Copyright and the Challenge of the New

Editors

\begin{itemize}
  \item Brad Sherman
  \item Leanne Wiseman
\end{itemize}
CHAPTER 10
A Square Peg in a Round Hole? Copyright Protection for Computer Programs

Pamela Samuelson*

In the 1960s and 1970s, intellectual property professionals around the world started discussing what form (if any) of legal protection was or should be available for computer programs. Back then, it was far from obvious that a consensus would emerge that copyright was a suitable legal regime for protecting software. Early on, it was evident that trying to apply copyright law to computer programs would be, as one court later put it, rather like trying to fit a square peg into a round hole.¹ Nor was it obvious that less than fifty years later, annual software industry revenues in the US alone would exceed USD 260 billion.² And yet these non-obvious things happened.

This chapter explains why the economic and legal arguments for extending copyright protection to computer programs in the 1960s were initially perceived to be weak because copyright had never before protected such a deeply functional type of intellectual creation. It then recounts how the objections to copyrighting programs were overcome. By the late 1970s and early 1980s, as people began conceptualizing programs as 'literary works', software came to be seen as an appropriate subject matter for copyright protection. By the late 1980s and early 1990s, however, courts and commentators grew increasingly dissatisfied with the literary work metaphor because it obscured recognition of the functional nature of programs. Once program functionality was recognized, courts began according them a limited scope of copyright protection.

* I wish to thank Kathryn Hashimoto and Andrea Yankovsky for research assistance and Hank Barry and Bob Gushko for editorial suggestions.
Chapter 10: Computer Programs: A Square Peg in Copyright's Round Hole

induce their creators to produce these programs. Even developers of more general purpose application programs tended to sell their products to customers as complete solutions that included installation, training and maintenance services through which software development costs could be recouped. Based on these observations and concerns about some costs that copyright might impose on this industry, Breyer concluded that the economic case for copyrighting software was weak.

The legal case for extending copyright protection to computer programs was equally uneasy. Even if source code forms of programs, that is, texts written in computer programming languages, could be assimilated into copyright as literary works, machine-executable programs (often known as object code) could not for two principal reasons. One was that machine-executable programs are functional processes (that is, machines that just happen to be constructed in text). Under Baker v. Selden and its progeny, copyright protection had long been unavailable to functional processes or machines. Second, machine-executable programs are not intelligible to humans and hence were seemingly not "copies" of source code texts within the meaning of copyright law.

---

5. Ibid. at 281. The title says it all.
6. Ibid. at 344. The indifference of software developers to copyright was evident from the fact that only 200 programs had been registered with the US Copyright Office in the five years since the Office had begun accepting programs as registrable subject matter. Ibid. at 345. The decision of this Office to begin registering programs is discussed infra notes 17-19 and accompanying text.
7. Ibid. at 344.
8. Ibid. at 345.
9. Ibid. at 345.
10. Ibid.
§10.01

When the US Copyright Office undertook in 1964 an assessment of whether it should accept applications to register computer program copyrights, the utility and unintelligibility of programs gave rise to serious doubts within the Office about programs as copyright subject matter. 17 The Office ultimately decided to accept registration applications for programs anyway, although it did so under its so-called rule of doubt. That is, the Office would issue registration certificates for programs, but the certificates reflected the Office's doubts about whether programs were really copyrightable. 18 It left to registrants the challenge of defending the copyrightability of programs, should this later become necessary in litigation. The Office also made registration contingent upon the programmer's deposit of the full text of the source code form of the program. 19

During the legislative hearings on copyright law revision bills during the mid-1960s, serious concerns also emerged about copyright protection for computer programs. Among the strongest critics was Professor Arthur Miller, who recommended denying copyright to computer programs because programs were 'functional item[s] that were plainly distinguishable from 'books or plays or motion pictures or poetry - the forms of expression that traditionally have been covered by our copyright legislation'. 20 To extend copyright protection to programs would, he opined, effectively grant patent-like rights, without 'the safeguards and limitations that surround a patent grant.' 21 This could, he warned, 'very seriously stultify the programming art' in large part because computer programming 'is, by and large, a derivative art based on fairly well established and commonly used mathematical and logical principles.' 22 Miller doubted that the software industry would have arisen 'had there been copyright protection for programs in the past'. 23

The first US software copyright decision also conceived of object code as uncopyrightable subject matter. In explaining why executable forms of programs were not protectable by copyright law, the court analogized source code to architectural plans, both of which could be copyrighted, and analogous object code to a house built with copyrighted architectural plans, neither of which under then-existing US law qualified for copyright protection. 24

The unsuitability of copyright as a form of legal protection for software was also the subject of some commentary in the US legal literature as well as of legislative proposals in the international arena. One senior IBM attorney, for instance, proposed

18. Ibid.
19. Ibid.
21. Ibid. at 197, 199.
22. Ibid.
23. Ibid. at 199.

Chapter 10: Computer Programs: A Square Peg in Copyright's Round Hole §10.02

that a sui generis form of legal protection should be created for computer programs, which would protect the applied know-how embodied in programs from misappropriation. 25 The World Intellectual Property Organization (WIPO) even developed a model sui generis law to protect computer programs in the late 1970s. 26

Given the US Copyright Office's doubts about programs as copyright subject matter, the questions raised about programs in the legislative history of the 1976 Act and WIPO's endorsement of a sui generis proposal, one might have expected that sui generis legislation would have become the legal norm for protecting computer programs instead of copyright. So what happened to change the momentum in favour of sui generis legislation to one in favour of copyright? It is to that part of the story we now turn.

