreds of tiny improvements. Given Word's current shortcomings in the proof checking department and the irregularity of the English language, many more patented improvements will be needed before it achieves technical closure.[256] Incidentally, this relatively slow progress brings into perspective the commonly voiced concern that patents last too long relative to the speed of progress of the software industry. Word processing software packages are now about 25 years old, and innovation is not slowing. The speed of development seems to be not dissimilar to that of the typewriter a century ago.

Much criticism of minor software patents—such as those characterized above—is directed to their obviousness and lack of novelty. The matter of obviousness is partly a difference in perception between software practice and patent examination. A software practitioner thinks of a patent as being obvious if he or she can readily replicate the patented code. Whereas, a patent examiner would ask whether an individual "with ordinary skill in the art" could readily make the innovation, at the time the invention was made. That is, without the benefit of hindsight and without having seen the invention. Surely the practitioners as well as examiner would be satisfied with the nonobviousness of the RSA patent, but probably not the Microsoft Word patents. However, there is much more to the '451 patent, for example, than putting a wavy red line under misspelled words. Besides working in the background, using otherwise wasted processor power, the innovation keeps a record of which words have been spell checked, and when the file is saved the state of spelling correction is preserved along with the text so the state of the spelling checker persists between editing sessions.

It is often suggested that the sheer number of software patents being granted is evidence enough that the threshold of obviousness is too low. It may be true that there are too many obvious patents, but the number granted does not support the contention. In 2001 (the latest year for which there are statistics) out of a total of 166,000 patents issued,

256. On technical closure, see generally NATHAN ROSENBERG, INSIDE THE BLACK BOX: TECHNOLOGY AND ECONOMICS (1982).

20,000-plus software patents were granted.[257] By comparison, the number of patents in the major classifications chemical, electrical, and mechanical were approximately 46,000, 57,000 and 63,000 respectively.[258] Given that software is now one of the top U.S. industries—certainly in the same league as chemical, electrical and mechanical manufacturing—the number of patents is not self-evidently disproportionate.

Establishing novelty is also problematic. The patent examiner has three main sources for prior art: the patent database, the academic technical literature, and the ephemeral literature. Now that software patents have been granted in large numbers for a decade or more the patent literature is becoming a significant source of prior art. There is a comprehensive classification system, frequently updated, and a publicly accessible search engine to the patent database. For example, the '451 patent is in class 715/533: that is "data processing: presentation processing of document" subclass "spell check." There are not many patents to search through to establish prior art. The academic literature is also reasonably easy to search, much as in any other scientific discipline. However, the academic literature is not a good source for publications in practical areas such as spell checking, as the literature tends to focus on weightier topics like mathematical algorithms. This is a long-standing cultural schism in computer science, reflecting the subject's origins in mathematics departments; the academy tends to value theoretical results more than empirical. A topic such as spell checking is much more likely to be found in the ephemeral literature. This would include product literature, published conference proceedings, and university research reports. There is very little of this material readily available, even in the Library of Congress—indeed, not much survives anywhere for more than a decade after publication. In 1995, Professor Bernard Galler of the University of Michigan, a distinguished computer scientist, led the establishment of the Software Patent Institute which aims to collect the ephemeral literature and make it easily searchable.[259] This is a long and

257. The USPTO does not break out software patents in its statistical summaries. The estimate of 20,000-plus is widely used. *See, e.g.*, Robert Hunt and James Bessen, *The Software Patent Experiment*, BUS. REV. (Fed. Reserve Bank of Phila.), Third Quarter 2004, at 22.

258. *See* U.S. PATENT AND TRADEMARK OFFICE, ALL TECHNOLOGIES REPORT, JANUARY 1/1963–DECEMBER 31, 2001 (Mar. 2002); U.S. PAT. & TRADEMARK OFF., TECHNOLOGY ASSESSMENT AND FORECAST REPORT: CHEMICAL CLASSES, 1977–DECEMBER 2001 (Apr. 2002); U.S. PAT. & TRADEMARK OFF., TECHNOLOGY ASSESSMENT AND FORECAST REPORT: ELECTRICAL CLASSES, 1977–DECEMBER 2001 (Apr. 2002); U.S. PAT. & TRADEMARK OFF., TECHNOLOGY ASSESSMENT AND FORECAST REPORT: MECHANICAL CLASSES, 1977–DECEMBER 2001 (Apr. 2002).

259. *See* Software Patent Institute's website, *at* http://www.spi.org/ (last visited May 8, 2005).

difficult project, and its coverage is necessarily partial. The Software Patent Institute is typical of the way in which, historically, the patent system has adjusted to change.

