Marquette Intellectual Property Law Review

Volume 9 | Issue 2 Article 3

Summer 2005

A Technical Critique of Fifty Software Patents

Martin Campbell-Kelly
Warwick University

Patrick Valduriez
INRIA, National Center for Computer Science in France

A TECHNICAL CRITIQUE OF FIFTY SOFTWARE PATENTS

MARTIN CAMPBELL-KELLY* & PATRICK VALDURIEZ**

I. INTRODUCTION

There has been a great deal of discussion on the desirability or otherwise of software patents in the legal, economic, and technical academic literature. All of this literature has some limitations that we seek to overcome in this article.

The legal literature is for the most part based on case law.[1] As a result, discussions are often highly particular, focusing on the nuanced interpretation of particular cases as they percolate through the legal system. An obvious limitation of this literature is that it suffers from the small sample problem—it is difficult to infer universal lessons from individual cases. Of course, the legal literature does cover legal precedents, which give guidance to likely rulings in the future, but the small sample problem remains.

The economic literature on patents is typically based on the statistical analysis of large numbers of patents.[2] This overcomes the small sample problem. The economic literature is particularly useful for assessing the overall costs and benefits of patents, but the *technological*

* Martin Campbell-Kelly is a professor in the Department of Computer Science, Warwick University. He is an 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).
** Patrick Valduriez is a director of research at INRIA, the national center for computer science in France, working on data management in distributed systems. His most recent book is the second edition of PRINCIPLES OF DISTRIBUTED DATABASE SYSTEMS, co-authored with Tamer Özsu.

1. *See, e.g.*, ROBERT P. MERGES ET AL., INTELLECTUAL PROPERTY IN THE NEW TECHNICAL AGE 855-987 (3d ed., 2003); Jeffrey R. Kuester & Ann K. Moceyunas, *Patents for Software-Related Inventions*, 2 ITERATIONS: AN INTERDISC. J. SOFTWARE HIST. 1-14 (2003).

2. *See, e.g.*, JAMES BESSEN & ROBERT M. HUNT, AN EMPIRICAL LOOK AT SOFTWARE PATENTS, (Fed. Reserve Bank of Phila. Working Paper No. 03-17/R, 2004); Starling Hunter, *Have Business Method Patents Gotten a Bum Rap? Some Empirical Evidence*, (Center for eBusiness@MIT, Paper 182, 2003).

arguments for or against software patents tend to recede into the background.

The technical literature—frequently hostile to patents—is typically based on an examination of a small number of pathologically "bad" patents.[3] The Amazon.com 1-click patent, for instance, featured in the recent literature. However, the small sample problem makes it difficult to draw general conclusions. For example, are these "bad" patents typical of all patents, or are they simply rogues?

This paper focuses on the technical merits of what, on the surface at least, should be a set of fifty "good" patents. Our approach has been to conduct a detailed technical examination of all the patents in the set. This is a medium-sized sample that we believe is sufficiently large for some general lessons to be drawn. Our analysis, while technical (written as it is by a computer scientist and a computer historian), is nonetheless geared at policy dimensions of the current patent system. In other words, all of our inquiries focus on what insights technical specialists might provide lawyers, economists, and policymakers studying software patents.

We have chosen as our set of "good" patents the fifty most cited software patents (our precise criteria for inclusion are detailed in the next section). We believe that these are therefore good patents, in that they have been sufficiently useful or prominent to attract a large number of citations. They may even represent a proxy for best practice in patent applications, at least in a limited sense. We did consider using a set of randomly chosen patents, but we felt that such a set would offer no guidance or benchmark, because they would not conform to either best or worst practice.

We realize that our approach cannot address all of the controversies about software patents. For example, we have not been able to comment usefully about the risk of inadvertent infringement. However, we do think we can make a useful commentary on the following questions that are frequently discussed in the software-patent literature:

Are software patents too obvious?

Is the level of disclosure adequate?

Are patents "real"—that is, do they represent real innovations or are they largely "strategic"?

Are software patents too broad?

3. *See, e.g.*, Richard Stallman & Simson Garfinkle, *Viewpoint: Against Software Patents*, 35(1) COMM. OF THE ACM 17-22 (1992); James Gleick, *Patently Absurd*, N.Y. TIMES MAG., Mar. 12, 2000, at 44.

Do software patents last too long?

Is copyright protection also required for software inventions?

These questions are each discussed in turn, below. For this set of patents, we conclude:

These software patents, in general, are not too obvious.

The level of disclosure is often less than optimal, indicating the need for reform.

Almost all were issued for real innovations.

Most of the patents are not too broad.

Software patents last too long, but this is not a critical issue.

Some software inventions need both patent and copyright protection.

II. SAMPLE SET OF THE FIFTY MOST CITED SOFTWARE PATENTS

A. *How the Patents Were Identified*

The patents considered in this paper are the product of two patent datasets. The first is directly from the United States Patent and Trademark Office (USPTO) and consists of all patents granted from 1976 through 2000 that have the International Patent Classification (IPC) G06F.[4] The second dataset is from the National Bureau of Economic Research (NBER)[5] and presents citation data for all patents from 1963 through 1999. The citation data from NBER was used to provide a list of all patents that cited the patents from our USPTO dataset. The list of fifty patents is the combination of two separate measures of quality.[6]

The first measure, which makes up forty-one of the top patents, is based on a raw-count of forward citations, not including forward citations from an assignee's own patents.[7] Forward citations are

4. For information on the IPC, see *International Classifications at WIPO*, WIPO, *at* http://www.wipo.int/classifications/fulltext/new_ipc/ (last visited Apr. 3, 2005). G06F covers "Electric Digital Data Processing." *Id.* at http://www.wipo.int/classifications/fulltext/new_ipc/index.htm. Arguably, this classification can be considered as a proxy for identifying software patents.

5. *See* Bronwyn H. Hall, Adam B. Jaffe & Manuel Tratjenberg, *The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools* (NBER Working Paper 8498, 2001), *available at* http:// www.nber.org/patents (last updated Dec. 1, 2003).

6. *See infra* Appendix B for details on each of the fifty patents.

7. Forward citations were classified as self-citations based on a comparison of Committee on Uniform Securities Identification Procedures (CUSIP) IDs from the NBER dataset. The CUSIP to assignee match is based on the universe of companies from 1989. If

frequently taken as a measure of a patent's influence on subsequent innovations.[8] The second measure, which makes up nine of the top fifty, is similar to the first, but limited to patents granted in or after 1990 and non-self-forward citations within three years of the patent grant. There were two reasons for including these latter patents. First, citations naturally increase with time, so the older patents would tend to be over-represented. Second, the more recent patents are more controversial—they are patents that were issued after the series of court rulings that opened the door wider to software patents. By limiting the sub-sample to 1990 and later, we get a contrast with older patents and also include more of the controversial ones.

The software patents from the USPTO dataset were sorted based on the two sets of quality criteria listed above. Because the original criterion for selecting software patents (IPC G06F) is a somewhat blunt instrument, there were patents in the dataset that probably should not be classified as pure software. Individual analysis of the top patents from each quality criteria resulted in the de-classification of several (about thirty) patents that were deemed hardware/firmware.[9]

B. Categorizing the Patents

Similar to research papers in computer science that describe an invention, a software patent can be anything between very broad, for example, describe a new graphic user interface system, and very deep, for example, describe a new programming language optimization technique. The international patent classification system is useful for key-word searches of patents but too fine to help the analysis of our sample of patents from a technical point of view. Therefore, we propose a simple classification of software patents along two dimensions,

---

the assignee for the patent receiving or giving the citation had no corresponding CUSIP, the citation was not classified as a self-citation. This approach slightly over-counts the number of qualifying forward citations.

8. *See, e.g.*, Adam B. Jaffe & Josh Lerner, *Reinventing Public R&D: Patent Policy and the Commercialization of National Laboratory Technologies*, 32 RAND J. ECON. 167 (2001); Daniel K.N. Johnson & David Popp, *Forced Out of the Closet: The Impact of the American Inventors Protection Act on the Timing of Patent Disclosure*, 34 RAND J. ECON. 96 (2003); Manuel Trajtenberg, *A Penny for Your Quotes: Patent Citations and the Value of Innovations*, 21 RAND J. ECON. 172 (1990).

9. The boundary between hardware and software is blurry since software cannot be made fully independent of hardware. For instance, to bootstrap, or "boot," an operating system, a processor needs to execute special instructions that are built into the hardware. Firmware, or microcode, which consists of computer code stored in programmable read-only memory (ROM), designates such special-purpose software. Thus, firmware patents can mix hardware and software inventions thereby making their classification and evaluation difficult.

software levels and granularity levels, which we apply to our sample of patents. Thus, we obtain four "classes" of patents. For each class of patents, we further group patents by technology such as CAD (computer-aided design), CASE (computer-aided software engineering), GUI (graphical user interface), database, middleware, etc. For each patent, we consider various detailed technical indicators such as numbers of pages, block diagrams, flowcharts, code skeletons, and code fragments, which are useful to assess technical depth and disclosure level.

Software is a generic term that designates data and code that establish computer processes. Software can be divided in two major categories: system software that provides the basic application-independent functions of the computer and application software that provides application-specific functions. A user can typically use system or application software functions to perform the desired tasks.

System software provides hardware-independence to system users and other software so that they are not concerned with low-level details. It is responsible for controlling and managing the individual hardware components of a computer system. Typically, system software designates an operating system (such as GNU/Linux, Unix or Windows) that can be further decomposed in an operating system kernel (with basic functions) and operating system services such as user authentication, display management, file management, and network management. Middleware, which provide basic services for the interoperability of heterogeneous software programs, is also a form of system software.

Application software runs on top of system software and provides application-specific functions to users or other independent applications. Examples of such software are text-processing, spreadsheet, database, GUI, enterprise resource planning (ERP) applications, or end-user applications (programmed by those using application or system software). Interestingly, the use of business components (reusable elements that implement business concepts or processes), which users can either buy out or create themselves, makes the boundary between vendor-supplied packages and user-developed software blurry.

Even with this simple classification, some patents are difficult to classify between application and system software. For instance, middleware that provides advanced services such as distributed transactions can be considered application software.

Software may consist of anything from one program (or component) to many programs (or components) and a patent can bear on parts of that software. To ease analysis, we consider only two granularity levels[10] for software patents: fine-grain (one to a few components) and large-grain (several to many components). For instance, a technique to optimize programming language or an image viewer is fine-grain while an application suite (such as Microsoft Office or Star Office) or a document management system is large-grain. Fine-grain patents are likely to describe a particular technique for a specific problem and involve data structures and associated code. Large-grain patents are broader and can describe an entire system. They primarily involve interfaces and possibly data structures and code, but perhaps not at a very deep level of detail.

Table 1 in Appendix A gives our classification of our sample set of the fifty most cited patents. For each combination of software/granularity level, we further classify patents by technology. Then for each technology, we list each patent by the assignee company in decreasing order of number of citations. Note that some patents have not been filed on behalf of a company, in which case we simply mention the inventor's name with the indication "ind." for individual.

Distinguishing between application and system software patents was sometimes difficult. For instance, most of the patents in communication, multimedia, and security technologies include elements that are system software (or even have some hardware design for the multimedia patents). This is the case for IBM's 5,333,266 patent,[11] but a detailed analysis reveals that many elements are application-specific, in particular, to deal with specific media types.

Our classification clearly shows that the vast majority of patents in our sample set are in application software (thirty-nine patents) rather than in system software. And most of these (thirty-seven) are large-grain. It is noticeable that the highest numbers of patents are in GUI (ten patents) and database management (eight patents), which corresponds to the impact of object-oriented programming on these technologies. Large-grain patents are broad and focus on interfaces

10. Initially, we started with three levels, small-grain, medium-grain, and large-grain, but when applied to our sample set, it turned out that the difference between medium-grain and large-grain was quite arbitrary, the difference dependant upon the software technology. For instance, the same software could be classified as medium-grain if described using modules or large-grain if described using object classes.

11. This patent is classified as G06F 13/00: "Interconnection of, or transfer of information or other signals between, memories, input/output devices[,] or central processing units." *International Classifications*, *supra* note 4, at http://www.wipo.int/classifications/fulltext/new_ipc/index.htm.

between components rather than on component implementation details. By contrast, fine-grain system software patents focus on implementation details.

A more detailed technical analysis of the patent dataset is given in Appendix B.

III. ARE SOFTWARE PATENTS TOO OBVIOUS?

Much of the criticism about software patents focuses on examples that are perceived to be trivial by programming professionals. The Amazon.com 1-click patent perhaps gets more attention than any other in this regard, but about a dozen patents are commonly cited.[12] However, although these patents are cited in order to condemn software patents in general, in reality they represent an infinitesimal fraction of the one hundred thousand-plus software patents that have been issued. In this section, we will examine our set of patents from the perspective of a software practitioner to see if they suffer from the same weaknesses as those characterized as trivial by critics of software patents.