§10.02 FROM CONTU TO TRIPS: HOW COPYRIGHT BECAME AN INTERNATIONAL NORM FOR SOFTWARE PROTECTION

No thought whatever was given to the copyright implications of computer technologies in the 1950s when revision of US copyright law first commenced. 27 By the early 1960s, however, computer technology issues began to surface, including questions about whether computer programs were copyrightable. 28 Among the other computer-related issues that perplexed participants in the legislative deliberations on copyright reform was whether scanning of in-copyright works by libraries or others to store them in computers and/or to make indexes of their contents should be considered copyright infringement. 29 Less perplexing, but far more contentious, was the issue of whether libraries should be able to make photocopies of in-copyright works in their collections (e.g., articles from journals or chapters from books) for their patrons. Libraries and educational institutions asserted that these acts should be privileged, either by a specific exception or as a fair use, while publishers and certain author groups argued that photocopying constituted infringement. 30

27. The Copyright Office commissioned a series of studies on copyright reform topics in the 1950s. These studies are available on the Office's website at http://www.copyright.gov/history/studies.html.
28. A history of the Congressional proceedings on copyright revision, with special attention to the controversial new technology issues, can be found in Appendix A to NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1978) [hereinafter CONTU Report]. Unless otherwise footnoted, the historical assertions in this section of the chapter come from this Appendix.
29. These are, ironically enough, the very issues in litigation in the Authors Guild v. Google, Inc. lawsuit, which challenged Google's scanning and indexing the contents of copyrighted books for its Book Search Project. Roughly four decades after these questions were first posed in Congress, they are in litigation and remain hotly contested.
30. The history of the legislative debate over the photocopy issue is reviewed in detail, CONTU Report, supra note 28, at 89-96. This question was in litigation during the pendency of the copyright revision bills in Williams & Wilkins Co. v. United States, 487 F.2d 1345 (Ct. Cl. 1973),
By the late 1960s, these new technology issues were so contentious that they were holding up enactment of the omnibus copyright revision legislation. Eventually, a compromise was reached whereby the omnibus bill could move forward and a commission would be established to address the new technology controversies. In late 1974, Congress passed legislation to create a National Commission on New Technological Uses of Copyrighted Works (CONTU). 31

CONTU's specific legislative charter was to make recommendations for legislation or other measures in respect of these uses of copyrighted works:

(1) the reproduction and use of copyrighted works of authorship —
(A) in conjunction with automatic systems capable of storing, processing, retrieving and transferring information, and
(B) by various forms of machine reproduction, not including reproduction by or at the request of instructors for use in face-to-face teaching activities; and
(2) the creation of new works by the application or intervention of such automatic systems of machine reproduction.32

CONTU fulfilled its charter on the computer use issues by opining in its Final Report that inputting a copyrighted work into a computer was a copyright-significant reproduction of that work, that compilations of data in a computer database might be copyrightable if original, and that computer-generated works might sometimes be eligible for copyright protection.33 It fulfilled its charter on photocopying with a set of guidelines.34

Yet, CONTU is most famous — and justifiably so — for a recommendation that was not within its statutory charter, namely, that copyright protection should be available to computer programs as literary works.35 Ironically, Arthur Miller, who in 1967 had spoken so eloquently against copyrighting computer programs, was now the chair of the CONTU subcommittee that favoured copyright for programs. In stark contrast to Breyer's intensely empirical assessment of the case for (and against) copyright protection for computer programs,36 the CONTU Report did not consider the state of the computer software industry or cite economic evidence that copyright protection was actually needed to induce the development and distribution of programs.

The CONTU Report instead asserted that computer programs were already copyrightable insofar as they were original works of authorship fixed in a tangible medium of expression under the Copyright Act of 1976.37 It argued that copyright was

aff'd by an equally divided Court, 420 U.S. 376 (1975) (holding that a government library's photocopying of individual articles for medical researchers was fair use).


32. ibid., § 201.

33. CONTU Report, supra note 28, at 38–44.

34. ibid., at 47–78.

35. ibid. at 10–26.

36. Miller could not have been unaware of Breyer's article, as they were colleagues on the faculty of Harvard Law School and Miller himself had once voiced similar skepticism about copyright for programs.

37. CONTU Report, supra note 28, at 15–16. Miller seemingly had a contrary position about whether the pending bill would encompass software when he was testifying before Congress in 1967. See supra notes 20–23 and accompanying text.