It would be fair to say that determining the prior art is difficult for an experienced patent examiner or patent attorney, but next to impossible for a software practitioner. Worse, few software practitioners would think even to look.

Suppose a developer in an SME had seen Microsoft Word's spell checker and wanted to use something similar in his own program. (This is not an entirely hypothetical example—there are several clones of Microsoft Word.) First, the abiding culture of small-time software development is that if one has seen a feature one likes in another program, it is legitimate to use it in one's own program. This might surprise people in other industries. A pharmaceutical manufacturer would not suppose it was legitimate to copy the drug of another manufacturer simply because it was easy to do. Although the patent protects a particular drug, it is understood that the patent is just the end point of a long expensive road of discovery, fabrication, and testing with patients. In the same way, although a spell-check patent protects an apparently simple piece of code, it too is the end result of a process of discovery, fabrication, and user testing. However, it must also be said that there is a difference in degree between a spelling checker and a new antibiotic. This has not been reflected in the disproportionate damages awarded in some of the more celebrated software patent infringement suits—the recent $500 million *Eolas v Microsoft* suit being just the latest example.[260] The courts could do a much better job in assessing reasonable damages.

Suppose, however, that a small-time software firm wanted to be sure its product did not infringe on a patent. Where would it begin? One problem is that, unlike pharmaceuticals, software typically makes use of tens or hundreds of collateral inventions, the great majority of which are in the public domain, but a few might be patented.[261] Which ones? It is rarely obvious that a feature of a software product is patent protected. For example, the Microsoft Word program gives no hint that many of its features are patented (although it does have copyright notices for the various dictionaries and other components bundled with it). Unlike a large firm, an SME does not have the resources to make an extended search for prior art—a ten person firm does not have a legal department, and its developers are too busy cranking out code, trying

260. Robert A. Guth & Marcelo Prince, *Microsoft Faces $521 Million Verdict*, WALL ST. J. (Eastern edition), Aug. 12, 2003, at A3.
261. BESSEN & MASKIN, *supra* note 77.

to make a living in a very competitive industry. Indeed, a developer could implement a background spelling checker faster than he could determine whether or not he was infringing an existing patent. No wonder SMEs are hostile to software patents. They get in the way, and life was surely easier with trade secrets.

It is interesting that hardware manufacturers have little difficulty with the patent environment. A laptop manufacturer, for example, assembles finished goods from many sources, which are collectively covered by hundreds if not thousands of patents. However, when electronic components are sourced, any licensing fees for patents are bundled invisibly in the price. Likewise in the sub-components of components, and so on up the supply chain. Thus patent transaction costs for end manufacturers are negligible. By contrast, there is no meaningful supply chain for software. The industry is almost unique for the monolithic production of finished goods. Software components have been suggested as a solution to the perennial software crisis on many occasions, but the components market remains stubbornly undeveloped.[262] This is due not least to problems of IP appropriation for which patents have only just begun to be explored.[263]

Frustration with software patents by SMEs is partly a consequence of the unusual structure of the software industry. The industry is very low in concentration compared with other industries, such as pharmaceuticals, automobiles, or consumer electronics. In these industries there are typically fewer than a score of global players. These vast, vertically integrated firms created the consumer society of the twentieth century.[264] The same historical processes were also at work in the information technology industries. In the early 1900s there were about ten global typewriter manufacturers; in the 1960s there were eight mainframe computer manufacturers (IBM and the seven dwarves).[265] While these industries started with many entrants, they became fewer after consolidation and shake out. Software has done the opposite. In the mid 1960s there were perhaps a hundred software firms, and the number has steadily grown. Today there are at least 35,000 firms worldwide.[266] One reason for this phenomenon is that in typewriter and mainframe computer manufacture the capital requirements for entry

262. W. Wayt Gibbs, *Software's Chronic Crisis*, SCI. AM., Sept. 1994, at 86.

263. *See* Knut Blind & Jakob Edler, *Idiosyncrasies of the Software Development Process and Their Relation to Software Patents: Theoretical Considerations and Empirical Evidence*, 5 NETNOMICS 71 (2003); Lemley & O'Brien, *supra* note 14.

264. The rise of vertically integrated firms is best described in CHANDLER, *supra* note 79.

265. For typewriter firms, see *supra* note 76. For mainframe manufacturers, see FISHER, *supra* note 105, at 65–98.

266. *See supra* note 4.

were high, and became higher. In software, the entry costs have steadily declined. Today, all one needs is a PC and an Internet connection.