One reason for the debate over obviousness is the common misconception about the level of originality that is necessary for a patent to be granted. People's expectation of inventiveness is distorted by famous patents such as Edison's for the incandescent light bulb[13] or Ted Hoff's for the first microprocessor.[14] Such "foundation" patents are very rare. The fact is that the great majority of patents are issued for small incremental improvements of existing technology. The vast majority of patents are obtained by research workers simply doing their job, making incremental improvements.

The distinguished economist, Richard Nelson, has observed that software development shares characteristics with the typical industrial research and development that traditionally receives patent protection:

> Most industrial design and development work, like software creation, involves professionals putting in effort, skill, and often some creativity, but it generally does not result in a design that professionals would regard as "non-obvious." In my opinion, many of the patents granted in industrial research and development are on "inventions" that result from professionals doing their jobs in ways that other professionals would if

12. *See* Stallman & Garfinkle, *supra* note 3, at 18. For a rejoinder see Paul Heckel, *Debunking the Software Patent Myths*, 35 COMM. OF THE ACM 121 (1992).

13. U.S. Patent No. 223,898 (issued Jan. 27, 1880).

14. U.S. Patent No. 3,821,715 (issued June 28, 1974).

> assigned the same task. Since software development does not differ significantly in this regard from much of industrial research and development, then if patents are appropriate for many of the modest routine advances of the latter, they may not be inappropriate on this count alone for protecting the former.[15]

Thus—to borrow from Edison—the patent system rewards "perspiration as well as inspiration."[16]

Much invention is simply the systematic searching for a solution to a problem. One of the most famous examples is Edison's search for a filament for his light bulb. He and a team of assistants spent two years trying out hundreds of different substances for a suitable filament, eventually settling on carbonized bamboo that combined acceptable light output with acceptable longevity.[17] Christopher Latham Sholes, the inventor of the typewriter, spent five years and built fifty prototypes before settling on the design that became the Remington typewriter.[18] The same process occurs in the invention of new drugs, where the laborious process of searching for candidate therapies is additionally followed by a long and expensive process of regulatory approval.

The process of software invention is similar; programmers who have a problem to solve search for a solution, systematically or serendipitously. Software is an unusual technology in that it is extremely malleable, so that it is possible for a programmer to generate and discard candidate solutions to a problem very quickly. This rapidity of prototyping makes software development seem superficially easy, but the process is not fundamentally different to other technologies.

A well-documented example of the rapid development of candidate solutions is the spreadsheet. In 1978, the inventors Dan Bricklin and Bob Frankston spent just a few months systematically tweaking and refining a crude prototype into VisiCalc.[19] They perfected the menu

15. Richard R. Nelson, *Intellectual Property Protection for Cumulative Systems Technology*, 94 COLUM. L. REV. 2674, 2675 (1994). Nelson, no fan of the patent system, continued: "On the other hand, one could argue that patents granted on inventions coming from industrial research and development that do not meet the 'nonobviousness' standard should not have been granted." *Id.*

16. John A. Gibby, *Software Patent Developments: A Programmer's Perspective*, 23 RUTGERS COMPUTER & TECH. L.J. 293, 352 (1997) (quoting Arthur R. Miller & Michael H. Davis, INTELLECTUAL PROPERTY IN A NUTSHELL: PATENTS, TRADEMARKS AND COPYRIGHT 84 (1990)).

17. PAUL ISRAEL, EDISON: A LIFE OF INVENTION 196-97 (1998).

18. BRUCE BLIVEN, JR., THE WONDERFUL WRITING MACHINE 48 (1954).

19. Interview by Martin Campbell-Kelly with Dan Bricklin & Bob Frankston (May 6, 2004) (transcript forthcoming at *Oral Histories*, Charles Babbage Institute, *at*

structure, the on-screen presentation of data, and the tightly coupled interaction between user and machine.[20] Given the idea of a computerized spreadsheet, most competent programmers would eventually have come up with a competent solution. It is in this light that obviousness needs to be assessed.

Because software progress is cumulative, it may be hard to evaluate the degree of innovation (or obviousness) of a patent by itself and with respect to the state-of-the-art (i.e., other patents or published papers). Although this has been the subject of much debate, this issue is much the same as in academic research in computer science when papers are selected for publication in journals and conferences after peer review based on a combination of criteria such as originality, significance, technical soundness, etc. However, we observe that there are many more papers produced in computer science today than ever. To cope with increasing numbers of paper submissions (and thus potential inventions), the peer review process in computer science has become quite selective and rigorous. An interesting issue is distinguishing between "breakthrough" papers that propose a very novel idea but with minimal detail and validation, and "delta" papers that propose a slight improvement over a known technique in great detail and extensive evidence of superiority. A good example of a breakthrough paper is Edgar F. Codd's presenting the relational data model in 1970, then an IBM researcher.[21] This ten-page paper gave database management a mathematical foundation that revolutionized the entire database industry.[22] It took about ten years of intensive research, mostly disseminated in fundamental papers and delta papers, and development to produce the first relational database management system (DBMS)

---

http://www.cbi.umn.edu/collections/oralhistories.html (last updated July 8, 2003). VisiCalc was the first computer spreadsheet. *See* VisiCalc *at* http://www.bricklin.com/visicalc.htm (last visited Apr. 3, 2005).

20. Bricklin and Frankston wanted to patent VisiCalc, but software patents were then not routinely granted and they were advised that an application was unlikely to be successful. *See* Martin Campbell-Kelly, *The Rise and Rise of the Spreadsheet*, *in* FROM SUMER TO SPREADSHEETS: THE HISTORY OF MATHEMATICAL TABLES 322 (Martin Campbell-Kelly et al. eds., 2003).

21. Edgar F. Codd, *A Relational Model of Data for Large Shared Data Banks*, 13 COMM. OF THE ACM 377 (1970). In 1981, Dr. Codd received the A.M. Turing Award for his invention of the relational model. *See A.M Turing Award*, Association for Computing Machinery, *at* http://www.acm.org/awards/taward.html (last updated Feb. 16, 2005). In the field of computer science, the Turing Award is equivalent to the Nobel Prize.

22. David Mindell, *The Rise of Relational Databases*, *in* FUNDING A REVOLUTION: GOVERNMENT SUPPORT FOR COMPUTING RESEARCH 159 (National Research Council ed., 1999).

products such as Oracle and IBM DB2. This ten-year period is interesting because it corresponds to the same period the major database research conferences used when granting awards for the most influential inventions.[23] It took another ten years to improve the relational technology and the products, mostly through delta papers that improved the initial inventions.

Depending on how software patents are written, we may be able to apply a similar reviewing process and distinguish between foundation patents and incremental patents. The intense synergy between academic research and industry research, as evidenced by the strong participation (as program committee members, authors and participants) of researchers of major computer companies in computer science conferences, should ease this.

A significant finding is that all patents in our set are incremental. On average, each refers to about ten previous patents. Most of the patents also refer to published research papers and books, from which the inventors exploited theoretical results from academic research. However, some patents are less incremental than others are. For instance, the most-cited patent[24] in our sample set describes a software-version-management system for enabling the automatic recompiling of updated versions of component software objects over a computer network. This patent was very novel; it referred to only one previous patent and several papers and books on source-code-version management. On the other hand, another patent[25] refers to forty previous patents and many technical papers on electronic fund transfer.

Only two patents in our set could be considered technically obvious. The first describes a system to assist a user in purchasing goods or services sold by several vendors.[26] The system consists of several application programs that access a database containing information about different products and/or services. The system also provides the user with an interface to interact with the database and help refine the

23. For example, conferences organized by the ACM Special Interest Group on Management of Data (SIGMOD), see *SIGMOD Conferences*, ACM SIGMOD Online, *at* http://www.sigmod.org/sigmod/conferences/index.html (last visited Apr. 3, 2005) or the Very Large Databases (VLDB) Endowment, see *Very Large Data Bases (VLDB) Conference*, VDLB, *at* http://www.vldb.org/dblp/db/conf/vldb/index.html (last updated Mar. 2, 2005).

24. U.S. Patent No. 4,558,413 (issued Dec. 10, 1985). The patent is assigned to Xerox Corporation.

25. U.S. Patent No. 5,220,501 (issued June 15, 1993). The patent is assigned to Online Resources.

26. U.S. Patent No. 4,992,940 (issued Feb. 12, 1991). The patent is assigned to H-Renee, Inc.

searching of products. Finally, after the user has selected one or more items for immediate purchase, the system automatically transmits the order to the appropriate vendor. For prior art, the patent refers to four previous patents and one research paper.

From a technical point of view, this patent is relatively obvious being merely a straightforward purchasing application that uses a database. It is, however, less obvious than the Amazon 1-Click patent, which is not considered particularly obvious by everyone.[27] Indeed, Tim O'Reilly, founder of O'Reilly & Associates, the technical publisher who famously spearheaded a campaign against the 1-Click patent, eventually conceded, "that Amazon's patents might actually hold more water as original inventions than I thought."[28] O'Reilly still thought Amazon's patents were "a bad idea," but not, presumably, on the grounds of obviousness or triviality.

The second technically trivial patent describes a method for browsing the Web using an HTML browser.[29] While the client waits for a reply or as the document is being downloaded, the browser displays a "mini-Web page" with informational messages to the user, e.g., advertisements, notices, messages, or copyright information. Such information is added as comments attached to the links and thus ignored by the browser when displaying the document. However, the idea of embedding information in HTML comments is obvious and has been already used in other document description languages.

IV. IS THE DEGREE OF DISCLOSURE ADEQUATE?

A patent is a bargain between society and the inventor. In exchange for a temporary monopoly, the inventor discloses the invention. The purpose of this disclosure is two-fold. First, it enables other firms or individuals to make use of the invention when the patent has expired. Second, it enables competitors to see the patented invention, so that

27. *See* P. Michael Nugent, *Financial Business Method Patents—Nothing New under the Sun*, *in* COPY FIGHTS: THE FUTURE OF INTELLECTUAL PROPERTY IN THE INFORMATION AGE 229, 232 (Adam Thierer & Clyde Wayne Crews, Jr. eds., 2002); 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, 257-58.

28. Tim O'Reilly, *Amazon's Patent Reform Proposal*, O'Reilly Network, *at* http://www.oreillynet.com/pub/a/oreilly/ask_tim/2000/patent_reform_0300.html (last visited Apr. 3, 2005). *See also*, Tim O'Reilly, *The Internet Patent Land Grab*, 43 COMM. OF THE ACM 29 (2000), *available at* http://tim.oreilly.com/articles/cacm3.html (summarizing subsequent research by O'Reilly regarding the Amazon 1-Click patent debate).

29. U.S. Patent No. 5,572,643 (issued Nov. 5, 1996).

they can seek to improve on the invention or invent around it; in either case, society will benefit from the evolution of an improved product.

It is a requirement of the Patent Act that the disclosure should encapsulate the "best mode" of realizing the invention, and that this realization should be possible without "undue experimentation."[30] In determining whether these requirements have been met, the concept of the "person having ordinary skill in the art," sometimes known by the acronym, "PHOSITA," is invoked.[31] In the case of a software patent, this is understood to mean a time-served journeyman programmer, neither a rank amateur nor a programmer of exceptional skill.[32]

Clearly, software patents that include the full source code of an invention are likely to fulfill the disclosure requirement. However, disclosure in the form of a high-level functional specification, block diagrams, or flowcharts might require a great deal of work or experimentation. For this reason, it has been argued that the disclosure of source code should be mandatory.[33] Apart from the impracticality of appending one-million lines of source code to a patent, this would in fact represent a degree of disclosure far beyond the norm for other kinds of patents.

In order to gain an insight into what might be an appropriate degree of disclosure in software patents, let us consider a patent for a much older information technology, the classic Underwood typewriter that became the dominant design for the machine-writing industry in the early years of the last century. The inventor in this 1896 patent "Type Writing Machine" was Franz X. Wagner, who assigned it to the Underwood Corporation.[34] Implicit in the disclosure in the patent was the fact that several hundred typewriter patents already existed, with thirty firms in the typewriter industry, and writing machines were mass-produced items. Hence, the implicit capabilities of the PHOSITA were not those of an amateur with a home workshop, but those of an experienced practitioner in the industry.

30. Lawrence D. Graham & Richard O. Zerbe, Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure*, 22 RUTGERS COMPUTER & TECH. L.J. 61, 96 (1994) (citing 35 U.S.C. § 112 (1994)).

31. For the etymology of PHOSITA, see Dan L. Burk & Mark A. Lemley, *Is Patent Law Technology-Specific?*, 17 BERKELEY TECH. L.J. 1155, 1185-86 & n.126 (2002).

32. *See id.* at 188.

33. Graham & Zerbe, *supra* note 30, at 138.

34. U.S. Patent No. 559,345 (issued Apr. 28, 1896). For an in-depth discussion of the patent history of typewriters, see Martin Campbell-Kelly, *Not All Bad: A Historical Perspective on Software Patents*, 12 MICH. TELECOMM. & TECH. L. REV. (forthcoming 2005).