Chapter 10: Computer Programs: A Square Peg in Copyright's Round Hole

superior to other existing forms of legal protection and ignored entirely the sui generis option then under consideration at WIPO.38 It also ignored the possibility that developers would assert patent and trade secret protections, as well as copyrights, in program innovations. It expressed confidence that courts would be able to distinguish between the protectable expression and the unprotected ideas and methods embodied in programs.39 In short, it saw no difficulties whatever in using copyright to protect programs.

Neither the CONTU Commissioners nor the CONTU staff had any computer programming expertise, which may explain why its Final Report contained misstatements about the nature of computer programs.40 CONTU was also overoptimistic about the ease with which copyright law could be applied to programs. None of the difficult scope of protection issues that bedevilled the courts in the 1980s and 1990s was anticipated in the CONTU Report.

Despite CONTU's professed confidence that Congress had already spoken in favour of copyrighting programs, the Commission recommended some amendments to US copyright law just to make sure. This included a definition of the term 'computer program' and an exception to allow lawful possessors to make copies of programs in the course of using them, to make backup copies and certain adaptations, and to resell the programs (as long as all copies of the program were transferred to the new owner).41 In the waning days of 1980, Congress adopted CONTU's recommended changes.42

This legislation happened at a strategically important moment in time. Although WIPO had proposed a model sui generis law for the legal protection of software in the late 1970s, no country had adopted it. By so clearly endorsing copyright as a form of legal protection for programs, the US Congress had not only clarified lingering doubts about copyrightability of programs domestically, but also set a precedent that the US could be expected to urge others to follow in the international arena.

The US had a strong interest in persuading other countries to follow the US lead on this matter because US firms dominated the increasingly global market for software.43 It did not take a genius to realize that without international consensus about and adoption of an appropriate form of legal protection for programs, the US lead might erode, perhaps substantially. A sui generis option for protecting programs, however conceptually elegant it might be, would lack a viable infrastructure for building an

38. ibid. at 16–18.

39. ibid. at 18–19.

40. See Samuelson, CONTU Revisited, supra note 14 (noting the lack of CONTU expertise about programs and detailing these misstatements). CONTU compared object code to sound recordings and that both could only be 'read' by machines. CONTU Report, supra note 28, at 15–16. However, processing a sound recording in a machine permits humans to listen to the protectable expression in the work; the same is not generally true of computer programs. ibid. at 29 (Hersey Dissent). Using the word 'read' to describe computer processing of object code is thus misleading.

41. CONTU Report, supra note 28, at 12–13. Congress changed the exception for transfers of copies of software so that only 'owners of copies' and not 'lawful possessors' of software were entitled to make such transfers.


43. Campbell-Kelly, supra note 3, at 22–23.
international consensus on software protection. A copyright treaty infrastructure was, however, already in place if a consensus could be built about protecting software through copyright law.44

The vulnerability of US software developers to foreignimitators was already becoming apparent in the late 1970s and early 1980s. IBM Corporation was, for example, engaged in bitter competition with non-US computer manufacturers such as Fujitsu and Hitachi. These Japanese firms developed clones of IBM systems software so they could sell IBM-compatible mainframe computers (at a lower price, of course, than IBM's computers). IBM sued Fujitsu over the former's claim that the latter had infringed IBM copyrights in its systems software, a dispute that eventually settled with Fujitsu's agreement to pay a substantial sum to IBM and with a complex arrangement for Fujitsu's continued development of compatible systems.45

US software developers had reason to be concerned that a sui generis regime, such as the one Japan was contemplating in the early 1980s,46 would provide a narrower scope and shorter duration of protection than copyright seemed to promise, and this might make it easier for foreign competitors to copy US software innovations. Japan was not the only nation that gave serious consideration to the sui generis option. The European Union did so as well,47 and had it opted for sui generis protection, the international conversation about the appropriate legal regime for computer programs might have evolved differently. In 1991, however, the EU adopted a directive endorsing copyright protection as a form of legal protection for programs, albeit with some special provisions to exclude program interfaces from the scope of copyright.48

Although academic interest in sui generis protection persisted into the 1990s,49 the international debate on this issue essentially ended with adoption of the Agreement on Trade-Related Aspects of Intellectual Property Rights (TRIPS) in 1994, under which Member States agreed to protect computer programs as literary works by copyright law.50

Chapter 10: Computer Programs: A Square Peg in Copyright's Round Hole

$10.03$ FROM LITERARY WORKS TO FUNCTIONAL WRITINGS: EVOLVING CONCEPTIONS OF COMPUTER PROGRAMS IN THE COPYRIGHT CASE LAW

In orndary discourse, the term 'literary works' typically conjures up novels, dramatic plays, poetry and the like; it may also encompass non-fiction works, for references abound to social science, legal and scientific literatures. Few people would, however, if asked, describe tables of numbers (e.g., pork belly prices on commodity exchanges) or catalogues of lawn mower parts as literary works. Yet, under US copyright law, these works and other distinctly non-literary texts fall within the statutory definition of this term. 'Literary works', as a matter of US copyright law, consist of 'works, other than audiovisual works, expressed in words, numbers or other verbal or numerical symbols or indicia, regardless of the nature of the material objects ... in which they are embodied'.51 A similarly broad definition is enshrined in a major international copyright treaty.52

Source code forms of programs are typically expressed in words, numbers and symbols, so they easily fall within this broad definition. Object code does not really 'express' anything, for this term connotes communicating to or among humans, which object code does not do. Conceptualizing object code as a literary work can only be accomplished by assuring ourselves that there is such a close correspondence between source and object code that literary character, if any, that exists in the former must have been carried over to the latter in the process of converting source code into object code.

