The Strange Odyssey of Software Interfaces and Intellectual Property Law

Pamela Samuelson
UC Berkeley

The Strange Odyssey of Software Interfaces and Intellectual Property Law

Abstract

This book chapter traces the strange odyssey of interfaces through various forms of intellectual property protection. Interface specifications were initially either public domain documents or protected as trade secrets, depending on whether or not they were published. For a time, it seemed as though sui generis protection would be the best way to deal with the interoperability challenges posed by programs, but then copyright became the norm for software protection. Whelan made it seem that interface specifications would be protectable by copyright law as program SSO. Altai and Sega, however, dashed those expectations. Software developers then shifted to patent protection for interfaces, as well as pinning their hopes on the enforceability of anti-reverse engineering clauses in software license contracts. Recent developments give hint of a new shift toward regulated licensing of patented interfaces. No other intellectual artifact has had a comparable tortuous journey through IP law.

The Strange Odyssey of Software Interfaces and Intellectual Property Law

by
Pamela Samuelson*

To make an appliance that can interoperate with the electrical grid, its developer must configure the plug so that it fits neatly into the standard wall socket that has come to be used to connect appliances to the grid. Other aspects of the appliance's design may vary in structure or function, but the plug design has got to be exactly right, or else the appliance won't work. The same principle applies to computer programs, although software interfaces are more complex than the plug and socket interface.

No one would ever imagine that copyright protection would be available for electrical plug and socket designs because they are neither literary nor artistic works. Trade secrecy protection is also impossible because their designs are readily apparent from examining existing plugs and sockets. As machine designs, they might once have been patentable, but any such patent would have expired long ago.

With software interfaces, the obvious analogy to plugs and sockets took many years to take hold. In the first phase of the odyssey of interfaces in relation to IP law, software interfaces were either published and seemingly free of IP protection or were maintained as trade secrets by commercial distribution of programs in unreadable binary code. In the second phase of this odyssey, some firms and IP lawyers took advantage of a Congressional decision to protect programs by copyright law and argued that interfaces were parts of the "structure, sequence, and organization" (SSO) of programs that should be protected by copyright law. After courts rejected copyright protection for interfaces, a third phase began in which patent protection was sought for interfaces. In the current fourth stage of this odyssey, policymakers, courts, and other institutions have taken or are considering some steps to mute the impact of patent protection for interfaces out of concern about competition and follow-on innovation. No other intellectual product has traversed as many forms of IP protection as software interfaces and none has transformed the law so much as it passed through these forms, which is why the tale of this odyssey is worth recounting in a book on Con/texts of Invention. Because copyright was the most pivotal stage in the IP-in-interfaces odyssey, this tale will begin with the key case that brought it to shore.

James Williams did not set out to establish a bold new precedent in intellectual property (IP) law when he talked his old friend Claude Arney into leaving his job as a programmer with Computer Associates International, Inc. (CA) in order to join the programming team at the Texas start-up company, Altai, Inc., which had lured Williams

* Richard M. Sherman Distinguished Professor of Law and Information, University of California at Berkeley. I wish to thank Thomas J. Kearney for his research assistance and Jonathan Band, Robert Barr and Bob Glushko for their comments on an earlier draft.

away from CA some years before.[1] Williams was in charge of the team of programmers tasked with converting Altai's major product, the Zeke scheduling program, to work on IBM computers running the MVS as well as the VSE operating systems. Zeke had originally been designed to run on VSE. Because some of Altai's customers had leased both kinds of IBM mainframes, Altai realized that Zeke would be more attractive to a broader array of customers if it ran on both platforms. Williams considered Arney a good addition to the Altai team because he had worked on the CA-Scheduler program, Zeke's main competitor, and was thus familiar with the tasks that both programs performed.

Shortly after joining Altai, Arney persuaded Williams that the smartest way to redesign Zeke to make it compatible with both VSE and MVS was to build a new compatibility component for Zeke, that is, a sub-program (given the name Oscar) designed to transpose Zeke's commands for specific tasks into the appropriate format for interacting with VSE and MVS so that the operating system could, in turn, properly instruct the IBM hardware to carry out the commands. This new design would avoid the need to customize each module of Zeke for the two operating systems; and if Altai wanted to adapt Zeke in the future to be compatible with additional operating systems, Williams and his team would only need to rewrite parts of Oscar, not the whole of Zeke. Unbeknownst to Williams, CA had adopted the very same approach in the latest version of CA-Scheduler, a project on which Arney had worked when he had been in its employ.

Williams' confidence in Arney's programming prowess seemed well-placed when Arney completed Oscar's VSE compatibility component three months later and the MVS component in an additional month. Altai shipped the Oscar-enhanced version of Zeke from 1985 until August of 1988. These shipments stopped after Computer Associates sued Altai for copyright infringement and trade secret misappropriation, alleging that Oscar contained code misappropriated from CA-Scheduler. Upon learning of the lawsuit, Williams called Arney into his office to ask if the charges had merit. Arney confessed that he had taken a copy of his former employer's source code when he left the firm and had directly copied portions of this code when developing Oscar.[2]

Altai's management decided that the company should take immediate steps to purge Oscar of the tainted code. Williams talked to Arney to find out which parts of Oscar had been copied from CA-Scheduler. He assigned a new team of Altai programmers to revise Oscar. He provided them with a list of Zeke's services and directed them to analyze how to make the services compatible with VSE and MVS, after which they were to write new non-infringing code. Six months later, the team produced a new version of Oscar. Altai then began offering a free "upgrade" of Zeke to its existing customer base as well as selling the revised Zeke to new customers.