An examination of the Wagner patent shows that it contains a detailed functional description of the typewriter, which includes sixteen drawings over six pages and a five-page technical narrative of the operation of its working parts. Hence, the reader of the patent would certainly be able to clearly understand how the typewriter worked and appraise all of its constituent parts. However, for three reasons it would be a very long step to the best-mode recreation of the machine.

First, the patent does not contain remotely sufficient technical data to go into production without a great deal of experimentation. Although the drawings convey the functionality of the typewriter, they are not in any sense the machine drawings that would be needed to manufacture the article; for example, there are no linear or radial dimensions, and not every component is clearly illustrated. Second, many details cannot be easily communicated in a machine drawing, for instance, the type of metal used in levers and type bars and the tension of springs. It is true that some of these desiderata could be determined by disassembling an actual typewriter, but such reverse engineering is not a presumption of granting a patent. Third, there is an implicit need for sophisticated assembly techniques. A typewriter cannot simply be assembled from of a bundle of parts like a child's constructor set; specialized jigs are needed to hold the dozens of type bars in place, to align the type, and adjust spring tensions.[35]

Therefore, in the case of the typewriter, there was no effortless route leading from the patent to the artifact. Specialized tools and a degree of experimentation were both necessary. Nonetheless, the level of disclosure conformed to the contemporary norms of the Patent Office and the typewriter industry. Those who advocate that a software patent should include a listing of source code are in fact advocating a disclosure requirement much greater than for other patents.[36] If the source code for a word processing program, the modern equivalent of a typewriter, was wholly disclosed, then it would be possible for an individual to go from patent to artifact with less than ordinary skill in the art and without any experimentation or specialized equipment.

The optimal level of disclosure in a software patent can vary from a high-level architectural description, through block diagrams and

35. If the reader doubts this, try taking apart a typewriter and putting it back together again! For a discussion of typewriter manufacture, see DONALD R. HOKE, INGENIOUS YANKEES: THE RISE OF THE AMERICAN SYSTEM OF MANUFACTURES IN THE PRIVATE SECTOR 132-78 (1990).

36. Thomas P. Burke, *Software Patent Protection: Debugging the Current System*, 69 NOTRE DAME L. REV. 1115, 1158 (1994).

flowcharts, to source code. Different software inventions call for different forms of disclosure.. For example, a one-million-line word processing program would probably be unintelligible if it was disclosed only through source code; it would be more effectively described in terms of a high-level architecture with block diagrams and flowcharts for its detailed features. However, in a fine-grained software component—such as data compression algorithm—the source code would be its essence and it should therefore be included to disclose fully the invention.

It would also seem reasonable to assume that the PHOSITA for a software invention is something more than an amateur programmer equipped with nothing more than a PC and an Internet connection. A time-served, journeyman programmer wanting to reproduce the invention would likely have access to computer-aided software engineering (CASE) tools of the kind used in industry. These expensive and powerful tools provide facilities for writing software from a specification, complementary code libraries, and the means for experimental testing and code integration.

Good indicators of disclosure level of a software patent are its page length, the number of figures, block diagrams, flowcharts, and tables, and appendices (not included in the page count). A high number of pages typically suggests a detailed description, but it may also suggest that the technical breadth of the patent is wide (for example, a DBMS kernel). However, a low number of pages may be enough for disclosure if the patent is of narrow technical breadth (for example, one that described a basic sorting technique). Furthermore, some patents may have very long appendices with lots of details. For instance, a patent assigned to Persistence Software is only twenty-four pages long but has two appendices, one with the copyrighted source code of the entire system and one with the user's manual.[37]

There is very high variation between the minimum and maximum number of pages. In our set, the shortest patent for which we consider the disclosure level quite low has seven pages[38] and the longest one for which we consider the disclosure level quite high has 438 pages.[39] The average patent has between thirty and forty pages.

37. U.S. Patent No. 5,499,371 (issued Mar. 12, 1996). The patent is assigned to Persistence Software, Inc.

38. U.S. Patent No. 5,550,984 (issued Aug. 27, 1996). The patent is assigned to Matsushita Electric Corporation of America.

39. U.S. Patent No. 4,953,080 (issued Aug. 28, 1990). The patent is assigned to Hewlett-Packard Company.

Figures are often used to illustrate a patent's main elements with schematic diagrams of various kinds and examples of user interfaces such as a menu picture. Two kinds of figures are typically used for disclosure of technical details: block diagrams and flowcharts. Block diagrams are used to represent the principal parts of the system with drawings to show both their basic functions and their functional relationships. Flowcharts are used to represent graphically programs, depicting inputs, outputs, and units of basic activity. In addition to figures, we can find tables that may describe protocol commands, code skeletons (for example, function definitions with input/output arguments), and, sometimes, even code fragments in a programming language.

Twenty-three patents in our set have medium disclosure levels and twenty-one have low disclosure levels. Only six have high disclosure levels and include very detailed code. Thus, the level of disclosure for a PHOSITA to recreate the invention is typically insufficient. The implicit assumption behind a patent is that the PHOSITA has the skills, resources, and tools of the inventor but these are not specified. Considering the many ways we can develop software today, this is clearly inappropriate. Indeed, Burk and Lemley have commented on the low disclosure requirements for software patents (at least, compared with biotechnology patents) noting that "[t]he Federal Circuit has essentially excused software inventions from compliance with the enablement and best mode requirements."[40]

Disclosure practices differ for fine-grain and large-grain patents. Fine-grain patents are likely to describe a particular technique for a specific problem and involve a data structure and associated code. Disclosure typically involves flowcharts, code skeletons, and code fragments in a programming language. For instance, a representative patent[41] in our set describes a method for controlling the execution of an object-oriented program to effect a defined action (e.g., stopping the program) when a specified function is invoked on a specified object during execution of the program thus, a "breakpoint" can be inserted at a previously determined place in the program. The invention is useful for a debugger, a tool that aids software development by giving a user control over and access to a running program. The patent's disclosure level is high. The patent is twenty-five pages long and has twelve

40. Burk & Lemley, *supra* note 31, at 1156.

41. U.S. Patent No. 5,093,914 (issued Mar. 3, 1992). The patent is assigned to AT&T Bell Laboratories.

figures, of which eight are block diagrams and one a flowchart with code skeleton. In addition, it has eleven tables that illustrate program execution control with C++ code fragments.

Large-grain patents are broader and can describe an entire system. They primarily involve interfaces and possibly data structures and code, but perhaps not at a very deep level of detail. Thus, disclosure typically involves user interfaces, block diagrams, protocol commands, sometimes high-level flowcharts but no code fragments. For instance, the longest patent discussed above that describes an object file system, Hewlett-Packard's 080 patent, has eighty-three figures with five block diagrams, fifty-nine screen displays describing an entire session, and nineteen data structures.[42] Thus, the disclosure level is high.

Disclosure practices also differ for system and application software patents, independent of whether they are fine-grain or large-grain. System patents tend to give more technical and implementation details through flowcharts and code fragments, while application patents tend to focus more on interfaces and thus have less disclosure. In our set, out of eleven system patents, three have high disclosure level; four have medium; and four have low. Out of thirty-nine application patents, three have high disclosure levels; nineteen have medium; and seventeen have low.

Over time, software has become better and broader addressing an ever-increasing number of user requirements and thus has become more complex and sophisticated. To deal with increasing complexity, software-programming languages have become higher-level, hiding low-level details to programmers and fostering software design rationalization through decomposition into manageable pieces. Software engineering has evolved towards higher levels of abstraction from simple routines and procedures to more organized units such as modules, packages, object classes, components, and services. Thus, we suspect that the level of disclosure has not really changed much over time but the way software is disclosed has changed. From our set, where most patents are over a short period (between 1985 and 1995), it is hard to make any definite observation. However, the more recent patents on GUI typically rely on object-oriented technology and use object classes for disclosure. Because object classes have well-defined interfaces and can be organized in a hierarchical fashion, they provide a good means to disclose both design and implementation aspects of a complex system.

42. *See supra* note 39.

Although there seems no compelling reason that the level of disclosure for software patents should be very much higher than other classes of invention, there are two practical reasons why reverse engineering should not be necessary to replicate an invention.[43] First, patent disclosure does not require access to an artifact to discover its secrets. Second, software licenses typically forbid the use of reverse engineering in any form. Reverse engineering enables one to understand the implementation, not the design, of the software, which is better conveyed through higher-level means. Furthermore, reverse engineering is getting less and less useful as development relies on sophisticated tools that produce much code automatically and different tools will produce different code, for the same specification. Thus reverse engineering may be useful only for fine-grain patents with low disclosure level. In our set, only one fine-grain patent[44] could be successfully reverse engineered.

V. HOW "REAL" ARE SOFTWARE PATENTS?

Part of the reason for the antagonism toward software patents is a suspicion that they are not granted for real inventions, but are simply a strategic tactic by big firms to lock-out competitors. Such suspicions have been partly whipped up by popular books such as Seth Shulman's *Owning the Future*.[45] The tone of Shulman's work is set at the outset:

> We find university researchers thrown in jail for "stealing" their own ideas; software firms holding the entire industry to ransom over basic widely used programming techniques; and life saving cancer treatments kept from dying patients by legal wrangles over the underling technology.[46]

Another seam of literature encourages firms to exploit their patent "mines," not just for products but also to generate a royalty stream. The tenor of this discussion is exemplified in Kevin G. Rivette and David Kline's *Rembrandts in the Attic*:[47]

---

43. Here, reverse engineering would mean the acquisition of an example of the patented software and discovering its secrets by experimentation, for example, by observing its input-output behavior or by "decompiling" the binary code to reveal the source code.

44. U.S. Patent No. 4,821,220 (issued Apr. 11, 1989). The patent is assigned to Tektronix, Inc.

45. SETH SHULMAN, OWNING THE FUTURE (1999).

46. *Id.* front cover flap.

47. KEVIN G. RIVETTE & DAVID KLINE, REMBRANDTS IN THE ATTIC: UNLOCKING THE HIDDEN VALUE OF PATENTS (2000).

> [This work] provide[s] the first practical and strategic guide that shows CEOs and other managers how to unlock the enormous financial and competitive power in their patent portfolios. . . . The competitive battles once fought for control of markets and raw materials are today increasingly being waged over the exclusive rights to new ideas and innovations.[48]

Intellectual property and strategic patenting are also hot topics in the more restrained professional and scholarly literature.[49] Internet and software patents have been among the most prominent targets of the criticism that firms are running "patent mills" rather than bona fide research and development (R&D) operations.[50]

The aim of this section of our paper is to explore whether or not the software patents in our sample are "real" patents in the sense that they protect significant innovations solving real problems. We seek to do this, first, by giving an objective and quantitative measure of whether software patents compare favorably with patents in general and, second, by establishing qualitatively whether software patents reflect the real research agendas of the software community.

In 2000, John R. Allison and Mark A. Lemley established a useful set of quantitative measures for measuring the novelty and depth of patents.[51] Their two principal measures were the number of prior art references and the number of claims in a patent. Their paper considered patents in some fourteen major areas of technology: pharmaceutical, medical devices, biotechnology, computer-related, software, semiconductor, electronics, chemistry, mechanics, acoustics, optics, automotive-related, energy-related, and communications-related. A random sample of approximately one thousand patents issued during the two-year period from June 1996 to May 1998 was examined and each assigned to the related technology area. Of the sample, seventy-six

48. *Id.* front cover flap.

49. *See, e.g.*, H. JACKSON KNIGHT, PATENT STRATEGY FOR RESEARCHERS AND RESEARCH MANAGERS (2d ed. 1996) (focusing on the strategy and reasons for obtaining a patent); HOWARD B. ROCKMAN, INTELLECTUAL PROPERTY LAW FOR ENGINEERS AND SCIENTISTS (2004) (explaining the necessity to engineers and scientists that they retain the services of intellectual property attorneys).

50. John R. Allison & Emerson H. Tiller, *Internet Business Method Patents*, *in* PATENTS IN THE KNOWLEDGE-BASED ECONOMY 259, 259 (Wesley M. Cohen & Stephen A. Merrill eds., 2003), *available at* http://www.nap.edu/books/0309086361/html/ (last visited Apr. 3, 2005).

51. John R. Allison & Mark A. Lemley, *Who's Patenting What? An Empirical Exploration of Patent Prosecution*, 53 VAND. L. REV. 2099, 2174 (2000).

patents were for software inventions.[52] Extrapolating, the authors estimated that of the 230,000 U.S. patents granted during their period of interest, some 18,000 were for software inventions.

For each of the fourteen areas of technology, the number of prior art references was determined, with separate counts for U.S. patent citations, foreign patent citations, and non-patent citations. The results for software, with a comparison of the average for all the patents, are shown in the first two rows of Table 2 in Appendix A. Not surprisingly, given the recent development of software patents, the number of patent citations was somewhat less than average. The lack of patent citations was partly compensated for by an above average number of non-patent citations. The authors noted:

> Interestingly, despite vocal criticism from some quarters, the software industry actually cited relatively more non-patent prior art than most other areas of technology. While we think this is an encouraging sign, it does not necessarily mean that the PTO is doing a good job of finding the relevant prior art in the software field. Many commentators have suggested that virtually all the relevant art in the software industry is non-patent art. If so, the fact that most prior art cited in software patents still consists of other patents may mean that the PTO isn't citing nearly *enough* non-patent prior art in the field.[53]

Thus, these authors leave us in no doubt that given the newness of the field, software patents have as much technical depth as patents in other technological areas. Allison and Tiller made a similar analysis of Internet-related software patents, shown in row 3 of Table 2. This suggests that for more recent software patents the number of citations is increasing and now exceeds the average.