Still, if a legislature enacts a law that requires treating object code as a literary work, one can expect courts will do their best to fulfill the legislative intent. This was not difficult in software copyright cases involving exact copying of object code. But there were at least four highly contentious issues litigated in the US in the 1980s and 1990s for which the copyright caselaw on literary works provided little or no useful guidance. Because CONTU provided no guidance on these issues, the courts had to fend for themselves.

One contentious issue was whether the structure, sequence and organization (SSO) of programs was protectable by copyright law, and if so, to what extent. A second was whether the 'look and feel' of programs (e.g., of their user interfaces or the sequence of observable program behaviours) was within the scope of protection provided by copyright. A third was whether internal interfaces of programs could be protected by copyright law insofar as their use was necessary for achieving interoperability with other programs. A fourth was whether making copies of object code in the course of reverse engineering to access to interface information was copyright infringement.

Courts in some early cases involving these four issues conferred on programs a broader scope of copyright protection than later cases deemed appropriate. Judges in


47. Commission Green Paper on Copyright and the Challenge of Technology – Copyright Issues Requiring Immediate Action, COM(88) 172 final (7 June 1988).


49. See, e.g., Samuelson et al., Manifesto, supra note 14.


these early cases seem to have been mislaid to some extent by the literary work metaphor. As the courts in the US came to conceptualize programs as functional writings, they realized programs should be eligible for only a ‘thin’ scope of copyright protection because program functionality, which necessarily pervades program design, is beyond the scope of protection that copyright law can afford to program developers.

Although it was an ‘easy’ case because it involved exact copying of object code, Apple Computer, Inc. v. Franklin Computer Corp. merits some discussion because it inadvertently set the stage for later misinterpretations of the proper scope of copyright protection for programs. Apple sued Franklin because Franklin installed exact copies of Apple’s operating systems (OS) programs in its computers. Franklin’s plan was to attract customers to its lower priced machines by advertising that its computers were compatible with the Apple II (that is, they could run programs developed for the Apple II platform).

One of Franklin’s defences was that CONTU had meant for copyright protection to be available only for applications programs. The court rejected that defence because the Apple OS programs fell within the statutory definition of computer program recommended by CONTU. A second defence was that the Apple OS programs were unprotectable by copyright law because they were functional processes. The court rejected this characterization, saying that CONTU and Congress had decided to treat programs as literary works. Franklin also asserted an idea/expression merger defence, that is, an argument that the only way it could make its computers functionally compatible with the Apple II was by installing exact copies of the Apple programs on its machines. However, compatibility was, in that court’s view, ‘a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged.’ Hence, Franklin’s merger defence also failed.

There is no question that the court in Franklin reached the right result. If Congress’ decision to protect computer programs with copyright law was to be respected, it had to mean that exact copying of object code would get defendants in trouble, particularly as in Franklin, where the copyst did not even try to reimplement the Apple II OS functionality in different code.

Yet the Franklin decision was troublesome in at least two respects. One was in its strident rejection of compatibility as a possible justification for some copying from an existing program, dicta which took almost a decade to overcome. The other was in its narrow conception of §102(b) of US copyright law, which excludes from the scope of copyright protection ‘any idea, procedure, process, system, method of operation, concept, principle, or discovery’ that may be embodied in a copyrighted work. Although the court was right that this provision should not be interpreted to allow the

53. 714 F.2d 1240 (3d Cir. 1983).
54. Ibid. at 1247–52. Franklin’s argument was credible given that CONTU only discussed applications programs.
55. Ibid. at 1249.
56. Ibid. at 1253.
57. 17 U.S.C. §102(b)(2006). This provision codifies the principal holding of Baxter v. Selden, 101 U.S. 99 (1880), discussed supra note 15 and accompanying text, in which the Supreme Court held that

Chapter 10: Computer Programs: A Square Peg in Copyright’s Round Hole

exact copying of object code because this code is a ‘process’, it ignored the legislative history indicating that Congress had included this provision in the 1976 Act to ensure that the scope of copyright protection in computer programs would not be too broad (e.g., by excluding methods and processes). The Franklin decision regarded abstract ideas as the only type of unprotectable element in copyrighted programs, even though computer programs, unlike conventional literary works, typically implement functional procedures, processes, systems and/or methods of operation.