---

[1] The facts in this and succeeding paragraphs are derived from Computer Associates Int'l, Inc. v. Altai, Inc., 775 F. Supp. 544 (E.D.N.Y. 1991), aff'd, 982 F.2d 693 (2d Cir. 1992).

[2] Arney may have thought it wasteful to be forced to write new code for Altai when he (or one of his fellow employees) had already produced very efficient code to do the very same set of functions at Computer Associates. However, copying code from CA-Scheduler was a legally risky strategy, as it was in breach of his employment agreement with Computer Associates and would have been illegal even in the absence of this agreement.

Altai accepted it was liable for copyright infringement as to the code directly and literally copied from CA-Scheduler, but believed the rewrite of Oscar had immunized it from further liability. CA, however, asserted that the revised Oscar program was still substantially similar in SSO to the compatibility sub-program of CA-Scheduler, particularly in the manner in which the program interfaces were structured.

Interfaces are important parts of computer programs because they are specially designed to enable the exchange of information between programs and program components. MVS and VSE were both operating system programs for IBM mainframe computers, but the interfaces of the two programs were not the same. That was why it was necessary to develop a subprogram like Oscar to, in essence, translate Zeke's commands into a format that MVS or VSE could comprehend so that the IBM hardware could carry out the scheduling program's commands correctly.[3]

Interfaces of computer programs are unquestionably parts of program SSO.[4] What was unclear before the *Altai* case was whether similarities in program interfaces could be the basis for claims of copyright infringement.[5]

Before delving into how the court in *Altai* resolved this question, it is worth considering where the IP journey for interfaces started. In the early days of computing (i.e., before the mid-1960's), developers of hardware and software, such as IBM, either published documents containing interface information (known as "interface specifications") or maintained interfaces as trade secrets.[6]

Firms often have incentives to publish interface specifications and encourage others to make unrestricted use of them so that customers and developers of software or peripheral equipment could produce or adapt programs and other products to operate on that hardware or with the software installed on that hardware. Some firms, however, decided to maintain interfaces as trade secrets in order to control which software and peripheral equipment could be developed for their platforms.[7]

It is well-known that a positive feedback loop can promote the success of computing platforms: the larger is the number of applications available for a platform, the more attractive, in general, that system will be in the marketplace, and the more customers the platform has, the stronger are incentives for software developers to write programs for that platform. This feedback loop, which is often referred to as exhibiting

---

[3] See, e.g., Michael Jacobs, *Copyright and Compatibility*, 30 Jurimetrics J. 91 (1989)(discussing program interfaces and the commercial importance of compatibility).
[4] Id. at 103.
[5] Id. at 103-04.
[6] See, e.g., JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL 19 (1995).
[7] This strategy has been common in videogame industry. See, e.g., DAVID SHEFF, GAME OVER: HOW NINTENDO ZAPPED AN AMERICAN INDUSTRY, CAPTURED YOUR DOLLARS, AND ENSLAVED YOUR CHILDREN 286-91 (1993) (discussing Nintendo's closed platform and efforts to stave off unlicensed compatible games).

"network effects," has implications for incentives for firms to publish (or not) interface information.[8]

IBM, for example, initially published interface specifications and provided customers with source code to make its computer systems more attractive to those customers. As it became a dominant firm in the computer industry, it realized that interfaces were commercially valuable in their own right. It then began to treat interface specifications as trade secrets.[9] This not only gave IBM control over development of compatible software and peripheral equipment; it also impeded development of competing platforms capable of interoperating with applications written for IBM computers.[10]

For some years, IBM bundled its proprietary hardware, software, and peripherals together and ceased distributing source code and interface specifications to its customers. Even after IBM began unbundling software and peripherals, in part under pressure from antitrust authorities,[11] it did not publish its interfaces, but rather licensed these trade secrets on royalty-bearing terms to developers of software and peripheral equipment.

Some firms sought to avoid entering into such trade secret licensing agreements by undertaking the tedious and time-consuming task of reverse-engineering other firms' programs to extract interface information in order to make compatible, if unlicensed, products.[12] To block this form of unlicensed access to program interfaces, many firms have inserted anti-reverse engineering clauses into their license agreements.[13]

Prior to 1980, it was unclear whether either patent or copyright protection would or should be available to protect computer programs or their interfaces. In the 1970's, two Supreme Court decisions ruled that certain program-related inventions were ineligible for patent protection.[14] Although the Copyright Office began accepting registrations of computer programs in 1965, it did so under its "rule of doubt;" indeed, the registration certificates indicated the Office's doubt about the copyrightability of programs in machine-executable form.[15] Unsurprisingly, few programmers registered

---

[8] Joseph Farrell, *Standardization and Intellectual Property*, 30 Jurimetrics J. 35 (1989).
[9] See, e.g., Band & Katoh, supra note 6, at 20-25.
[10] Fujitsu was one of IBM's competitors that developed an IBM-compatible platform, which IBM challenged as infringing its copyrights. Id. at 27-28.
[11] Id. at 22. European competition law authorities charged IBM with abusing its dominant position by changing interfaces in a manner that rendered IBM-compatible peripherals inoperable; IBM settled the lawsuit by agreeing to pre-disclose changes to its interfaces to aid makers of peripherals in adapting their products in a timely manner. See F.M. Scherer, *Thinking About the European Microsoft Case: F.M. Scherer Discusses the Relevance of the IBM Case*, 84 Antitrust & Trade Reg. Rep. 2090 (Jan. 24, 2003).
[12] See, e.g., Sega Enter. Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992).
[13] See, e.g., Vault Corp. v. Quaid Software Ltd., 847 F.2d 255 (5th Cir. 1988).
[14] See Parker v. Flook, 437 U.S. 584 (1978) (computer program to update alarm limits for catalytic conversion); Gottschalk v. Benson, 409 U.S. 63 (1972).
[15] Copyright Office Circular 31-D (1965), reprinted in Duncan M. Davidson, *Protecting Computer Software: A Comprehensive Analysis*, 1983 ARIZ. ST. L.J. 611, 652 n.72.