In the last row of Table 2, we have computed the citation counts for the fifty most cited patents. As might be expected, since these are probably among the leading software patents, the average number of citations in them is greater than for the average software patent, but not wildly so.[54] We conclude that the prior art cited in the fifty most cited software patents is at least as good as patents in general and as good as the general run of software patents.

52. There were 242 computer-related patents, including 76 for software. *Id.* at 2115.

53. *Id.* at 2131.

54. We found that the nature of non-patent art varied depending on the patent area. Narrow patents tended to refer more to the academic literature because there is a well-identified research area (e.g., GUI, database), whereas broad patents tend to refer more to trade press articles and software vendor papers.

Allison and Lemley also undertook an analysis of the number of claims in patents in the different technological areas.[55] As shown in the first two rows of Table 3 in Appendix A, the number of claims for software patents is somewhat above the average. The corresponding claim count for the fifty most cited software patents is shown in the third row of Table 3. Assuming that the number of claims is not unrelated to the degree of inventiveness, this data suggests that software patents are more inventive than average patents, and the fifty most cited software patents are more inventive still.

We next wish to establish whether software patents reflect the research agendas of the software community. We suggest this can be done qualitatively by comparing the subject matter of the patents with the research agendas of the academic-industry research community at the time that a patent was applied for/issued.

The highest numbers of patents in our set are in GUI (ten patents) and database (eight), two areas that correspond to a very active research agenda since the 1980s.

This research agenda was recognized early by the two major computer science associations, the Association of Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). The ACM created special interest groups (SIGs) in these areas: SIGCHI (Computer Human Interaction) and SIGMOD (Management of Data) with conferences and associated journals, *ACM Transactions on Computer-Human Interaction* and *ACM Transactions on Database Systems*. Similarly, the IEEE held conferences on these topics and created journals, such as *IEEE Transactions on Visualization and Computer Graphics* and *IEEE Transactions on Knowledge and Data Engineering*. This research agenda also led to important new products and players such as Xerox and Apple in GUI, and IBM, Oracle and others in databases.[56] All the other patents involve areas which have been the subject of intense research (CAD, CASE, information retrieval, knowledge base, multimedia, etc.) for which there are well-identified research communities, products, and companies.

We conclude that there is a strong relationship between the academic and industrial research agendas. The former is reflected in the subject matter of conference themes and research publications, while

55. *See* Allison & Lemley, *supra* note 51, at 2149, 2158-60.

56. For the commercial history of DBMS and GUIs, see MARTIN CAMPBELL-KELLY, FROM AIRLINE RESERVATIONS TO SONIC THE HEDGEHOG: A HISTORY OF THE SOFTWARE INDUSTRY 145-49, 247-51 (2003).

the latter is evidenced in patents, company formations, and new products. The relationship between the two is so obvious we would describe it as "academic-industry lock-step."

VI. ARE SOFTWARE PATENTS TOO BROAD?

A "broad" patent is one which covers a wide area of the inventive space. Both wide patent scope and long duration are desirable attributes from an inventor's viewpoint. Indeed, a strand of the economic literature suggests that "greater scope [is] roughly similar to greater duration in terms of its incentive effect on initial invention."[57] However, with almost no exceptions, the twenty-year duration of a patent is statutorily determined and is non-negotiable.[58] In contrast, the breadth of a patent is somewhat elastic and is determined at the outset by a patent examiner and by the courts if the patent is contested.[59]

There is a significant economic literature on patent scope, much of which predates the patentability of software. There are two principal and contrary viewpoints: one that broad patents are better, the other that narrow patents are better.

The most influential "broad is better" argument is related to the "the prospect theory" of patents, which asserts that a new invention space is akin to a mining prospect.[60] Broad protection has the socially desirable effect of excluding competitors from mining the prospect. Such exclusion is desirable because it prevents the waste of too many innovators searching for the same invention, releasing them to make non-competing inventions or to seek other opportunities. On the other hand, the advocates of "narrow is better" argue that narrow patents "avoid stifling progress"[61] by encouraging competitors to develop closely related inventions so that the technology develops more rapidly than if just a single firm or a small number of firms participated.

The broad-versus-narrow debate has to be qualified by the type of invention that is being considered. Two styles of invention are relevant

57. Robert P. Merges & Richard R. Nelson, *On the Complex Economics of Patent Scope*, 90 COLUM. L. REV. 839, 869 (1990).

58. One exception is the extension of patent terms on pharmaceuticals to compensate for regulatory lag. *Id.* at 839.

59. The duration of patents is discussed *supra* Part VII.

60. Edmund W. Kitch, *The Nature and Function of the Patent System*, 20 J.L. & ECON. 265 (1977).

61. Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CAL. L. REV. 3, 16 (2001).

in the software debate: discrete computer-related inventions and pure software inventions.

A discrete invention is a well-defined invention that "does not define any broad prospect."[62] Classic examples are King Gillette's safety razor and Lazlo and Georg Biro's ball-point pen. In computer-related inventions, a good example would be Texas Instruments' "Speak & Spell."[63] Although the innovator possessing a broad patent may "profit handsomely" from it, "possession by that firm of a proprietary lock on the invention is not a serious hindrance to inventive work by many other firms."[64] In short, the progress of civilization is not going to be threatened by a temporary monopoly in safety razors, ball-point pens, or talking toys, whereas the possession of a patent may encourage the inventor to invest in production with the assurance that it cannot be rapidly imitated.

Pure software inventions are often characterized as a "cumulative" technology, proceeding by sequential innovation whereby each generation of software builds on the innovations of previous generations. With pure software inventions, therefore, society would be disadvantaged if a broad software patent could hold up a subsequent software innovation because it required a unique, patented technology. A complex piece of software, it is argued, might need to make use of hundreds of patents and the owner of a single patent could then hold up all progress.[65] For this reason, it is argued that patents should be defined narrowly so that it is feasible for inventors to generate substitutes. A good example of a software patent that was defined relatively narrowly was the MP3 music compression algorithm. Although the MP3 patent[66] has proved profitable for its owners, it is sufficiently narrow that there are several substitutes—some patented some not—such as Microsoft's WMA, Real Networks RAM, Apple Computer QIF, and the open-source OGG format. A whole industry from Napster to iTunes is currently developing around such algorithms. In our experience, few software inventions are unique in the sense that adequate substitutes cannot be devised.

62. Merges & Nelson, *supra* note 57, at 880.

63. U.S. Patent No. 4,516,260 (issued Aug. 29, 1980). Incidentally, this patent has exemplary disclosure and includes a fifty-page program listing.

64. Merges & Nelson, *supra* note 57, at 881.

65. Bryan Pfaffenberger, *The Coming Software Patent Crisis: Can Linux Survive?*, LINUX JOURNAL (Aug. 10, 1999), *at* http://www.linuxjournal.com/article.php?sid=5079.

66. U.S. Patent No. 5,701,346 (issued Dec. 23, 1997).

Another complication of pure software inventions is the impact of network effects, by which consumers benefit from the adoption of a common technology or standard.[67] In personal computing—where network effects are most apparent—investments in learning and the ability to share files have led to dominant applications that have maintained an eighty-plus percent market share, largely independent of the technical merits of the software.[68] Of course, although network effects can lock users into an inferior technology for a significant period of time, there is substantial evidence that superior products do ultimately win out.[69] But network effects undoubtedly protect incumbents from non-innovative, "me too" competitors.

Because of the overwhelming power of network effects in software, it can be argued that they are a sufficient reward in themselves, and patents simply confer another gratuitous layer of protection. Of course, this may be true for market leaders such as IBM and Microsoft, but not for innovators in general. Indeed, the existence of a patent may be crucial for a small innovative firm in preventing appropriation of its technology by a dominant firm. For example, a major innovative area in the mid-1990s involved Web-authoring tools, for which Vermeer Technologies applied for a set of patents before the concept diffused very widely.[70] This gave the firm protection against major players such as Microsoft and Netscape Communications; in the end, Microsoft acquired Vermeer and its FrontPage software became a Microsoft product.

In this instance we do not know whether Microsoft purchased Vermeer for its patents or for the lead-time of an existing product—probably both. In fact, Vermeer's patents were not so broadly drawn that it was impossible for others to develop Webpage-authoring tools and it became a major software category.[71]

Our dataset of the fifty most cited patents enables us to make some general observations about the appropriateness of the breadth of

67. Richard N. Langlois, *External Economies and Economic Progress: The Case of the Microcomputer Industry*, 66 BUS. HIST. REV. 1 (1992).

68. For a software-focused discussion of network effects, see STANLEY J. LIEBOWITZ & STEPHEN E. MARGOLIS, WINNERS, LOSERS AND MICROSOFT: COMPETITION AND ANTITRUST IN HIGH TECHNOLOGY (1999).

69. Liebowitz and Margolis have studied this phenomenon in several software categories, including word processors, spreadsheets, and web browsers. *Id.* at 163-200, 217-23.

70. CHARLES H. FERGUSON, HIGH ST@KES, NO PRISONERS: A WINNER'S TALE OF GREED AND GLORY IN THE INTERNET WARS 102-35 (1999).

71. Mainstream products competing with FrontPage included HomePage by Claris, DreamWeaver by Macromedia, HotMetal by XMetaL.

software patents. Are patents generally construed narrowly, or are they so broad that they impede progress?

From a legal standpoint, the breadth of a patent is determined by the scope of its claims, which is likely to be broader than the scope of enablement. In the last analysis, the scope of a patent can be determined only by a court in the event that the patent is infringed or challenged. As technical experts, we can however, evaluate the technical breadth of patents, that is their scope of enablement. This has been done in Table 6 in Appendix A, where we classify patents being of broad, medium, or narrow technical breadth. Among the fifty patents of study, eighteen are broad, six medium and twenty-six narrow.

We find that there is no simple correspondence between the technical breadth of a patent and the number of claims. Some narrow patents have many claims, while some broad patents have few claims. For example, H-Renee's 4,992,940 patent has only twelve claims but is technically broad, while a patent assigned to ActaMed Corporation has eighty-five detailed claims and its technical breadth is narrow.[72]

We found a few patents in our set that were so broad that they could potentially foreclose the inventive space for competitors. This is the case, for instance, of two patents from Teknowledge. One patent has seventy claims,[73] the other sixty-nine,[74] and both cover broadly the space of knowledge base using inference rules. However, we think there would still be a way to invent around the patents, for instance, by specializing the knowledge base for specific applications. Indeed, there is actually a patent from Westinghouse[75] in our set which applies a knowledge base for monitoring and diagnosing sensor and interactive based process systems. The only case where we believe it would be difficult to invent around is when the patent is both broad and obvious, as the H-Renee's patent.

Although at the micro-level of individual patents there is no simple correspondence between technical breadth and the number of claims, the sample of fifty patents taken as a whole shows a strong correlation.[76] Patents we classify as broad have, on average, nearly forty claims, medium breadth patents about twenty-five claims, and narrow patents about twenty-two claims. We also note that the number of prior art

72. U.S. Patent No. 5,560,005 (issued Sept. 24, 1996).
73. U.S. Patent No. 4,591,983 (issued May 27, 1986)
74. U.S. Patent No. 4,658,370 (issued Apr. 14, 1987)
75. U.S. Patent No. 4,649,515 (issued Mar. 10, 1987).
76. *See infra* Table 4, app. A.

references is strongly correlated to technical breadth: broad patents have an average of about twenty-seven references, medium breadth eighteen references, and narrow about thirteen. Thus, our evaluation of technical breadth conforms to the positive correlations of scope with claims and prior art references observed elsewhere in the economic literature.[77] Interestingly, the number of citations from subsequent patents is negatively correlated to technical breadth. It has been noted that the value of a patent is positively correlated to the number of citations, rather like the number of citations of a scientific paper is a measure of its value.[78] This would suggest that software techniques described by technically narrow patents are of greater utility than broad software inventions.

VII. DO SOFTWARE PATENTS LAST TOO LONG?

The effective life of a software patent is generally believed to be short. The term "effective life" has been defined as "the expected time until a patented product is replaced in the market."[79] Even though the actual life of a U.S. patent is twenty years from the date of filing, this is immaterial for inventions when the marketable life of the invention is considerably less. Patent protection ceases when either its twenty-year life is expired or a new innovation has rendered it "irrelevant."[80]

It is frequently asserted that software patents last too long.[81] Indeed, Pamela Samuelson has commented that the pace of innovation in software is so fast that by the time a patent has been issued the useful life of an invention may be over.[82] Whether one agrees or not, software patents are far from unique in this criticism. Several authors have asserted that the effective life of patents in many technologies is far less

77. *See, e.g.*, Dietmar Harhoff et al., *Citations, Family Size, Opposition and the Value of Patent Rights*, 32 RES. POL'Y 1343–63 (2003);.Jean O. Lanjouw & Mark Schankerman, *The Quality of Ideas: Measuring Innovation with Multiple Indicators*, (NBER Working Paper 7345, 1999); Joshua Lerner, *The Importance of Patent Scope: An Empirical Analysis*, 25 RAND J. ECON. 319-33 (1994).