This misinterpretation of §102(b) was unfortunately given further reinforcement in Whelan Associates, Inc. v. Jaslow Dental Lab., Inc. Rand Jaslow had commissioned Whelan to develop a program to automate common business processes of his and other dental laboratories. The two intended to exploit this program as partners, but after a falling out, Jaslow decided to develop a program to do the same functions for IBM-PCs. Although his program and Whelan’s were written in different programming languages and used different algorithms, the overall structure of the two programs was similar, as were some data and file structures, and the two programs performed some functions in the same way. Based on these three types of similarities, the trial and appellate courts held that Jaslow infringed Whelan’s copyright.

Whelan noted that CONTU had characterized computer programs as literary works, and reasoned that because copyright law had long protected non-literal elements, such as the SSO of other types of literary works, such as novels and plays, copyright should protect the SSO of programs as well. Anyone was free to copy the ideas from existing programs, just as with novels and plays, but program SSO, Whelan opined, was protectable as long as there was more than one way to structure a program to achieve the program’s functions. Because Jaslow could have used different file, data and overall structures than Whelan, his use of the same or similar ones showed he was an infringer. The court also reasoned that 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, there would be too little protection to provide proper incentives to invest in developing computer programs.

Whelan also endorsed extending copyright protection to the ‘look and feel’ of programs, as there were similarities in the sequence of operations and screen displays for five subroutines of the Whelan and Jaslow programs. This seemingly extended copyright protection to the manner in which programs behaved (i.e., how they performed their functions). The court did not realize that these similarities may have been due to the fact that both programs implemented the same business processes and procedures, which §102(b) excludes from copyright protection.

Shortly after Whelan endorsed copyright protection for look and feel, Apple sued Microsoft for copying the look and feel of the Macintosh graphical user interface (GUI) in its new version of the Windows OS. In addition, Lotus Development Corporation
sued Paperback Software for copying the look and feel of its popular spreadsheet program. Apple eventually lost its case against Microsoft, largely because the courts recognized that much of the look and feel of the Apple program was too functional to be protectable by copyright law. Although Lotus won its lawsuit against Paperback for cloning the look and feel of Lotus 1-2-3, its later very similar lawsuit against Borland International collapsed after the appellate court ruled that the command hierarchy of the Lotus spreadsheet program, on which the look and feel claim had been predicated, was outside the scope of copyright protection under §102(b).

Look and feel thereafter ceased to be a viable theory of software copyright infringement in the U.S. (except when the case involved copying of fanciful audiovisual elements of videogames). Copyright law requires software developers to write their own programs, but it is fair game to implement the same functionality as another program because program behaviour is outside the scope of protection that copyright law should be understood to provide to programs.

The most significant and influential case in U.S. software copyright history, though, has been Computer Associates Int'l, Inc. v. Altai, Inc. The three most important aspects of the Altai decision were these: first, it rejected the plaintiff's claim that copyright protection extended to program interfaces, which are part of program SSO; second, it criticized Whelan and its overly broad conception of the scope of copyright protection in programs; and third, it recognized the functionality of programs and devised a test for judging infringement in software copyright cases that was more consistent with traditional principles of copyright law than the approach taken in Whelan.

But first a few facts: Computer Associates (CA) and Altai were competitors in the market for scheduling programs designed to run on IBM mainframe computers. CA sued Altai for copyright infringement after learning that Altai's Zeke program contained code misappropriated from its CA-Scheduler program. Altai was apparently unaware that one of CA's former employees whom it had hired directly copied portions of CA's code in developing the compatibility modules of Zeke. Altai purged the tainted code from Zeke and assigned a clean-room team of fresh programmers to reimplement the compatibility components of Zeke in new non-infringing code.

Altai accepted liability as to the code copied from CA-Scheduler, but asserted that the rewrite of Zeke had purged the infringement. Relying heavily on Whelan, CA argued that there were still substantial similarities in SSO in the compatibility components of Zeke and CA-Scheduler, especially as to their parameter lists (i.e., lists of information that needed to be sent and received by subroutines of the affected programs).


65. Computer Associates Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706–11 (2d Cir. 1993). Although Altai did not expressly address whether application program interfaces (APIs) are protectable by copyright law, it suggested that they were not. An important recent ruling has specifically addressed this question, ruling that APIs are unprotectable, relying in part on Altai. See Oracle Am., Inc. v. Google Inc., No. C-10-03561 WHA, 2012 U.S. Dist. LEXIS 75890 (N.D. Cal. 31 May 2012) (ruling that Android’s use of 37 Java APIs did not infringe Oracle’s copyrights).