copyright claims in this period,[16] and no litigation tested whether computer programs were copyrightable subject matter until the very end of the 1970's.[17]

Doubts about the patentability of programs arose because programs are texts and because many information innovations embedded in programs, such as algorithms, are "mental processes" (that is, processes that can be carried out in the human mind or with the aid of a pen and paper).[18] Patent law had long excluded "printed matter," such as texts, from the scope of its protection,[19] even though printed matter is a manufactured artifact, and hence literally within the meaning of the term "manufacture," one of the four categories of inventions for which patents may issue.[20] Mental processes are likewise processes that would literally seem to be patentable subject matter;[21] yet, the U.S. Patent & Trademark Office (PTO) and the courts have long regarded mental processes as unpatentable.[22] When faced with a patent claim for an algorithm for transforming binary coded decimals to pure binary form, the Supreme Court in Gottschalk v. Benson ruled that this mathematical process was ineligible for patent protection, suggesting that processes should be eligible for patent protection only if they transform matter from one physical state to another.[23]

Computer programs, by contrast, seemed copyrightable in human-readable source code form or as depicted in flow charts, but once transformed into machine-executable form, programs seemed uncopyrightable as functional processes.[24] Copyright cases dating back to the 1880's excluded useful arts, such as machines and mechanical processes, from the scope of copyright protection.[25] The Supreme Court had stated that "[t]o give to the author of the book an exclusive property in the [useful] art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of

---

[16] See NAT'L COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1979) (hereinafter "CONTU Report") (reporting that about 1200 programs had been registered between 1965 and 1978). Another disincentive to registration, apart from the doubt reflected in the certificate, was that the Office required deposit of the full source code of programs as a condition of registration, which would make it impossible to claim trade secret protection for program source code.

[17] Data Cash Sys., Inc. v. JS&A Group, Inc., 480 F. Supp. 1063 (N.D. Ill. 1979)(ruling that machine-executable programs were not copyrightable subject matter, analogizing source code to architectural drawings and object code to buildings constructed from those drawings, which at that time were unprotected by U.S. copyright law), aff'd on other grounds, 628 F.2d 1038 (7th Cir. 1980).

[18] See, e.g., Pamela Samuelson, Benson *Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 Emory L. J. 1025 (1990).

[19] See, e.g., In re Rice, 132 F.2d 140 (C.C.P.A. 1942).

[20] 35 U.S.C. sec. 101.

[21] Id.

[22] See, e.g., In re Abrams, 188 F.2d 165 (C.C.P.A. 1951).

[23] *Benson*, 409 U.S. at 64-70. See also In re Comiskey, 499 F.3d 1365 (Fed. Cir. 2007)(arbitration process held unpatentable non-technological process).

[24] See, e.g. Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 Duke L. J. 663 (1984).

[25] Baker v. Selden, 101 U.S. 99 (1880)(bookkeeping system and other "useful arts" are ineligible for copyright protection, although they may qualify for patents). For a discussion of *Baker* and its progeny, see, e.g., Pamela Samuelson, *Why Copyright Excludes Systems and Processes From the Scope of Its Protection*, 85 Tex. L. Rev. 1921 (2007).

copyright."[26] In this and other cases, the courts have made a sharp distinction between the provinces of copyright and utility patent law, perceiving no overlap in their subject matters.[27]

A key turning point in the evolution of IP protection for computer programs came in 1979 when the National Commission on New Technological Uses of Copyrighted Works (CONTU) issued a report to Congress.[28] CONTU had been established to consider the implications of several new technology issues for copyright law, including photocopying, inter-library loans, electronic database protection, and inputting of books into computers.[29] CONTU went beyond its initial charter in recommending that copyright protection should be available for computer programs as "literary works."[30]

CONTU expressed confidence that copyright law could evolve to make appropriate distinctions between program ideas (which of course would not be protectable) and program expression (which would be).[31] Perhaps because the CONTU Commissioners were mostly copyright lawyers who knew very little about computer programs, the Report did not meaningfully address important scope of protection issues, including the protectability (or not) of program interfaces.[32] In 1980, Congress amended copyright law, as CONTU recommended.[33]

Even after the 1980 amendments, there were some vestiges of doubt about the copyrightability of programs. Two appellate court decisions snuffed out these doubts by rejecting challenges to copyrights in Apple Computer's operating system programs.[34] Makers of clone computers claimed, among other things, that it was necessary to copy the Apple II operating system programs in order for their computers to achieve interoperability with programs written for the Apple platform.[35] One court observed:

> Franklin may wish to achieve total compatibility with independently developed application programs written for the Apple II, but that is a commercial and competitive objective which does not enter into the