78. Harhoff, *supra* note 77, at 1350.

79. Ted O'Donoghue et al., *Patent Breadth, Patent Life, and the Pace of Technological Progress*, 7 J. ECON. & MGMT. STRATEGY 1, 2 (1998).

80. *Id.*

81. Cohen & Lemley, *supra* note 61, at 46; *see* Jason V. Morgan, *Chaining Open Source Software: The Case against Software Patents*, League for Programming Freedom, *at* http://lpf.ai.mit.edu/Patents/chaining-oss.html (last modified Jan. 15, 2002).

82. Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, n.134 (1994).

than twenty years and may be much closer to four or five years.[83] Nor does everyone agree that excess patent duration actually matters. For example, William Nordhaus observed as long ago as 1972 that, despite the short effective life of patents, "there does not seem to be a strong case for making changes."[84]

As noted above in Part VI, it has been argued that broad patents of short duration or narrow patents of long duration are both effective in protecting innovations. A broad patent of short duration, less than two years, would foreclose entry to a large product space enabling an innovating firm to gain a first-mover advantage; when the patent had expired, the firm would then be subject to full competition. In the case of the software industry, this would not, in practice, be very different from the way the industry currently operates, where first-mover advantage is sometimes argued to be sufficient and patent protection is therefore unnecessary.[85] Where inventions are not patented, however, developers typically work in trade secrecy in order to be first to market, and imitators have to use reverse engineering to discover the novel technologies. Where a firm takes out a patent, the economic loss from both trade secrecy and reverse engineering are reduced. By contrast, a narrow patent of long duration would confer longer-term advantage to the innovation, but because the product space was narrow, it would be much easier for competitors to create non-infringing substitutes.

Since patent length has only recently been changed from seventeen years after issuance to twenty years after filing, it is unlikely that there will be another change to statutory patent length in the near future. There is even less reason to suppose that a special provision will be made for software inventions. There are therefore two issues we wanted to address in this paper: the effective life of software patents and the harm, if any, caused by a twenty-year software patent.

First, what is the effective life of the patents in our dataset? Does the average length confirm empirically that the effective life of parents is much less than twenty years? Detailed examination shows that the patents' effective life is between three to ten years after issuance, with an average of five. The effective life of patents depends on the life of the technologies on which they bear, because system-software patents

83. O'Donoghue, *supra* note 79, at 2.

84. William D. Nordhaus, *The Optimum Life of a Patent: A Reply*, 62 AM. ECON. REV. 428, 428 (1972).

85. *See, e.g.*, Campaign Against Software Patents, *The Danger of Software Patents to Europe*, 2002, *at* http://patents.caliu.info/explicacio.en.html (Matt Bonner trans.) (last modified May 2, 2002).

typically last longer than application patents whose technologies change more rapidly. For example, patents on programming language support may well last ten years or more because of the longevity of the programming language. However, patents in object-oriented databases or knowledge bases did not last long because the technologies almost disappeared after a ten-year period.

Second, given that patent length is fixed, is any actual harm done by the twenty-year duration of software patents? Do patents block progress in the software field because a superior invention would infringe on them. For example, Cohen and Lemley have argued that a patent drawn too broadly might block innovations two or three generations subsequent to the original innovation.[86]

We find that patents with a wide technical breadth create a potential for the kind of bottleneck Cohen and Lemley discuss, but such patents are not insurmountable obstacles, and time tends to reduce the potential. For example, the two patents in the dataset from Teknowledge are broad and have the capability of holding up subsequent innovations in knowledge base management. Although these patents have high technical depth, the knowledge system they protect is described using high-level rules and a general inference engine. However, these technically broad patents are no longer a problem, because the knowledge base approach, as originally designed, did not really succeed. A different and superior technical approach emerged despite broad patents. Blocking is a potential problem only if the inventor has chosen the "best" solution path. Another patent with such capacity is Encyclopaedia Britannica's, which has low technical depth and a high number of claims.[87] However, this patent was subsequently re-examined and its breadth narrowed.

The patents with a narrow technical breadth in our set do not have obvious hold-up potential. Depending on how broadly claims are drawn and interpreted, we believe that substitute innovations could be devised for all the patents. For example, AT&T's patent for software controlling the execution of object-oriented programs using breakpoints is quite broadly drawn, and the claims are not specific to a particular programming language or data structure.[88] However, among patents citing to this patent, there are several for controlling program execution that use breakpoints. These patents are non-infringing, or at least have

86. Cohen & Lemley, *supra* note 61, at 48.
87. U.S. Patent No. 5,241,671 (issued Aug. 31, 1993).
88. *See supra* note 51 and accompanying text.

not been challenged by AT&T. Although not all algorithms are equally good, it is possible to devise substitutes. For example, if a developer had not wanted to pay the license fee to use the famous LZW data compression patent, then there were plenty of other algorithms to choose from, although they were not all equally good.[89] The developer was faced with the choice of paying to use LZW or using a less efficient, royalty-free algorithm. This way the patent system rewarded the inventor of LZW, but did not impede the progress of software in general.

VIII. COPYRIGHT, PATENTS, OR BOTH?

There is considerable controversy over which form of IP protection is appropriate for software. Some argue that copyright alone is sufficient,[90] while others argue that patents are needed to protect against rapid imitation.[91] Here we summarize very briefly the principal arguments of the two forms of protection.

The benefits and convenience of copyright protection are superficially appealing to a software author. Copyright protects both the source code and the object code. It is free of cost and lasts for seventy years after the death of the author. However, copyright provides no protection for the functions of software, and a competitor can legally imitate them. Although there have been legal cases that have tested the limits to which imitation is permissible, in general copyright protects only the exposition of the program, not its concepts and ideas. In practice, therefore, the source code of software is only rarely disclosed and the author actually relies on trade secrecy. The need for trade secrecy is recognized by the Copyright Office: if an author wishes to formally register copyright, it is permissible to deposit a "striped" version of the source code listing,[92] or only the first and last twenty-five pages of the listing.[93]

89. U.S. Patent No. 4,558,302 (issued Dec. 10, 1985). The patent was assigned to the Sperry Corp. and is now expired.

90. *See* Pamela Samuelson, Benson *Revisited: The Case against Patent Protection for Algorithms and Other Computer-Program Related Inventions*, 39 EMORY L.J. 1025, 1135-36 (1990); John Swinson, *Copyright or Patent or Both: An Algorithmic Approach to Computer Software Protection*, 5 HARV. J.L. & TECH. 145, 147 (1991).

91. Patrick K. Bobko, *Open-Source Software and the Demise of Copyright*, 27 RUTGERS COMPUTER & TECH. L.J. 51 (2001).

92. That is, a photocopy taken through a diagonal zebra striped mask that makes the source code unusable.

93. COPYRIGHT BASICS, U.S. COPYRIGHT OFFICE, CIRC. NO. 1, at 8 (Sept. 2000), *available at* http://www.copyright.gov/circs/circ01.pdf.

Copyright has distinct benefits and disadvantages for society. Because products can be freely imitated for any new software genre, the market rapidly supplies low cost imitations or "clones" that drive prices down. There are two important disadvantages of this rapid imitation. First, the easy cloning of products does not necessarily result in a rich ecology of competing products, because entrepreneurs do not gain sufficient rewards to invest in R&D. Second, trade secrecy inhibits the diffusion of knowledge. Competing products are inevitably written from scratch and developers are constantly "reinventing the wheel."[94]

For a developer, the most important benefit of a patent is that it protects the functions of software. The main disadvantage of a patent for a developer, however, is that in disclosing the technology, it becomes more vulnerable to improvement—and therefore obsolescence—in the marketplace than if held as a trade secret. Additionally, patents are relatively expensive in terms of costs for patent attorney, application, and renewal fees.

Again, software patents have distinct benefits and costs for society. The main benefits are that technical knowledge is diffused instead of remaining a trade secret, and after expiration of the patent, this knowledge can be freely used. The cost to society is, of course, that the monopoly enjoyed by an inventor may cause prices to be higher than otherwise would be the case.

The fact that software is currently protected by patents *and* copyright seems, on the face of it, to be having the best of both worlds. Here, we argue by examining actual patents in our dataset that both forms of protection are needed and that they serve to protect different aspects of software. We will consider both large-grain and fine-grain patents, since their cases are somewhat different. Large-grain software patents often protect a whole product while fine-grain patents protect a single component or part of a product.

A very clear example of a large-grain software artifact needing both forms of protection is the CD-ROM encyclopedia of the kind exemplified by Encyclopaedia Britannica's patent in our dataset.[95] An electronic encyclopedia contains two kinds of intellectual investments. First, there is the software whose functions and code create the on-line experience for the user and provide different pathways to access the information inside the encyclopedia. In the rapidly evolving technology

94. Mark A. Haynes, Commentary, *Black Holes of Innovation in the Software Arts*, 14 BERKELEY TECH. L.J. 567, 569 (1999).

95. U.S. Patent No. 5,241,671, *supra* note 87.

of online encyclopedias, the twenty-year duration of the patent seems perfectly adequate. The second form of protection is needed for the content—principally the text of the encyclopedia, which has been created not by software writers but by scholars commissioned by the encyclopedia's editors. In almost all cases, the text for CD-ROM encyclopedias has been derived from a hardcopy version.[96] The life cycle of an encyclopedia is traditionally very long; for example, the eleventh edition of the Encyclopaedia Britannica, published in 1911, was not replaced entirely until the fifteenth edition was produced in 1974, at a cost of thirty-two million dollars—which is the basis of today's edition. Of course, the garden of knowledge is constantly pruned and renewed, but it is only very occasionally that the entire work is revised. In this context, the statutory duration of copyright seems proportionate.

The Encyclopaedia Britannica patent is therefore protecting a substantial software investment made when repurposing its copyright-protected, text-based encyclopedia into electronic form. The degree of novelty is comparable with the reorganization of the encyclopedia that took place for the 1974 edition, when the concepts of "macropedia" and "micropedia" were introduced. For example, in the CD-ROM version, information could be accessed by novel means such as timelines, graphical point-and-click menus, or a "researcher's assistant," which were not possible with the bound volumes.

These innovations were made in the context of an earlier market entry, the Grolier CD-ROM Encyclopedia, which was almost entirely text based. Clearly, Encyclopaedia Britannica gained from patent protection since its innovations could not be copied, but what did society gain from the patent? Principally, it gained from the disclosure of knowledge.

The patent contains some twenty pages of architecture and flow diagrams. Although the disclosure level is quite low, and the patent can be criticized on these grounds, it is of a level similar to that which could be gained from conventional, and costly, reverse engineering. It would serve as an effective primer for any one entering the CD-ROM encyclopedia business, or for a competitor seeking to improve its product.

Incidentally, the subsequent history of this patent is interesting. It was re-examined shortly after being issued, and several of its forty-one

96. For histories of CD-ROM encyclopedias, see RANDALL E., STROSS, THE MICROSOFT WAY: THE REAL STORY OF HOW THE COMPANY OUTSMARTS ITS COMPETITION 78-93 (1996); CAMPBELL-KELLY, *supra* note 56, at 288-94.

claims were cancelled. These included such ideas as timelines and graphical point-and-click menus, which were found in a more thorough search of the prior art. This no doubt reflects partly the poor quality of prior art searching when the patent was applied for in 1989; it has been much improved since.[97] However, it is worth noting that patentees cannot claim large product spaces in important markets without challenge. In fact, the mid 1990s turned out to be the heyday of CD-ROM encyclopedias with about six major players.

The CD-ROM encyclopedia was in the vanguard of a convergence of code and content in software products that is still in progress. Other examples include computer games, financial websites, and tax preparation systems.[98] In fact, large-grain software artifacts, especially end-user products, typically contain elements that require copyright protection—such as help files, dictionaries, tutorial materials, and user interfaces.

Where disclosure in a software patent is particularly full, more knowledge may be revealed than is covered by the claims. Several patents in our set include a statement of the form:

> A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.[99]

If the entire source code was appended to the patent—as some advocate—then all the programming work not embodied in the invention could be freely appropriated.[100] For this reason, the additional protection of copyright is necessary or developers would tend to become as economical as possible in their disclosure.

The situation for fine-grain inventions is different. These do not typically protect a product, but rather a single component or part of a product. In our opinion, the essence of these patents is the code; it should be disclosed so that inventors can improve on the invention and make use of it when the patent has expired—in both cases without

97. *See* Examination Guidelines for Computer-Related Inventions, Final Version, 61 C.F.R. § 7478 (1996).