67. Altai, 982 F.2d at 710.

68. Ibid. at 711.
§10.03 Pamela Samuelson

It regarded CA’s economic argument for broad software copyright protection to be inconsistent with a recent Supreme Court decision.69 Adopting CA’s theory would, the court added, ‘have a corrosive effect on certain fundamental tenets of copyright doctrine’.70 Trying to apply copyright to computer programs was, it believed, like trying to fit a square peg in a round hole. Yet, however ill-suited copyright law might be for protecting program innovations, Congress was the proper venue in which to argue that programs needed additional legal protection. The court in Altat also suggested that patents might be a more suitable form of protection for program innovations than copyright.71

The Altat decision may not initially have induced software developers and their lawyers to start patenting interfaces and other program SSO, in part because it took some years for Altat to defeat Whelan in the subsequent caselaw and emerge as the leading decision for judging claims of software copyright infringement.72 However, the patent option became more salient after the Ninth Circuit Court of Appeals issued its ruling in Sega Enterprises, Ltd. v. Accolade, Inc.73 less than a month after the Second Circuit’s Altat ruling.

Sega was important for at least four reasons. For one thing, it embraced Altat’s conceptualization of computer programs as utilitarian works eligible for only a thin scope of copyright protection.74 Second, Sega followed Altat 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’.75 Third, the court ruled that copying program code in the course of reverse engineering for a legitimate purpose, such as extracting interface information to make a compatible program, did not infringe any copyright in that code. 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."

Fourth, it indicated that even copying some exact code from another program was not infringement insofar as that code was essential to achieving interoperability.77


70. Altat, 982 F.2d at 717.

71. Ibid.

72. Altat has been followed in more than 70 subsequent cases in the U.S.

73. 977 F.2d 1510 (9th Cir. 1992).

74. Ibid. at 1526 (‘Under the Copyright Act, if a work is largely functional, it receives only weak protection.’).

75. Ibid. at 1525–26.

76. Ibid. at 1525.

77. Sega had programmed its system so that games would not be executable on the Sega platform unless a small segment of code was reproduced exactly. Accolade had included this code in its

§10.04

Chapter 10: Computer Programs: A Square Peg in Copyright’s Round Hole

Because Sega allowed unlicensed reverse engineering of code to extract unprotected elements of programs,78 it seemed to imperil developer efforts to protect interfaces and other parts of program SSO as trade secrets. Sega signaled that the only reliable means for protecting the functional designs embodied in programs was by patenting them. Since the mid-1990s, there has, in fact, been a dramatic surge in the patenting of software innovations.79 To sum up, it took nearly a decade and a half for courts in the US to figure out how to apply copyright to computer programs in a sensible manner, both in terms of consistency with traditional copyright doctrines and in terms of interpretations that would promote innovation and competition in the software industry. The many precedents that had applied copyright law to novels and plays offered very little guidance about how to think about protecting program interfaces, other program SSO and program behaviour. Courts consequently ended up developing a sui generis form of protection for programs within the body of copyright law.

There has been far less litigation on software copyright issues in other countries. One consequence is that there are fewer judicial precedents elsewhere, and so the scope of copyright in software in other countries is less clearly defined than in the US. In general, however, the issues that proved controversial in the US cases have been resolved in a similar manner in other jurisdictions.80

§10.04 IS COPYRIGHT RESPONSIBLE FOR THE SOFTWARE INDUSTRY’S SUCCESS?

In 1980, the year that Congress enacted legislation to implement CONTU’s recommendations on copyright protection for computer programs, the US software industry’s annual revenues were already close to USD 15 billion.83 Only a decade earlier, annual program so that its games could be played on the Sega platform. The court ruled that exact copying of this part of Sega’s code was fair use because it was necessary to interoperability.

78. 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 Gropman, Decompliation and Disassembly: Undoing Software Protection, COMPUTER LAW., February 1984, at 1.


80. See, e.g., Jonathan Band & Masanobu Raito, Interfaces on TRAD 2.0 135–82 (2011). But see Data Access Corp. v. Powerflex Services Pty., Ltd. [1996] 63 FCR 336 (Aust.), discussed in Band & Raito, supra, at 148–51 (holding copying of program SSO was infringing). An important recent case in the EU is SAS Institute, Inc. v. World Programming Ltd., [2010] EWHC 1829 (Ch.), challenging a competitor’s emulation of the SSO of the behaviour of SAS’s popular statistical analysis program. The English High Court held that WPL’s use of the same programming language, interfaces and program functionality did not infringe copyright. Yet this court referred several questions to the Court of Justice of the European Union (CJEU) for its interpretation of the EU Software Directive. In May 2012, the CJEU issued an opinion substantially agreeing with the English High Court’s decision on these issues. See Case C-406/10, SAS Inst Inc v. World Programming Ltd, 2012 EUR-Lex C/ELJ No. 62010CJ0406 (2 May 2012).

81. Campbell-Kelly, supra note 3, at 18–19.

264

265
revenues had been just under USD 2 billion. At least three factors contributed to the sevenfold-plus growth in that decade. One was that IBM, which in the late 1960s had a 70% share of the market for mainframe computers, decided in 1970, under pressure from antitrust authorities, to stop bundling applications programs with its mainframe computers. This opened up a host of new opportunities for independent firms to develop software for IBM mainframe computers. A second was that in the latter part of that decade the personal computer (PC) was born, and with it came new opportunities to sell applications programs to a larger array of customers. A third was that a high demand existed for software services, which then accounted for 60% of overall software industry revenues. There is, however, no reason to believe that copyright protection was a factor in the industry’s growth during this period.