---

[26] *Baker*, 101 U.S. at 102.
[27] See, e.g., Pamela Samuelson, Baker v. Selden: *Sharpening the Distinction Between Authorship and Invention*, in INTELLECTUAL PROPERTY STORIES (Rochelle C. Dreyfuss & Jane C. Ginsburg, eds. 2005).
[28] CONTU Report, supra note 16.
[29] Pub. L. No. 93-573, Title II (establishing CONTU).
[30] CONTU Report, supra note 16, at 1, 9-26.
[31] Id. at 18-23.
[32] See, e.g., Peter Menell, *Tailoring Legal Protection for Computer Programs*, 39 Stan. L. Rev. 1329 (1985) (critical of CONTU for failing to consider compatibility-related issues); Samuelson, supra note 24 (criticizing CONTU for its misleading and incorrect statements about computer programs and likely difficulty of applying copyright to programs because of their functionality).
[33] Pub. L. No. 96-517, 94 Stat. 3007, 3028 (codified at 17 U.S.C. §§ 101, 117 (1982)).
[34] See Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983); Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d 521 (9th Cir. 1984).
[35] *Franklin*, 714 F.2d at 1245-46. Franklin also argued that the CONTU Commission had only intended to protect application programs that interacted with people, not purely functional programs such as operating systems. Id. at 1246-52.

somewhat metaphysical issue of whether particular ideas and expressions have merged.[36]

This did not augur well for future defenses to claims of infringement based on copying of interfaces.[37]

Even more worrisome for compatible program developers like Altai were decisions such as Whelan Associates, Inc. v. Jaslow Dental Lab., Inc.[38] *Whelan* involved a dispute between two former partners about a dental laboratory business program. After a falling out, Jaslow decided to develop his own version of the Dentalab program. Although his program and hers were written in different programming languages and used different algorithms, the overall structure of the programs was similar, as were some data and file structures, and the two programs performed some of the same functions in the same manner. The lower court found Jaslow to have infringed Whelan's copyright. Jaslow's principal defense on appeal was that copyright protection was only available for source and object code, not for program SSO. The Third Circuit Court of Appeals in Pennsylvania affirmed.

Whelan agreed with CONTU that computer programs should be regarded as "literary works." It reasoned that since copyright law had long protected non-literal elements (i.e., structure and organization) of other types of literary works, such as novels and plays, it should protect the SSO of programs as well.[39] *Whelan* deemed all program SSO to be protectable by copyright law as long as there was more than one way to structure a program to achieve the program's functions.[40] If there was only one way to structure a program to perform particular functions, then the "idea" of that function and its structural "expression" should be considered "merged," and treated as among the unprotectable "ideas" in programs. The court in *Whelan* also endorsed extending copyright protection to the "look and feel" of programs, which seemingly included the manner in which the programs behaved (i.e., how they performed their functions).[41] Without broad copyright protection for computer programs, and in particular, for aspects of program SSO that were costly and difficult to develop as well as commercially significant, the court worried that there would be too little protection to provide proper incentives to develop computer programs.

CA relied heavily on *Whelan* and its progeny in arguing that the revised Oscar program infringed its copyright in CA-Scheduler.[42] It pointed to substantial similarities between the compatibility components of Zeke and CA-Scheduler, especially as to their

---

[36] Id. at 1253.
[37] The court responded to the CONTU intent-based argument by pointing out that the Apple software at issue were computer programs within the definition of that term in the copyright statute and CONTU had not meant to exclude operating system programs from the scope of copyright protection. Id. at 1251.
[38] 797 F.2d 1222 (3d Cir. 1986).
[39] Id. at 1234.
[40] Id. at 1236.
[41] Id. at 1228, 1247.
[42] See, e.g., Reply Brief for Plaintiff-Appellant, in Computer Associates Int'l, Inc. v. Altai, Inc., 1991 WL 11010234 (relying heavily on *Whelan*).

parameter lists (i.e., lists of information that needed to be sent and received by subroutines of the affected programs). These elements of program SSO had been carefully and precisely designed, making them costly to develop and commercially significant parts of programs. CA argued that incentives to invest in software development would be undermined if competitors such as Altai could appropriate program SSO without fear of liability. Parameter lists and other SSO elements of program interfaces are, moreover, complex and detailed, not abstract in content, which under *Whelan* made them protectable expression. In view of the SSO similarities, CA argued the revised Oscar still infringed the copyright in CA-Scheduler.

Altai faced a daunting challenge to parry Computer Associates' rhetorical thrusts. The *Apple* case was easy to distinguish because Franklin had not even tried to rewrite the Apple OS programs, but had instead copied the code exactly, bit for bit.[43] Altai, by contrast, had spent hundreds of man-hours and many months developing a new implementation of Zeke's interfaces to the IBM programs. *Whelan*'s analysis, however, could only be elided by developing a strong counter-rhetoric.