98. For a discussion on "code vs. content," see CAMPBELL-KELLY, *supra* note 56, at 288-89.

99. *See, e.g.*, U.S. Patent No. 5,119,475(issued June 2, 1992); U.S. Patent No. 5,151,987 (issued Sept. 29, 1992).

100. Burke, *supra* note 36, at 1160.

undue experimentation. Not making adequate disclosure is to want the best of both worlds: patent protection with trade secrecy. Worse, to protect code by copyright—so that use cannot be made of it when the patent has expired—seems to go against the most fundamental patent bargain.

In our sample, there are good and bad examples. A patent with exemplary disclosure is an AT&T debugging technique.[101] There are eleven pages of drawings showing detailed data structures, a highly detailed functional description, and code fragments, with no copyright assertions. On the other hand, an IBM patent, whose disclosure in any case is only moderate, contains a copyright notice at the beginning of the specification, and a code fragment has a copyright notice appended. Therefore, although the code fragment serves the purpose of disclosure, it could not be freely used once the patent has expired.[102]

IX. CONCLUSION

There is much discussion in legal and economic literature about the one-size-fits-all nature of the patent system.[103] Yet, surprisingly, the patent system does manage to apply a single set of conventions to inventions as disparate as zip-fasteners, Viagra, and radio sets. Software is better seen not as a *sui generis* phenomenon, but simply as a newcomer to the heterogeneous universe of patentable inventions. Although special IP regimes have very occasionally been devised for particular technologies,[104] it is generally accepted that the legislature cannot create a new IP environment for each new technology that happens to come along. However, there are "policy levers" by which the legislature, courts, and the USPTO can shape the norms of obviousness, disclosure, and breadth in software patents.[105] It is in that spirit that we make our conclusions.

101. U.S. Patent No. 5,093,914 (issued Mar. 3, 1992).

102. U.S. Patent No. 5,151,987 (issued Sept. 29, 1992).

103. R. Polk Wagner, *(Mostly) against Exceptionalism*, *in* PERSPECTIVES ON PROPERTIES OF THE HUMAN GENOME PROJECT 367-82 (Scott Kieff ed., 2003); JAMES BESSEN & ERIC MASKIN, SEQUENTIAL INNOVATION, PATENTS AND IMITATION, 7-8 (MIT Depart. of Econ. Working Paper No. 00-01, 2000).

104. For example, semiconductors were the subject of the Semiconductor Chip Protection Act 1984, 17 U.S.C. §§ 901-14 (2000).

105. *See* Dan L. Burk & Mark A. Lemley, *Policy Levers in Patent Law*, 89 VA. L. REV. 1575 (2003), *available at* http://repositories.cdlib.org/cgi/viewcontent.cgi?article=1089 &context=blewp.

First, we did not find that software patents were, generally, too obvious. In our set of fifty, only two could be described as obvious, and even those were not particularly obvious by the standard of patents in general. We found that the quality of patents was somewhat like the quality of computer-science research papers—certainly some were better than others were, but there was reasonably high standard of acceptance.

We found only six of our fifty patents had completely adequate disclosure. The remaining forty-four had medium or low disclosure that would make reproducing the invention either time-consuming or problematic. Even though disclosure is not low by the standard of mechanical inventions, we think that there are grounds for improvement, since it is relatively easy to improve the quality of disclosure by repurposing specifications and code used in the development process.

We found that software patents are "real" in the sense that they are not merely strategic devices intended to block the entry of competitors. They have genuine technical depth as measured by the number of prior art citations and claims. We found substantial evidence that the subject matter of patents was in line with the contemporary research agendas of the computer science community.

We found the assertion that software patents are over-broad difficult to refute or substantiate. There appeared to be no general correlation between the number of claims and patent breadth. However, empirical inspection of this group of patents showed that their potential for blocking or hold-up was minimal. Moreover, in one case an over-broad patent had been re-examined to narrow its claims.

We found that the effective life of patents in the dataset to be much shorter than the statutory twenty-year duration, perhaps as little as five years on average. Because the technologies so quickly were made obsolete, we did not see that this caused a significant problem. In any case, narrowing of patent breadth is an effective remedy for over-long patent duration.

We found that both copyright and software patents are needed to adequately protect large-grain software artifacts, particularly when they contain both code and literary content. In the case of fine-grain software inventions, we felt that there was a risk that copyrighting code could extend patent life by protecting the best mode of recreation.

APPENDIX A

Table 1 classifies the fifty most-cited software patents by technology. To facilitate deeper analysis, Table 5 lists the fifty patents ordered by class, technology and decreasing number of cites, and includes the patent title. To assess further the technical depth, disclosure level, claims, technical breadth, and prior art of software patents, Table 6 describes the fifty patents using several technical indicators:

"#Pages" gives the number of pages that describes the patent. A high number of pages suggest a detailed description.

"#Figs" gives the total number of figures of the patent. Figures are used to illustrate the patent's main elements with schematic diagrams of various kinds and examples of user interfaces, such as a menu picture. Two kinds of figures are typically used for disclosure of technical details: block diagrams (to represent graphically the principal parts of the system with their functional relationships) and flowcharts (to represent graphically programs). In addition to these, we can find tables that may describe protocol commands, code skeletons, and sometimes even code fragments in a programming language.

"Tech. depth" presents the technical depth of a patent as low (L), medium (M) or, high (H). Low technical depth usually means very high-level description that avoids important technical details. High technical depth means emphasis on the important technical details. Most patents in our sample set have high technical depth, which probably explains their high number of citations. In particular, almost all fine-grain/system software patents have high technical depth.

"Discl. level" presents the level of disclosure of each patent. We define this as the ease for a skilled practitioner to convert a patent into software programs. Obviously, there is a strong correlation between technical depth and disclosure level. In particular, high disclosure level requires high technical depth. On the other hand, low technical depth and low disclosure level mean that the patent is overly general.

"Tech. breadth" presents the technical breadth of each patent in our set as broad (B), medium (M), or narrow (N).

Table 1. Software patents classification by technology

	Fine-grain (one or a few components)	**Large-grain (several or many components)**
Application Software (specific) *39 patents*	**GUI** • Jeffrey S. Risberg (ind.) **Video application (Video)** • Robert M. Best (ind.) *2 patents*	**Banking/commerce (BC)** • IBM, Online Resources, Strategic Processing, Citibank. **Computer Aided Design (CAD)** • International Chip Corp. **Computer Aided Software Engineering (CASE)** • Xerox, Apollo, Minicom Data. **Communication (Com)** • Prodigy Services, IBM, National Semiconductor. **Database (DB)** • Eastman Kodak, HP, Tektronix, Bull, H-Renee, Persistence Software, ActaMed, Texas Instruments. **Graphical User Interface (GUI)** • Xerox (1), David H. Judson (ind.), AT&T, U. Pittsburgh, Schlumberger, Cadware, Tektronix, Xerox (2), Taligent, Sun. **Information Retrieval (IR)** • Sun. **Knowledge Base (KB)** • Teknowledge (1), Teknowledge (2), Westinghouse Electric. **Multimedia (MM)** • Encyclopaedia Britannica, Sony. **Security (Sec)** • International Security Note and Computer, Matsushita. *37 patents*
System Software (generic) *11 patents*	**Programming Language support (PL)** • AT&T, Tektronix, IBM, 501 Philon. **Task scheduler (TS)** • Tektronix. *5 patents*	**File system (FS)** • HP, R.J. Reynolds Tobacco. **Middleware (MW)** • Software AG, Tolerant Systems, Bull. **Real-time system (RTS)** • Hazox. *6 patents*

Table 2. Prior art

	Sample Size	Reference type	Mean	Median	Min	Max
All patents	995	U.S. Patent	10.34	7	0	137
		Foreign Patent	2.44	1	0	43
		Non-patent	2.37	0	0	68
		Total	15.16	10	0	163
Software patents	76	U.S. Patent	9.59	7	1	112
		Foreign Patent	1.26	0	0	12
		Non-patent	3.54	1	0	36
		Total	14.39	10	1	137
Internet software technology	330	Patent	12.03	n/a	0	353
		Non-patent	4.83	n/a	0	169
		Total	16.86	n/a	1	376
Fifty most cited software patents	50	U.S. Patent	10.80	8	1	38
		Foreign Patent	0.52	0	0	8
		Non-patent	6.88	3	0	45
		Total	18.20	12	3	84

Source: John R. Allison & Mark A. Lemley, *Who's Patenting What? An Empirical Exploration of Patent Prosecution*, 53 VAND. L. REV. 2099, 2149, 2158-60 (2000).

Table 3. Claims

	Sample Size	Mean	Median	Min	Max
All patents	995	14.87	12	1	120
Software patents	76	17.11	11	1	120
Fifty most cited software patents	50	28.66	21	6	85

Table 4. Technical breadth vs. claims, prior art references, and forward citations

Technical breadth	**Broad**	**Medium**	**Narrow**
Average number of claims	39.56	25.33	21.88
Average number of prior art references	26.78	18.17	12.65
Average number of forward citations	109.44	125.67	134.65

Table 5 describes our set of fifty patents, ordered by class (FGAS: fine-grain application software, FGSS: fine-grain system software, LGAS: large-grain application software, LGSS: large-grain system software), technology and decreasing number of cites.

Table 5. Patent descriptions

Patent Number	**Company**	**Year**	**Class**	**Tech.**	**No. of Cites**	**Title**
5339392	Independent inventor	1994	FGAS	GUI	109	Apparatus and method for creation of a user definable video displayed document showing changes in real time data
4305131	Independent inventor	1981	FGAS	Video	174	Dialog between TV movies and human viewers
5093914	AT&T	1992	FGSS	PL	175	Method of controlling the execution of object-oriented programs
4821220	Tektronix	1989	FGSS	PL	150	System for animating program operation and displaying time-based
5151987	IBM	1992	FGSS	PL	123	Recovery objects in an object oriented computing environment
4667290	501 Philon	1987	FGSS	PL	119	Compilers using a universal intermediate language
5136705	Tektronix	1992	FGSS	TS	109	Method of generating instruction sequences for controlling data flow
4277837	IBM	1981	LGAS	BC	144	Personal portable terminal for financial transactions
5220501	Online Res.	1993	LGAS	BC	142	Method and system for remote delivery of retail banking services
4799156	Strat. Process.	1989	LGAS	BC	138	Interactive market management system
5557518	Citibank	1996	LGAS	BC	54	Trusted agents for open electronic commerce

Patent Number	**Company**	**Year**	**Class**	**Tech.**	**No. of Cites**	**Title**
4922432	Int. Chip	1990	LGAS	CAD	138	Knowledge based method and apparatus for designing integrated circuits using functional specifications
4558413	Xerox	1985	LGAS	CASE	263	Software version management system
4809170	Apollo	1989	LGAS	CASE	145	Computer device for aiding in the development of software system
5155847	Minicom	1992	LGAS	CASE	117	Method and apparatus for updating software at remote locations
5347632	Prodigy	1994	LGAS	Com	136	Reception system for an interactive computer network and method of operation
5333266	IBM	1994	LGAS	Com	119	Method and apparatus for message handling in computer systems
5406557	National Semi-conductor	1995	LGAS	Com	56	Inter-enterprise electronic mail hub
5181162	Kodak	1993	LGAS	DB	198	Document management and production system
5133075	HP	1992	LGAS	DB	164	Method of monitoring changes in attribute values of object in an object-oriented database
4853843	Tektronix	1989	LGAS	DB	122	System for merging virtual partitions of a distributed database
4769772	Bull HN Info. Sys	1988	LGAS	DB	107	Automated query optimization method using both global and parallel local optimizations for materialization access planning for distributed databases

Patent Number	Company	Year	Class	Tech.	No. of Cites	Title
4992940	H-Renee	1991	LGAS	DB	107	System and method for automated selection of equipment for purchase through input of user desired specifications
5499371	Persistence	1996	LGAS	DB	59	Method and apparatus for automatic generation of object oriented code for mapping relational data to objects
5560005	ActaMed	1996	LGAS	DB	57	Methods and systems for object-based relational distributed databases
5437027	Texas Inst.	1995	LGAS	DB	52	System and method for database management supporting object-oriented programming
5008853	Xerox	1991	LGAS	GUI	193	Representation of collaborative multi-user activities relative to shared structured data objects in a networked workstation environment
5572643	Individual	1996	LGAS	GUI	173	Web browser with dynamic display of information objects during linking
4555775	AT&T	1985	LGAS	GUI	170	Dynamic generation and overlaying of graphic windows for multiple active program storage areas
5041992	U. Pittsburgh	1991	LGAS	GUI	169	Interactive method of developing software interfaces
5119475	Schlumberger Tech. Corp.	1992	LGAS	GUI	157	Object-oriented framework for menu definition
4813013	Cadware	1989	LGAS	GUI	149	Schematic diagram generating system using library of general purpose interactively selectable graphic primitives to create special applications icons