The extreme fragmentation of the software industry may not be optimal from an economic viewpoint. The industry structure is certainly very unusual. There has been considerable consolidation and Merger & Acquisition activity, resulting in a small number of global players—such as Microsoft, Oracle and Computer Associates—but at the same time there remains an enormous number of very small players, and their number is growing. Indeed, the software industry's structure has some parallels with the retail sector, where a few giant firms, such as Wal-Mart and Sears, co-exist with tens of thousands of mom-and-pop corner stores. While the giant retailers bring economic efficiency, we are all aware of the negative social consequences. By and large, in the United States the policy has been non-interventionist, in both retailing and software—let the market decide. This is not true in Europe, for example, where lobbying by the open-source community and SMEs has caused legislation on software patents—favored by large firms such as IBM and Siemens—to stall. Software patents clearly have a political dimension: Europe has only one global software player, but thousands of SMEs.[267]

It may be that the lack of concentration in the software industry is a reflection of its immaturity. Twenty or fifty years from now the industry may be much more concentrated, with a score of global players and relatively few SMEs. Whether or not this comes to pass, today's patent system should be determined by the broad needs of society, not the special interests of the producers—whether they be global power houses or shareware cooperatives.

## CONCLUSION

It is ten years since software patents have been issued in large numbers. The anxieties expressed in the early 1990s about the effect patents would have on the software industry have not been realized. History shows us that software patents are not so different from other patents in the information technology industries, and that the patent system is capable of adjusting to the particularities of individual industries. For example, early in the last century chemical processes were thought to be unpatentable, but the system soon adapted to a new real-

267. *See* BAKELS & HUGENHOLTZ, *supra* note 137; Andy Reinhardt, *Inventing a Better Patent Law: Can Europe Spur Software Innovation while Safeguarding Intellectual Property*, BUS. WK., Dec. 1, 2003 *at* http://www.businessweek.com/print/magazine/content/03_48/b3860058_mz054.htm?chan=mz&.

ity and now it is difficult to imagine this issue was once controversial. With software and business method patents the US Patent and Trademark Office is instituting changes that will make the system work better. For example, it has increased the number and quality of software patent examiners, and in time the databases and searching mechanisms for prior art will improve. It is also likely that cross-licensing, and patent pools and packages will mitigate the thicket problem.[268] In some cases firms have made their patents available free of royalty to open source developers.[269]

To offset the disadvantages of patent thickets, history shows that there are significant benefits of patents. Notably, the disclosure required by a patent specification has the potential to mitigate the trade secrecy that has been endemic to the software industry for most of its 50 year history. Today there is a huge economic loss arising from the constant reinvention that takes place in the software industry. It is possible that one day the industry will be based on the assembly of software components instead of always writing software from scratch. Such software reuse is widely practiced within individual firms in the software industry.[270] Patents would enable such software reuse to be externalized.

However, it is in the orderly development of broad prospects that the greatest benefits of software patents may come. In section 8 examples were given of how software patents have enabled the coordination of research by many players in the fields of screen technologies and speech recognition. These are just two of the problem domains that need to be conquered. We are at the very beginning of the information age, and other critical problems include Internet security (particularly the elimination of viruses and spam email), truly intelligent search engines (that can, for example, search images as well as text), and the whole realm of artificial intelligence. If private industry is to invest in developing these technologies it will do so more

268. For example, a patent pool MPEG LA has been created by a group of firms for MPEG video compression technology. The co-operating organizations include Columbia University, Electronics and Telecommunications Research Institute of Korea (ETRI), France Télécom, Fujitsu, LG Electronics, Matsushita, Mitsubishi, Microsoft, Motorola, Nokia, Philips, Robert Bosch GmbH, Samsung, Sharp, Sony, Toshiba, and Victor Company of Japan (JVC). Press Release, MPEG LA, *MPEG LA Announces Terms of Joint H.264/MPEG-4 AVC Patent License* (Nov. 17, 2003), *at* http://www.mpegla.com/news/n_03-11-17_avc.html.

269. For example, since participating in open-source development IBM has licensed, on a royalty-free basis, patents used in its contributions, a policy that has been endorsed by the Open Source Initiative. IBM Europe, *Response to the Services of the Directorate General for the Internal Market: The Patentability of Computer-Implemented Inventions* (Consultation Paper, Oct. 19, 2000), at 8, *at* http://europa.eu.int/comm/internal_market/en/indprop/comp/ibm.pdf.

270. *See* Lemley & O'Brien, *supra* note 14.

willingly with greater assurance for protection of their investment. Trade secrecy is antithetical to cooperation, while copyright is wholly inadequate in this context. In this regard, patent protection for software inventions may be the motivational force required to encourage this innovation and cooperation.