Altai sought to convince the court to conceptualize computer programs as utilitarian works (which were, therefore, meaningfully different from novels and plays).[44] It could then point to caselaw and other sources saying that utilitarian works enjoy, at best, a "thin" scope of copyright protection (that is, protection against only exact or near-exact copying).[45] Even more important was getting the court to recognize that external factors sometimes constrain the design choices of programmers. This includes not only the mechanical specifications of the computer hardware on which a program was designed to run, but also interface information,[46] such as the IBM protocols that enabled programs such as CA-Scheduler and Zeke to exchange information and interoperate with the MSV and VSE operating systems. Of course, the parameter lists of CA-Scheduler and Zeke were similar; both aimed to provide the same scheduling services to their customers and both were trying to interoperate with MVS and VSE.[47]

Altai won the rhetorical war before the Second Circuit Court of Appeals in New York and established an important precedent. The *Altai* decision not only rejected claims of copyright protection in interfaces, but also adopted a now widely used three step test for assessing claims of copyright infringement in computer programs. The first step involves constructing a hierarchy of abstractions, from most abstract to most detailed, for the plaintiff's program. The second step involves "filtering" out of consideration various elements of the program that are beyond the scope of copyright protection. The third step involves comparing any remaining "golden nuggets" of expression in the plaintiff's program with the defendant's program to determine if the defendant copied substantial amounts of expression from the plaintiff's program.[48]

---

[43] *Franklin*, 714 F.2d at 1245.
[44] *Altai*, 982 F.2d 704 ("the essentially utilitarian nature of computer programs").
[45] Id. at 704-05 (comparing computer programs to useful arts, such as recipes and bookkeeping systems).
[46] Id. at 709-10.
[47] See Brief of Defendant-Appellee, Computer Associates Int'l, Inc. v. Altai, Inc., 1991 WL 11010232 (making this argument).
[48] *Altai*, 982 F.2d at 706-11.

Application of the abstraction-filtration-comparison test generally results in programs having a thin scope of copyright protection because the second step requires filtering out three kinds of unprotectable elements. The first consists of elements of program design dictated by efficiency.[49] Hypothetically, there may be many ways to achieve certain program functions, but efficiency considerations will often narrow the range of practical solutions. Because programmers are constantly striving to achieve efficiency, adopting the same efficient solution may, moreover, be the product of independent work, not of copying. The second category of unprotectables includes design choices that are constrained by external factors, such as the hardware and software with which the program was designed to operate, demands of the industry being served, and widely accepted programming practices.[50] The third includes elements of programs that are in the public domain, such as commonplace programming techniques, ideas, and know-how.[51] Later decisions have extended *Altai* by directing courts to filter out functional design elements, such as procedures, processes, systems, and methods of operation.[52]

The court in *Altai* asserted that its test for software copyright infringement "not only comports with but advances the constitutional policies underlying the Copyright Act."[53] It recognized that CA might be right that thin copyright protection for programs could undermine incentives to invest in program development, but the Supreme Court had "flatly rejected" similar incentive-based arguments for broad copyright protection of factual works in Feist Publications, Inc. v. Rural Telephone Service Co.[54] To extend broad copyright protection to program SSO would "have a fundamentally corrosive effect on certain fundamental tenets of copyright doctrine."[55] Because copyright seemed ill-suited to protecting program innovations, the court in *Altai* suggested that Congress consider whether programs needed additional IP protection; it also suggested that patents might be a more suitable form of IP protection for program SSO.[56]

The *Altai* decision may not initially have induced software developers and their lawyers to start thinking seriously about patenting interfaces and other program SSO, in part because it took some years for *Altai* to defeat *Whelan* in the subsequent caselaw and emerge as the leading decision for judging claims of software copyright infringement.[57] However, the patent option became more urgent after the Ninth Circuit Court of Appeals

---

[49] Id. at 707-09.
[50] Id. at 709-10.
[51] Id. at 710.
[52] See, e.g., Gates Rubber Co. v. Bando Chemical Indus. Ltd., 9 F.3d 823 (10th Cir. 1993).
[53] *Altai*, 982 F.2d at 711.
[54] Id., citing Feist Pub., Inc. v. Rural Telephone Service Co., 499 U.S. 340 (1991)(rejecting "sweat of the brow" copyright claim in white pages listings of a telephone directory).
[55] *Altai*, 982 F.2d at 712. The court criticized *Whelan* for its unduly broad conception of the scope of copyright in computer programs, for its reliance on metaphysical distinctions rather than practical considerations, and for its outdated comprehension of computer science. Id. at 705-06.
[56] Id. at 712.
[57] *Altai* has been followed in at least 49 subsequent decisions.

in California issued its ruling in Sega Enterprises, Ltd. v. Accolade, Inc.,[58] decided less than a month after *Altai*.

Sega was important in the IP-in-interfaces odyssey for at least four reasons. For one thing, it embraced *Altai*'s rhetorical approach to conceptualizing computer programs as utilitarian works eligible for only a thin scope of copyright protection.[59] Second, *Sega* followed *Altai* in ruling that program interfaces were elements of programs that copyright law did not protect; indeed, *Sega* spoke of interface information as "functional requirements for achieving compatibility with other programs."[60] Third, the court ruled that copying program code in the course of reverse engineering it for a legitimate purpose such as extracting interface information to make a compatible program did not infringe any copyright in that code.[61] The court reasoned that

> [i]f disassembly of copyrighted object code is per se an unfair use, the owner of the copyright gains a de facto monopoly over the functional aspects of his work—aspects that were expressly denied copyright protection by Congress. In order to enjoy a lawful monopoly over the idea or functional principle underlying a work, the creator of the work must satisfy the more stringent standards imposed by the patent laws.[62]

Fourth, it indicated that even copying some exact code from another program would not be infringement insofar as that code was essential to achieving interoperability.[63]

After *Sega*, developers could no longer hope to protect interfaces by copyright. And because *Sega* allowed unlicensed reverse-engineering of code to extract interface information,[64] it imperiled developer efforts to protect its interfaces as trade secrets. *Sega* signaled that the only reliable means for protecting the functional requirements for achieving interoperability was by patenting them. Patents had at least one advantage over copyright law in protecting interfaces because patent law has no "merger" doctrine. Hence, if there is only one way to achieve a particular function and a developer has patented that one way, it can exercise its patent rights to stop unlicensed uses.