Patent Number	**Company**	**Year**	**Class**	**Tech.**	**No. of Cites**	**Title**
4885717	Tektronix	1989	LGAS	GUI	148	System for graphically representing operation of object-oriented programs
5072412	Xerox	1991	LGAS	GUI	120	User interface with multiple workspaces for sharing display system objects
5500929	Taligent	1996	LGAS	GUI	55	System for browsing a network resource book with tabs attached to pages
5524195	Sun Microsys., Inc.	1996	LGAS	GUI	48	Graphical user interface for interactive television with an animated agent
5530852	Sun Microsys., Inc.	1996	LGAS	IR	125	Method for extracting profiles and topics from a first file written in a first markup language and generating files in different markup languages containing the profiles and topics for use in accessing data described by the profiles and topics
4658370	Teknowledge	1987	LGAS	KB	114	Knowledge engineering tool
4591983	Teknowledge	1986	LGAS	KB	113	Hierarchical knowledge system
4649515	Westinghouse	1987	LGAS	KB	111	Methods and apparatus for system fault diagnosis and control
5241671	Encyclopedia Britannica	1993	LGAS	MM	137	Multimedia search system using a plurality of entry path means which indicate interrelatedness of information
5307456	Sony	1994	LGAS	MM	107	Integrated multi-media production and authoring system
4630201	Int. Security	1986	LGAS	Sec	122	On-line and off-line transaction security system using a code generated from a transaction parameter and a random number

Patent Number	Company	Year	Class	Tech.	No. of Cites	Title
5550984	Matsushita	1996	LGAS	Sec	55	Security system for preventing unauthorized communications between networks by translating communications received in ip protocol to non-ip protocol to remove address and routing services information
4953080	HP	1990	LGSS	FS	146	Object management facility for maintaining data in a computer system
5050090	R.J. Reynolds	1991	LGSS	FS	115	Object placement method and apparatus
5329619	Software AG	1994	LGSS	MW	127	Cooperative processing interface and communication broker for heterogeneous computing environments
4819159	Tolerant Sys.	1989	LGSS	MW	120	Distributed multiprocess transaction processing system and method
5497463	Bull HN Info. Sys.	1996	LGSS	MW	52	Ally mechanism for interconnecting non-distributed computing environment (DCE) and DCE systems to operate in a network system
5125091	Hazox	1992	LGSS	RTS	122	Object oriented control of real-time processing

Table 6 describes the various elements useful to assess technical depth and disclosure level (H: high, M: medium, L: low), technical breadth (B: broad, M: medium, N: narrow), and prior art (USP: US patents, FP: foreign patents, NP: non patents). It is ordered by increasing patent#.

Table 6. Technical depth, disclosure level, claims, scope and prior art

Patent#	#Pages	#Figs	#Block diags	#Flow charts	#Tables	Tech. depth	Discl. level	#Claims	Tech. breadth	Prior art			
										#USP	#FP	#NP	Total
4277837	14	8	3			M	L	6	N	13			13
4305131	45	21	6			H	M	49	B	21		5	26
4555775	19	5	2			H	M	15	M	10		3	13
4558413	58	29	6	10		H	M	6	N	1		26	27
4591983	99	13	1	7		H	M	70	B	7		5	12
4630201	11	3		2		M	L	17	N	22	1		23
4649515	21	8	2	6	7	H	M	34	B	2		4	6
4658370	45	11		4		H	M	69	B	6		24	30
4667290	33	2		1		H	M	40	N	2		5	7
4769772	30	8		1		M	L	18	N	18			18
4799156	57	34	2	32		M	L	43	B	9	1	2	12
4809170	20	6	4			M	L	20	N	12	1	1	14
4813013	27	16		11		H	M	15	N	3			3
4819159	30	9	7			H	M	6	N	8			8
4821220	78	19		1		H	H	16	N	3			3
4853843	45	18	8		4	H	M	18	N	7		1	8
4885717	41	12				H	M	17	N	7			7
4922432	26	15	4	3	1	M	L	20	M	10	1	10	21
4953080	438	83	5		9	H	H	24	M	8			8
4992940	16	8	1	1		L	L	12	B	4		1	5

Patent#	#Pages	#Figs	#Block diags	#Flow charts	#Tables	Tech. depth	Discl. level	#Claims	Tech. breadth	Prior art			
										#USP	#FP	#NP	Total
5008853	37	13	2			H	L	20	N	16		24	40
5041992	45	5			5	H	M	17	N	2		6	8
5050090	31	11				M	L	44	N	2		2	4
5072412	51	24	4	15	1	H	M	62	B	19	3	33	55
5093914	25	11		1	11	H	H	21	N	3		6	9
5119475	40	28				H	H	24	N	5		8	13
5125091	11	6	5	1		M	L	9	N	19		3	22
5133075	15	8	2	5		M	M	21	N	8			8
5136705	48	22	12			H	H	9	N	5		7	12
5151987	20	8		5	4	H	M	29	N	1		2	3
5155847	17	4		4		M	L	32	B	12			12
5181162	14	4				M	L	20	B	12			12
5220501	77	22	3	18		H	M	51	B	37	2	45	84
5241671	40	23		22		L	L	41	B	25	2	14	41
5307456	41	30	13	1		H	M	34	B	10		24	34
5329619	64	23	4		35	H	M	26	B	7	1	3	11
5333266	48	19	7	6		H	M	36	N	10			10
5339392	103	47	5	10		H	H	46	M	11	1	10	22
5347632	68	11	2	1		H	M	34	M	16		2	18
5406557	17	7				M	L	16	B	5	3	1	9
5437027	25	7	6	1	12	H	M	21	B	8		13	21

Patent#	#Pages	#Figs	#Block diags	#Flow charts	#Tables	Tech. depth	Discl. level	#Claims	Tech. breadth	Prior art			
										#USP	#FP	#NP	Total
5497463	86	4	1	1		H	M	14	N	8		4	12
5499371	24	8	1	6		H	H	30	N	6		1	7
5500929	18	10	1	1		L	L	19	B	38	1	11	50
5524195	33	15	2	4		M	M	64	B	14	1	3	18
5530852	27	13				M	M	12	N	2		5	7
5550984	7	1				L	L	13	M	26		1	27
5557518	118	44	12			H	L	49	B	26	8	10	44
5560005	69	29	2	7		H	M	85	N	5		12	17
5572643	14	8		1		L	L	19	N	19		7	26

APPENDIX B: DETAILED ANALYSIS OF SELECTED SOFTWARE PATENTS

B1. System software patents

This section analyzes in more detail some representative system software patents from our set. For each granularity and technology, we chose one or more representative patents for which we summarize the invention and discuss its technical depth, disclosure level, prior art, and technical breadth.

B1.1. Fine-grain patents

Programming language support

Patent identification: 5093914, 1992, AT&T Bell Laboratories

Patent title: Method of controlling the execution of object-oriented programs

Number of citations, 1975-1999: 175

The patent describes a method for controlling the execution of an object-oriented program to execute a defined action, e.g., stopping the program, when a specified function is invoked on a specified object during execution of the program. A breakpoint can thus be inserted at the determined function address. The invention is useful for a debugger, a tool that aids software development by giving a user control over and access to a running program.

The patent is twenty-five pages long and has twenty-one claims. It has twelve figures, of which eight are block diagrams and one a flowchart with code skeleton. In addition, it has eleven tables that illustrate program execution control with C++ code fragments. Technical depth and disclosure level are high, thanks to the many detailed code fragments. The patent refers to three previous patents and several published research papers on object-oriented programming debugging. The technical breadth is narrow as the patent describes a specific technique.

Task scheduler

Patent identification: 5136705, 1992, Tektronix, Inc.

Patent title: Method of generating instruction sequences for controlling data flow processes

Number of citations, 1975-1999: 109

The patent describes a method to automatically schedule tasks for changing the states of multiple-state resources that are interconnected, such as those used by computer-controlled test and measurement systems. The resources are modeled as data flow diagrams. The method can produce a sequence of tasks to control the test and measurement systems so as to assure valid data collection and protect physical resources from abuse.

The patent is forty-eight pages long and has nine claims. It has twenty-two figures, of which twelve are flowcharts and one a C++ program example. In addition, it has several detailed Smalltalk code fragments in Appendices. Technical depth and disclosure level are high, thanks to the many detailed code fragments. The patent refers to five previous patents and several published research papers and books on dataflow programming debugging. The technical breadth is narrow as the patent describes a specific technique.

B1.2. Large-grain patents

Video application

Patent identification: 4305131, 1981, Individual
Patent title: Dialog between TV movies and human viewers
Number of citations, 1975-1999: 174

The patent describes a video amusement system by which one or more viewers influence the course of a motion picture as if each viewer were a participant in a real-life drama or dialog. It uses a speech-recognition unit to recognize a few spoken words such as "yes" and "run," thus simulating a dialog between the screen actors and the viewer. The system may read an optical videodisc containing independently addressable video frames, blocks of compressed audio, or animated cartoon graphics. Several hardware circuits are described to control access to video information.

The patent is forty-five pages long and has forty-nine claims. It has twenty-one figures, of which several hardware diagrams for the special circuits and six block diagrams. Technical depth is high. Disclosure level is medium as there are no details about the code. The patent refers to twenty-one previous patents and several published research papers videodisc applications. The technical breadth is broad as the patent touches on various aspects of a video system.

GUI

Patent identification: 5339392, 1994, Individual
Patent title: Apparatus and method for creation of a user definable video displayed document showing changes in real time data
Number of citations, 1975-1999: 109

The patent describes a method and tools for a user to compose a custom document that shows data changes in real time. The method makes it possible to transform a blank display of a computer into an active document having one or more pages of user defined display showing the changing values of data in real time. The tools enable a user to control the "look" of the active document, possibly with script commands. Their use is illustrated with a financial analysis GUI application.

The patent is 103 pages long and has forty-six claims. It has forty-six figures, of which five are block diagrams and ten are flowcharts showing examples of menu class definition. In addition, it describes a scripting language with its full definition and examples. Technical depth is high but the techniques are generally simple. Disclosure level is high, thanks to the many detailed menu description and script examples. The patent refers to twelve previous patents and several published research papers and books on GUI. The technical breadth is medium.

B2. Application software patents

This section analyzes in more details some representative application software patents. For each granularity and technology, we chose one or more representative patents. For each patent, we summarize the invention and discuss its technical depth, disclosure level, prior art and technical breadth.

B2.1. Fine-grain patents

File system

Patent identification: 4953080, 1990, Hewlett-Packard Corporation

Patent title: Object management facility for maintaining data in a computer system

Number of citations, 1975-1999: 151

The patent describes an object-oriented file management system for storing objects belonging to different classes and having different data structures. The use of object data structures allows for the computer to refer to an object data structure and associated access files using a tag that is inaccessible to the user. The user refers to an object based on the physical location of the object on the screen. Additionally, the file management system includes a number of link data structures for complex objects.

This patent is the longest one in our sample set, 438 pages, and contains twenty-four claims. It has eighty-three figures, essentially to illustrate data structures and file organizations. It also includes nine tables to illustrate data structure contents. Technical depth is high. However, disclosure level is medium, as there is no detailed description of file operations. The patent refers to eight previous patents. It has medium technical breadth as it deals with specific techniques of a file system.

Middleware

Patent identification: 5329619, 1994, Software AG

Patent title: Cooperative processing interface and communication broker for heterogeneous computing environments

Number of citations, 1975-1999: 127

The patent describes an object-oriented middleware that supports three modes of inter-object communication: message processing, conversational communication, and remote procedure call. A service broker manages service requests between clients and servers which may reside on different hardware platforms and operating systems and may be connected to heterogeneous computer networks. It includes different application programming interfaces for allowing participants to access the functionality.

This patent is sixty-four pages long and has twenty-six claims. It has twenty-three figures, of which six are block diagrams. It also includes twenty-five tables to define all application programming interfaces in C language. Technical depth is high. However, disclosure level is medium, as there is no detailed description of how the service broker works. The patent refers to eight previous patents and to the standard specifications of the Object Management Group. The technical breadth is broad as the patent describes a large system.

Real-time system

Patent identification: 5125091, 1992, Hazox Corporation

Number of citations, 1975-1999: 122

The patent describes a method of controlling processing in a computer, particularly real-time processing, using computer data objects. Real-time or other input data received from data sources is classified according to pre-stored control data. The control data defines which data source provides the real-time data, how the real-time data is to be processed, where the real-time data is to be stored, and what reports the real-time data will be used in. The classified real-time data becomes a computer data object with its associated control data and all subsequent processing is performed on the computer data object.

This patent is only eleven pages long and has ten claims. It has six figures, of which five are block diagrams and one a flowchart. Technical depth is medium as the presentation is relatively high-level. Disclosure level is low. The patent refers to nineteen previous patents and three research papers. The technical breadth is narrow as the patent describes a specific technique.

B2.2. Large-grain patents

Banking/commerce

Patent identification: 5220501, 1993, Online Resources, Ltd

Patent title: Method and system for remote delivery of retail banking services

Number of citations, 1975-1999: 142

The patent describes a system and method for the remote distribution of financial services, such as home banking and bill paying, using portable terminals connected to a bank's central computer through a network service provider. Information exchange between the central computer and the terminal solicits information from the terminal user related to the requested financial services. The central computer then transmits a message over a conventional ATM network debiting the user's bank account in real time, and may pay the specified payees the specified amount electronically or in other ways as appropriate. The terminal interface is user-friendly.