As impressive as was the software industry’s growth between 1970 and 1980, a much more substantial burst of growth lay ahead. By 1990, annual US software industry revenues exceeded USD 100 billion. The single most important factor contributing to this phenomenal growth was that the PC market exploded in the 1980s. Between 1981 and 1990, annual shipments of PCs in the US rose from 780,000 to 9.3 million units, and the size of the installed base of PCs went from 1.74 million to 54.8 million in the US. In 1984 alone, IBM sold USD 4 billion worth of PCs. The openness of the IBM PC architecture meant that other computer companies could make and sell IBM-compatible PCs, as many of them did. The success of this PC platform created high interest among independent software firms in developing applications for it, and the availability of these applications helped to drive further demand for this platform.

During the 1980s, IBM was not only a very successful manufacturer of computers, but also the leading US producer of software. In fact, about 10% of the USD 100 billion annual US software industry revenues in 1990 went to IBM. Yet, despite its prowess in making computers and software, IBM made at least one strategic decision in the 1980s that it probably came to regret, namely, choosing Microsoft’s MS-DOS as the systems software for its PC. Every time IBM sold a PC, Microsoft got a sale as well; moreover, the same was true with every sale of IBM-compatible PCs, for Microsoft’s systems software was installed on them too. Microsoft also became the leading provider of applications programs capable of running on its own OS. It was in 1990 that

Microsoft’s annual revenues first topped USD 1 billion, about 40% of which was from the sale of systems software and nearly 50% from applications programs. Microsoft was, of course, far from the only highly successful developer of applications programs in the 1980s. Among the early very successful application programs for IBM-compatible PCs was the Lotus 1-2-3 spreadsheet program. In 1983, the year of the launch of this program, Lotus Development Corporation earned USD 53 million from sale of this software; by 1990, its annual revenues had grown to USD 566 million. Among the other leading developers of software applications in the 1980s were WordPerfect, Novell, Borland, Autodesk, Adobe and Symantec, many of whom also developed applications for the IBM PC platform.

These developers of mass-market software cared about copyright protection quite a lot. This was especially true for applications developers who tended to sell prepackaged software without installation, maintenance, or other services, at relatively modest prices in the hopes of attracting owners of PCs as customers. But it was also true for companies like Apple who sold off competitors like Franklin by asserting copyright protection in its systems software programs. After decisions such as Apple v. Franklin, no legitimate claim could be made that a competitor was entitled to sell exact copies of another company’s software. However, counterfeiting became a problem for makers of mass-market software. Software firms also worried about infringements by prospective customers who often preferred to make unauthorized copies of software borrowed from friends rather than buying their own copy. Sales of popular software could also be eroded by competitors who cloned the functionality of those programs by reimplementing it in different program code, as Paperback and Borland did to Lotus 1-2-3 in the late 1980s and early 1990s.

As important as copyright was to the mass-market sector of the software industry, the exclusive rights of copyright were not necessary to induce investment in all sectors of the software industry in the 1980s. Systems software development costs could generally be recouped, as before, through the sale of hardware. Copyright may not have been necessary to induce investment in other sectors of the software industry, such as software services, custom-developed software, internal development tools and amateur, nonprofit researcher, and government developed software. Yet, copyright was available to all software developers regardless of need.

The 1990s brought additional growth to the US software industry, even though it was not as explosive as before. Over the course of that decade, user expenditures on systems software rose from USD 16.4 billion to USD 41.7 billion, on applications software products from USD 17.3 billion to USD 63 billion, and on programming services from USD 10.4 billion to USD 33.4 billion. For most of that decade, IBM

82. Ibid.
83. Ibid. at 109–10. The IBM unbundling decision was ‘a turning point for the industry’. Ibid. at 6.
84. Ibid. at 89–91.
85. Ibid. at 203–03.
86. Ibid. at 18–19. In 1970, services accounted for about 75% of software industry revenues. Ibid.
87. The irrelevance of copyright in the 1970s was evident from the fact that relatively few programs were registered with the US Copyright Office before 1980 and by the doubts that continued to exist in the 1970s about whether copyright protection was really available for programs See supra notes 6, 17–19 and accompanying texts.
88. Campbell-Kelly, supra note 3, at 18–19.
89. Ibid. at 238. And in 1992, there were more than 63 million installed base PCs in the US. Ibid.
90. Band & Katoh, supra note 80, at 30.
91. Campbell-Kelly, supra note 3, at 237–42.
92. Ibid. at 174–75.
93. Ibid.
94. Ibid. at 237–42.
remained the leading producer of software; in 1998, however, Microsoft took over that crown.\textsuperscript{102} By 2000, Microsoft’s annual revenues were almost USD 23 billion, more than twenty times its 1990 revenues.\textsuperscript{103} Computer Associates became the second largest of the independent software companies in the 1990s, with close to USD 6.8 billion in revenues in 2000 from corporate customers.\textsuperscript{104} Pre-packaged mass-market software sold to individual consumers also did well in this period. Intuit, for example, which introduced Quicken in the market in 1989, saw its revenues climb above the USD 100 million mark in 1993 and the USD 1 billion mark in 2000. The videogame industry also became a multi-billion-dollar industry in this period as well.\textsuperscript{105}