Altai and *Sega* contributed to the eventual shift away from reliance on copyright protection for program SSO and interface innovations and toward reliance on patent protection. But developments on the patent side also made this form of protection for program SSO more plausible than in the 1970's. Especially important was the Supreme Court's 5-4 decision Diamond v. Diehr in 1981, which signaled a new receptivity to

---

[58] 977 F.2d 1510 (9th Cir. 1992).
[59] See, e.g., id. at 1526 ("Under the Copyright Act, if a work is largely functional, it receives only weak protection.")
[60] Id. at 1525-26.
[61] Id. at 1527-28 (holding that reverse engineering copies qualified as fair uses).
[62] Id. at 1525.
[63] Id. at 1516. See also id. at 1528-32 (treating some Sega code too functional for IP protection).
[64] Prior to *Sega*, some commentators had argued that reverse engineering of object code should be treated as both copyright infringement and trade secret misappropriation. See, e.g., Allen Grogan, *Decompilation and Disassembly: Undoing Software Protection*, COMPUTER LAW., Feb. 1984, at 1.

patenting computer program-related inventions.[65] Diehr claimed to have invented a new method of curing rubber that used a computer program to calculate when the temperature inside rubber molds had reached the proper curing point such that the molds should be opened. The PTO had rejected Diehr's claim because its only novelty lay in program calculations. Under earlier Supreme Court decisions, the Office thought this process was ineligible for protection.[66]

In *Diehr*, the Supreme Court rejected the "point of novelty" test and ruled that Diehr's process claimed patentable subject matter. Because the Court was so deeply divided, because the majority opinion did not repudiate the Court's earlier rulings on the unpatentability of certain program innovations, and because the case involved a traditional manufacturing process (i.e., curing rubber), the *Diehr* decision was initially perceived as a modest change in the patent landscape as to program-related inventions.

Certain language in *Diehr*, however, embraced a broad conception of patentable subject matter.[67] In the decade or so after *Diehr*, the intermediate appellate court that hears patent cases developed a very broad conception of patentable subject matter under which virtually all computer program-related inventions were patentable subject matter.[68] This, coupled with increasing "thinness" of copyright protection after *Altai* and *Sega* achieved widespread acceptance in the mid-1990's, led to big surge in patenting of software innovations.[69]

Software developers often seek patents for program internals, such as algorithms and data structures. Such patents are, however, generally more useful for defensive than for offensive purposes. That is, developers tend to seek patents on such internal design elements to assure themselves of having freedom to develop software embodying these innovations as well as to build a portfolio of IP assets so that the firms will have something to trade (e.g., by cross-licensing) if a competitor asserts patent claims against them.[70] Patents on program internal designs are often difficult to assert offensively (that is, to stop competitors from using them) because such elements are typically difficult to discern in commercially distributed object code. Because infringement is difficult to detect, patents on internal program designs are difficult to enforce.

---

[65] 450 U.S. 175 (1981).
[66] See, e.g., *Benson*, 409 U.S. 63.
[67] *Diehr*, 450 U.S. at 181 (patentable subject matter includes everything under the sun made by man).
[68] See, e.g., AT&T Corp. v. Excel Comm'ns, Inc., 172 F.3d 1352 (Fed. Cir. 1990).
[69] See Josh Lerner & Feng Zhu, *What is the Impact of Software Patent Shifts? Evidence from* Lotus v. Borland 10 (Nat'l Bur. Econ. Res. Working Paper No. 11168 2005) (presenting evidence of surge in patenting of software in the mid-1990's).
[70] See, e.g., Gideon Parchomovsky & R. Polk Wagner, *Patent Portfolios*, 154 U. Pa. L. Rev. 1 (2005). Software patents may be useful to firms in obtaining financing from venture capitalists. Ronald Mann, *Do Patents Facilitate Financing in the Software Industry?*, 83 Tex. L. Rev. 961, 972 (2007).

Patents on interface designs, by contrast, are more likely to be useful for offensive purposes.[71] They may confer on their owners an exclusive right to control the development not only of competing, but also of complementary, products, because the interface defines the boundaries between and among programs. It is generally easy to detect infringement of interface patents because if unlicensed products successfully interoperate with the patentee's products, they almost certainly infringe. Interface patents are among the most valuable patents that software developers can own, in part because such patents can be impossible to work around. Even a narrowly drawn interface patent may preclude interoperability as to a key component of the program.

Software interfaces are so essential to achieving interoperability that some have suggested that they should be unpatentable, or if patented, their use should be deemed non-infringing insofar as there is no equally efficient or effective way to achieve interoperability.[72] (The latter approach would, in essence, create a merger doctrine in patent law.) A similar approach is reflected in the policy adopted by the World Wide Web Consortium (W3C). W3C requires member firms, which include major industry players such as Microsoft and IBM, to agree that if they own patents that "read" on a standard adopted by W3C that is essential to interoperability on the Web, those patents will be licensed on a royalty-free (RF) basis.[73] OASIS (Organization for the Advancement of Structured Information Standards) does not mandate RF licensing of webservices interface patents, but offers two widely used RF licensing options for technical committees to adopt.[74] RF policies do not, of course, make such patents unenforceable, but they do substantially reduce the leverage that such patents would otherwise provide as well as their economic value. This, in turn, dampens incentives to acquire such patents. Some commentators have, moreover, called for abolition of software patents, [75] in part because interface patents pose such risks to open source programming.