This patent is seventy-eight pages long and contains fifty-one claims. It has twenty-two figures, of which three are block diagrams and eighteen are flowcharts illustrating program control steps. Technical depth is high. Disclosure level is medium. The patent refers to forty previous patents and many technical papers on electronic fund transfer. The technical breadth is broad as the patent describes a large system.

Patent identification: 5557518, 1996, Citibank, N.A.

Patent title: Trusted agents for open electronic commerce

Number of citations, 1996-1999: 59

The patent describes a system for open electronic commerce having a customer trusted agent securely communicating with a first money module, and a merchant trusted agent securely communicating with a second money module. Both trusted agents can exchange electronic merchandise through a secure session, while both money modules can transfer money through another secure session.

This patent is 118 pages long and has forty-nine claims. It has forty-four figures, of which 12 are block diagrams and thirty-nine, a high level, are protocol commands. Technical depth is high, but disclosure level is low considering the fact that the patent is quite broad and describes a large system. The patent refers to thirty-three previous patents and many technical papers.

CAD

Patent identification: 4922432, 1990, International Chip Corporation

Patent title: Knowledge based method and apparatus for designing integrated circuits using functional specifications

Number of citations, 1975-1999: 138

The patent describes a CAD system and method for designing an application specific integrated circuit that enables a user to define functional architecture-independent specifications and translate them into the detailed information needed for directly producing the integrated circuit. The specifications of the desired integrated circuit are defined in a flowchart format. From the flowchart, the system uses expert system technology to generate a system controller, to select the necessary integrated circuit hardware cells needed, and to generate data and control paths for operation of the integrated circuit.

This patent is twenty-six pages long and has twenty claims. It contains fifteen figures, of which four are block diagrams and three are flowcharts. It also has one table describing micro-operators and examples of expert system rules. Technical depth is medium. Disclosure level is low. The patent refers to eleven previous patents and several papers and books on VLSI design. The technical breadth is medium.

CASE

Patent identification: 4558413, 1985, Xerox Corporation

Patent title: Software version management system

Number of citations, 1975-1999: 267

The patent describes a software version management system that provides for automatically collecting and recompiling updated versions of component software objects over a computer network. The source versions of a particular component software object are represented by models. The component software objects are periodically updated, via a system editor, by various users at their personal computers and then stored in designated storage means. The system supports a notification service to help track the edited objects and alter their respective models to the current version thereof.

This is the most cited patent in our sample set. This patent is fifty-eight pages long and has six claims. It has twenty-nine figures, of which six are block diagrams and ten are flowcharts with interfaces and code skeletons. Technical depth is high. Disclosure level is medium. The patent refers to only one previous patent and several papers and books on source-code-version management. The technical breadth is narrow as the patent describes a number of specific techniques.

Communication

Patent identification: 5347632, 1994, Prodigy Services Company

Patent title: Reception system for an interactive computer network and method of operation

Number of citations, 1975-1999: 136

The patent describes an interactive computer network that enables a user with a PC to display desired information, such as news, financial and cultural information, and perform transactional services, such as banking and shopping. User inputs are translated into PC-independent data objects and executable code objects, which are then processed by the network. User characteristics are monitored by the system in order to generate and display specific advertisements to the user based on individual usage characteristics and predetermined interests.

This patent is sixty-eight pages long and has forty-three claims. It has eleven figures, of which two are block diagrams and one a flowchart. It also includes examples of programs and interface layouts. Technical depth is high. Disclosure level is medium. The patent refers to sixteen previous patents and two books. The technical breadth is medium.

Database

Patent identification: 5181162, 1993, Eastman Kodak Company

Number of citations, 1975-1999: 198

The patent describes an object-oriented document management and production system in which documents are represented as collections of logical components, or objects, that may be combined and physically mapped onto a page-by-page layout. The objects are stored, organized, accessed, and manipulated through a database management system. At a minimum, objects contain basic information-bearing constituents such as text, image, voice or graphics. Objects may also contain further data specifying appearance characteristics, relationships to other objects, and access restrictions.

This patent is only fourteen pages long and has twenty claims. It has four general figures, used, for example, to describe object hierarchies. Technical depth is medium. Disclosure level is low. The patent refers to twelve previous patents. The technical breadth is broad as it describes an entire system.

Patent number: 5133075, 1992, Hewlett-Packard Company

Patent title: Method of monitoring changes in attribute values of object in an object-oriented database

Number of citations, 1975-1999: 166

The patent describes a method of monitoring objects in an interactive object-oriented database system. Any software client can request monitoring of attributes of objects in the database. A record is kept of update transactions initiated by a client. When the client commits the changes, any client who has requested monitoring is notified of any change in the value of an attribute for which monitoring has been requested. The notification interrupts the client and invokes a predestinated client procedure. Overhead is minimized by creating partial view materialization paths and defining monitors ahead of time and by localizing the monitoring.

This patent is fifteen pages long and has twenty-one claims. It has eight figures of which two are block diagrams and two are flowcharts. Technical depth is medium. Disclosure level is medium. The patent refers to eight previous patents. The technical breadth is narrow as the patent describes a specific technique.

Patent identification: 4992940, 1991, H-Renee, Inc.

Patent title: System and method for automated selection of equipment for purchase through input of user-desired specifications

The patent describes an automated system to assist a user in locating and purchasing goods or services sold by several vendors. The system consists of several application programs which access a database containing information about different products and/or services, arranged in various categories. The system also provides the user with an interface to interact with the database and help refining the searching of products. Finally, after the user has selected one or more items for immediate purchase, the system automatically transmits the order to the appropriate vendor.

This patent is sixteen pages long and claims twelve original contributions. It has eight figures, of which one is a block diagram and one a flowchart. Technical depth is low so is disclosure level. The patent refers to four previous patents and one research paper. From a technical point of view it is relatively obvious as it is merely a straightforward purchasing application. The technical breadth is broad as the patent describes a large application.

Patent number: 5499371, 1996, Persistence Software, Inc.

Patent title: Method and apparatus for automatic generation of object oriented code for mapping relational data to objects

Number of citations, 1996-1999: 59

The patent describes a method to map automatically information between an object-oriented application and a structured database, such as a relational database. This is done by taking into account all of the semantics of an object model, such as inheritance and relationships among object classes, and using these semantics to generate a minimal set of routines for each object class that manipulate the object and other objects to which it is related or from which it inherits. By working with the objects, the user of such applications transparently manipulates the database without needing to know anything of its structure.

This patent is twenty-four pages long and has thirty claims. It has eight figures, of which two are block diagrams and six are flowcharts. In addition, it gives the entire (copyrighted) source code of the Persistence system and the user manual in Appendices. Technical depth and disclosure level are thus high. The patent refers to six previous patents and a product's user manual. The technical breadth is narrow as the patent describes a specific technique.

GUI

Patent identification: 5008853, 1991, Xerox Corporation

Patent title: Representation of collaborative multi-user activities relative to shared structured data objects in a networked workstation environment

Number of citations, 1975-1999: 208

The patent describes a multi-user collaborative system in which shared structured data objects can be concurrently accessed by different users at different workstations connected by a network. The system provides a "What you see is what you get" (WYSIWG) user-interface to represent the shared structured data objects and maintains current information relative to the shared structured objects to allow concurrent editing.

This patent is thirty-seven pages long and has twenty claims. It has fourteen figures, of which two are block diagrams. Technical depth is high; in particular, concurrent support through distributed locking. Disclosure level is low. The patent refers to 16 sixteen previous patents and many research papers on collaborative user interfaces. The technical breadth is narrow as the patent focuses on specific techniques for concurrent access to objects.

Patent identification: 5119475, 1992, Schlumberger Technology Corporation

Patent title: Object-oriented framework for menu definition

Number of citations, 1975-1999: 157

The patent describes a mechanism for specifying the behavior, appearance, and function of menus as part of an interactive object-oriented user interface. Menus are constructed from interchangeable object building blocks to obtain the characteristics wanted without the need to write code and maintaining a coherent interface. The approach is implemented by dissecting interface menu behavior into modularized objects.

This patent is forty pages long and has twenty-four claims. It has twenty-eight figures, illustrating menus and window displays; eight of these figures are code fragments in Lisp language. Technical depth and disclosure level are high and result from the use of Lisp, a high-level programming language. The patent refers to five previous patents and many research papers on user interfaces. The technical breadth is narrow as the patent focuses on a specific technique.

Patent number: 5572643, 1996, Individual

Patent title: Web browser with dynamic display of information objects during linking

Number of citations, 1996-1999: 173

The patent describes a method for browsing the World Wide Web using an HTML-compliant client supporting a graphical user interface and a browser. While the client waits for a reply or as the HTML document is being downloaded, the browser displays a "mini-web page" with one or more different types of informational messages to the user, e.g., advertisements, notices, messages, copyright information. Such information is added as comments attached to the links. Such comments are typically ignored by the browser when displaying the document.

This patent is only fourteen pages long and has nineteen claims. It has eight figures, mainly screen displays and HTML samples, and one flowchart. Technical depth and disclosure level are low. The idea of embedding information in HTML comments is quite obvious, and has been already used in SGML, thus potentially making the patent obvious. The patent refers to nineteen previous patents and several trade press papers. The technical breadth is narrow as the patent describes a specific technique.

Information retrieval

Patent number: 5530852, 1996, Sun Microsystems, Inc.

Patent title: Method for extracting profiles and topics from a first file written in a first markup language and generating files in different markup languages containing the profiles and topics for use in accessing data described by the profiles and topics

Number of citations, 1975-1999: 136

The patent describes an information retrieval system and method for automatically creating hypertext documents from information using profiles and topics, and providing that information to a user. The method proceeds in several steps whereby information is extracted using markup languages that describe document content.

This patent is twenty-seven pages long and has twelve claims. It has thirteen figures, illustrating the different steps' input/outputs in SGML markup language. Technical depth and disclosure level are medium. A portion of the disclosure of this patent contains material that is subject to copyright protection and to which a claim of copyright protection has been made. The patent refers to two previous patents and some research papers on SGML. The technical breadth is narrow as the patent describes a specific technique.

Knowledge base

Patent identification: 4658370, 1987, Teknowledge, Inc.

Patent title: Knowledge engineering tool

Number of citations, 1975-1999: 114

The patent describes a tool that is used for knowledge engineers for building and interpreting a knowledge base having control knowledge, factual knowledge, and judgmental rules. The tool has an inference engine applying the judgmental rules according to a built-in control procedure during a consultation with a user. The control knowledge is encoded in an applicative and imperative language. The tool can thus be used to build knowledge systems that can always explain their conclusions and reasoning, and that are intelligible and modifiable.

This patent is forty-five pages long and has sixty-nine claims. It has eleven figures, of which four are flowcharts describing the inference engine subroutines. It also presents ten examples of control blocks and rule programs. Technical depth is high and disclosure level is medium. The patent refers to six previous patents and to many research papers on expert systems. The technical breadth is broad as the patent describes a large system.

Multimedia

Patent identification: 5241671, 1993, Encyclopaedia Britannica, Inc.

Patent title: Multimedia search system using a plurality of entry path means which indicate interrelatedness of information

Number of citations, 1975-1999: 137

This patent describes a search system that retrieves multimedia information in a flexible, user-friendly way. The search system uses a multimedia database consisting of text, picture, audio and animated data. The main database can be searched not only through textual data but also through graphical data. Furthermore, the textual and graphical entry paths are interactive and can be interrelated thus providing flexible searching. Those entry paths include an idea search, a title finder search, a topic tree search, a picture explorer search, a history timeline search, a world atlas search, a researcher's assistant search, and a feature articles search. The patent is described with reference to an encyclopedia, but claims that it can be used with any information that can be stored in a database.

This patent is forty pages long and has forty-one claims. It has twenty-three figures, of which twenty-two are flowcharts. It refers to twenty-seven previous patents and to several technical papers on information retrieval and CD-ROM. Technical depth is rather low as is disclosure level. This patent was difficult to classify as multimedia technology since it could also pertain to database or information retrieval. However, multimedia support is the primary technology. Furthermore, the patent is quite broad.

Security

Patent identification: 4630201, 1986, International Security Note & Computer Corporation

Patent title: On-line and off-line transaction security system using a code generated from a transaction parameter and a random number

Number of citations, 1975-1999: 122

The patent describes a security system for transferring electronic checks. The system includes a central computer and a portable transaction device (PTD) such as a smart card. Electronic checks are generated by the central computer as a sequence of transaction numbers and associated random numbers that are stored in the PTD. When the user issues a check, the PTD generates a security code by combining the next available random number with a transaction parameter. The security code may be verified immediately or during the check clearing cycle at the central computer.

This patent is only eleven pages long and has seventeen claims. It has three figures, of which two are flowcharts. It refers to twenty-three previous patents. Technical depth is medium. Disclosure level is low. The originality seems low, a simple way of generating security codes for authenticating electronic checks. The technical breadth is narrow as the patent bears on a specific technique to generate security codes.