An alternative strategy is to mute the exclusionary character of interface patents is to allow their use, but to oblige users to compensate the patent's owner. This liability rule approach can be implemented in a number of ways. Under the Supreme Court's ruling in eBay, Inc. v. MercExchange, L.L.C., [76] courts have discretion to withhold injunctive relief in cases involving interfaces essential to interoperability and order payment of a reasonable royalty. In addition, the U.S. government has power to practice patented inventions and to authorize others to do the same as long as it provides

---

[71] See, e.g., Atari Games Corp. v. Nintendo of Am., Inc., 1993 U.S. Dist. LEXIS 8864, *3-*6 (N.D. Cal. June 30, 1993); Atari Games Corp. v. Nintendo of Am., Inc., 30 U.S.P.Q.2d (BNA) 1401, 1414 (N.D. Cal. 1993) (granting summary judgment to Nintendo because Atari had infringed an interface patent).
[72] See, e.g., Band & Katoh, supra note 6, at 332-34.
[73] W3C Patent Policy, Feb. 4, 2005, available at http://www.w3.org/Consortium/Patent-Policy-20040205/.
[74] See OASIS Intellectual Property Rights Policy, http://www.oasis-open.org/who/intellectualproperty.php. Robert J. Glushko, who serves on the OASIS Board, reports that virtually all of the TCs now use one of the two RF policies. Conversation with Robert J. Glushko, March 2, 2008.
[75] See, e.g., Brad Feld, *Abolish Software Patents*, Feld Thoughts, April 10, 2006, available at http://www.feld.com/blog/archives/2006/04/abolish_softwar.html.
[76] 126 S. Ct. 1837 (2006).

reasonable compensation for the use.[77] Antitrust authorities could also require licensing of interface patents.[78]

Japan has been considering a proposal to require licensing of patents on essential interfaces.[79] The European Parliament was asked to consider a similar proposal to oblige owners of patents on essential interfaces to license them on reasonable and non-discriminatory (RAND) terms during the contentious debate over the proposed directive on software patents.[80] Some standard-setting organizations require participating firms to agree in advance to license on RAND terms any patents that may be implicated by a standard adopted by those organizations.[81] The widespread practice of cross-licensing of software patents among large and medium-sized firms in the software industry similarly may lessen the exclusionary impacts of interface patents.

The story of the strange odyssey of software interfaces in IP law would not be complete without a brief discussion of an eddy into which some policy makers and commentators were swept up in the early 1980's and early 1990's before copyright emerged as the international standard for protecting programs.[82] During that time, there was considerable international interest in crafting a sui generis (of its own kind) form of IP protection to computer programs. In 1983, for example, the Japanese Ministry of International Trade and Industry (MITI) issued a report recommending a sui generis form of protection for software. The regime resembled copyright in some ways (e.g., automatic protection against unauthorized copying), but was shorter in duration and took into account the importance of interoperability, which copyright law could not readily do.[83] The European Commission also expressed interest in a sui generis form of IP protection for software.[84]

---

[77] 28 U.S.C. sec. 1498.

[78] See, e.g., Robert P. Merges & Richard R. Nelson, *On the Complex Economics of Patent Scope*, 90 Colum. L. Rev. 839 (1990) (giving examples of licenses induced by antitrust oversight).

[79] Ministry of Economy, Trade, & Industry, Press Release, Interim Report of the Study Group on the Legal Protection of Computer Programs and Promotion of Innovation, Oct. 11, 2005, available at http://www.meti.go.jp/english/information/data/051011SoftInnove.html. ("Industry could consider the propagation of some concept along the lines of "Creative Commons." Action should be taken to popularize, through agreements among enterprises, the business practices of mutual non-assertion of rights to such patented inventions as relating to certain categories of software, such as OSS, or to interoperability of software, thereby making this concept the standard in the public domain by utilizing the current patent office. Moreover, the compulsory licensing system and the enhanced application of the antimonopoly law are considered as further issues to be studied.") See also Interpretative Guidelines for Electronic Commerce (revised March 2007), 192-193, 201 (to assert patent rights to defeat interoperability may be considered an abuse of right), available at http://www.meti.go.jp/english/information/data/IT-policy/interpretative_guidelines_on_ec070628.pdf

[80] Foundation for a Free Information Infrastructure, Plenary Amendments: Interoperability, 9/16/05 (on file with the author).

[81] Mark A. Lemley, *Intellectual Property and Standard-Setting Organizations*, 90 Cal. L. Rev. 1889 (2002).

[82] See Agreement on Trade-Related Intellectual Property Rights, art. 10.

[83] See Dennis S. Karjala, *Lessons from the Computer Software Protection Debate in Japan*, 1984 Ariz. St. L.J. 53, 63.

[84] Green Paper on Copyright and the Challenge of Technology—Copyright Issues Requiring Immediate Action, Document COM (88), 172 final 7 June 1988. Although the EU ultimately chose to protect programs by copyright law, it adopted key sui generis features in its software protection directive by

Some American commentators also argued for sui generis protection for software.[85] One such article was "A Manifesto Concerning the Legal Protection of Computer Programs," which reported the results of a collaboration among Lotus Development Corp. founder Mitchell Kapor, computer scientist Randall Davis, law professor Jerome Reichman, and me.[86] It challenged the then-prevailing conception of computer programs as literary works and characterized programs as machines that happen to be constructed in text. The Manifesto pointed out that the most valuable aspects of programs lie not what they say or how they say it, but what the programs do and how well they do it. It proposed that the "industrial compilations of applied know-how" in computer programs, including their behaviors, should be protected from market-destructive appropriations.[87] It proposed a short term right to exclude others from cloning industrial compilation components of programs, followed by a term in which others could use these components subject to a right of compensation.

Interfaces are among the elements of programs that are best understood as industrial compilations of applied know-how. Because they are carefully drafted precise and detailed compilations of information, interfaces resemble copyright subject matter. Like copyright subject matter, they are relatively cheap and easy to copy, especially in digital form, and so seem amenable to the ban on unauthorized copying that is copyright law's principal hallmark. However, copyright law does not protect industrial compilations, such as interfaces, rule sets, recipes, and systematic organizations of information; nor does it protect know-how.[88] It is consequently unsuitable as a form of protection for interface specifications.

Although some firms patent interfaces, the textual nature of interface specifications and their information-intensive component parts make patents unsuitable for protecting interface specifications as such.[89] A sui generis regime focused on protecting interfaces as industrial compilations of applied know-how would be a more suitable regime for interface protection than patents.

Whatever the merits of a sui generis approach, the IP odyssey passed it by. Software developers now use copyrights to protect program code and expressive aspects of audiovisual displays (e.g., videogame animation). Distributing programs in object code form generally provides effective trade secret protection for internal designs,

---

excluding interfaces from the scope of protection and permitting decompilation for purposes of acquiring interface information. See Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs, Art. 5-6.

[85] Karjala, supra note 83; Menell, supra note 32, at 1364-65.

[86] Pamela Samuelson, et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 Colum. L. Rev. 2308 (1994).

[87] See also J.H. Reichman, *Computer Programs as Applied Scientific Know-How: Implications of Copyright Protection for Commercialized University Research*, 42 Vand. L. Rev. 639 (1989).

[88] See, e.g., Samuelson, supra note 25, at 1928-52 (discussing caselaw and policy rationale for exclusion of these innovations from the scope of copyright and legislative history for including a statutory exclusion).

[89] See supra notes 18-23 and accompanying text.

including interfaces.[90] Licensing programs on terms that forbid reverse-engineering gives developers reason to hope for a contractual bypass of the *Sega* decision.[91] While firms sometimes get patents on novel interface designs, their ability to exercise patents that implicate interoperability has been somewhat lessened in ways recounted above.

No other intellectual artifact has had a comparable journey through IP law, transforming each form as interfaces have passed through them. Interface specifications began as public domain documents or as trade secrets, depending on whether or not they were published. For a time, it seemed as though sui generis protection would be the best way to deal with the interoperability challenges posed by programs, but then copyright became the norm for software protection. *Whelan* made it seem that interface specifications would be protectable by copyright law as program SSO. *Altai* and *Sega*, however, dashed those expectations. Software developers then shifted to patent protection for interfaces, as well as pinning their hopes on the enforceability of anti-reverse engineering clauses in software license contracts. Recent developments give hint of a new shift toward regulated licensing of patented interfaces.

This strange odyssey of interfaces through various forms of IP law offers some insights worthy of inclusion in this volume. Interface specifications for software are an information innovation that does not fit neatly in traditional copyright and patent bins.[92] Those who sought to protect interfaces by copyright law emphasized their resemblance to other copyright subject matters. Those who fought against copyright protection for interfaces had to convince courts that interfaces were functional requirements for achieving interoperability, akin to electrical plugs and sockets, which seemed more appropriate for patent protection. Yet, interfaces are so important to competition and follow-on innovation that there is some reluctance to give patentees a free hand in exercising their rights under this law. Even though developing interfaces is expensive, time-consuming, and intellectually challenging, the huge success of the software industry in the past thirty years suggests that adequate incentives to develop computer programs do exist, even without IP protection for one commercially significant component, namely, interfaces.

The copyright part of the IP odyssey might have played out differently had James Williams not impressed the trial judge in *Altai* as a decent guy who did his best to repair the damage caused by his friend Claude Arney when he copied CA-Scheduler code into Oscar. Had Williams had come off as a "bad guy," the trial judge might have been more likely to follow *Whelan*, and copyright might have become the norm in protecting SSO, including interfaces, in the U.S. This would have been far more damaging to competition

[90] Developers of operating system programs, such as Microsoft's Windows and Vista, also benefit by the sheer complexity of their programs and the large number of interfaces they contain, which makes reverse engineering to discover them very difficult.

[91] The caselaw on the enforceability of anti-reverse engineering clauses is mixed.

[92] Synthetic biology, XML schemas, and computer languages are three other kinds of information innovations that also do not neatly fit into the patent and copyright bins. See Sapna Kumar & Arti Rai, *Synthetic Biology: The Intellectual Property Puzzle*, 83 Tex. L. Rev. 1745 (2007); Douglas E. Phillips, *XML Schemas and Computer Program Language Copyright: Filling in the Blanks in Blank Esperanto*, 9 J. Intell. Prop. L. 63 (2001).

and follow-on innovation than patents for interfaces because copyright protection attaches automatically by operation of law; patents, by contrast, can only be obtained by applying and subjecting one's claims to scrutiny by patent examiners, the creativity standard is far lower, and copyright protection lasts for about five times longer than patents. This would make no more sense that extending copyright protection to electrical plugs and sockets. So it was fortuitous indeed that Williams was a decent guy and the trial judge was willing to look more closely at what kind of program SSO he was being asked to protect by copyright law and how the protection being sought would affect the future of competition and follow-on innovation.