An important development in the 1990s was a substantial growth of software companies specializing in enterprise software, often specially tailored to the needs of particular businesses (e.g., financial services). Companies such as Oracle did well in the 1980s, but soared in the 1990s. Oracle alone had more than USD 10 billion in annual revenues in 2000.\textsuperscript{106} A company like Microsoft might sell 10 million copies of software at USD 200 per unit, but Oracle could make the same revenues by selling its database software to 10,000 customers at USD 200,000 a pop. Although US firms were dominant in most software industry sectors, the German company SAP emerged as one of the leaders in the enterprise software market, with revenues in 2000 of more than 12 billion euros.\textsuperscript{107}

Enterprise software providers typically offered their customers long-term contracts that included services and customization, which had the advantage of keeping revenues flowing into software firms coffers.\textsuperscript{108} The makers of pre-packaged software, by contrast, had to chase new opportunities for one-off sales to customers which meant that they sometimes faced dangerous drops in revenues if recessionary pressures lowered demand for their products.\textsuperscript{109}

The first decade of the twenty-first century has witnessed continued growth in the software industry. One US-based trade industry group estimated that the US software industry in 2007 contributed more than USD 261 billion to the US gross domestic product.\textsuperscript{110} Another industry study estimated the US software industry revenues for 2010 at about USD 150 billion.\textsuperscript{111} Thirty-six per cent of these revenues were attributable to the sale of systems software, 33.6% to applications programs, and the rest to various kinds of software services, including custom development.\textsuperscript{112}

Copyright remains important in the pre-packaged mass-market applications market. While this chapter has focused heavily on developments in the US, it is important to acknowledge that the global software industry has also grown substantially since

\textsuperscript{102} Ibid. at 232.
\textsuperscript{103} Ibid. at 233.
\textsuperscript{104} Ibid. at 176–80.
\textsuperscript{105} Ibid. at 284–88.
\textsuperscript{106} Ibid. at 186.
\textsuperscript{107} Ibid. at 192.
\textsuperscript{109} Ibid. at 29.
\textsuperscript{110} See BSA Report, supra note 2.
\textsuperscript{111} CASEY THORMAHL, IBIS WORLD INDUSTRY REPORT: SOFTWARE PUBLISHING IN THE U.S. 5 (2010).
\textsuperscript{112} Ibid. at 14.

1980. Global software industry revenues were just under USD 15 billion in the early 1980s, when the US had 70% share of this market.\textsuperscript{113} By 1990, global software industry revenues reached USD 110 billion, of which the US share was 57%.\textsuperscript{114} One recent study has estimated that global revenues from the software industry will rise to USD 330 billion by 2014, of which the US share is likely to be just under 50%.\textsuperscript{115}

As impressive as this pattern of growth certainly is, one cannot blithely assume that copyright law deserves all or substantially all of the credit for the high levels of investment in software over the past several decades. For one thing, 70% of the total US investment in software development, both currently and historically, has either been for custom-developed software or software developed by firms for their own internal use.\textsuperscript{116} These types of software would probably have been developed even if copyright protection had been unavailable.

Of the remaining 30% of software industry revenues, a substantial share derives from systems software.\textsuperscript{117} As noted earlier, the costs of developing this type of software are typically recouped through the sale of computers on which the systems software has been installed.\textsuperscript{118} The overwhelming majority of applications programs are sold to businesses,\textsuperscript{119} and often these applications come with long-term contracts to provide installation, system integration and other support services, as well as updates.\textsuperscript{120}

In the first decade of the twenty-first century, software is increasingly being made available 'in the cloud' as a service rather than being sold in packages that users download from the Internet or buy in the store.\textsuperscript{121} It is not clear that copyright protection is necessary to recoup the costs of this software, because people pay for access to this software, not for copies of it.

\textsuperscript{113} Campbell-Kelly, supra note 3, at 23. Much of the roughly USD 15 billion in software industry revenues reported supra note 81 and accompanying text included revenues from programming services.
\textsuperscript{114} Ibid. By 1990, Japanese firms had 13% of the global software market, while French, German and UK producers combined had 21% of the market, and 9% came from other nations.
\textsuperscript{115} Datamonitor, SOFTWARE GLOBAL INDUSTRY GUIDE (2010). See also BSA Report, supra note 2 (reporting that in 2008, the US software firms generated USD 136.6 billion in revenues globally from the sale of packaged software, representing a 45.9% share of that market).
\textsuperscript{116} Bureau of Economic Analysis, National Economics Account, Data Tables, Software Investment and Prices, by Type, Table 11, last updated on 4 August 2010, available at http://www.bea.gov/national/info_comp_tech.htm. These data show that in 2009, only 30% of the total software expenditures were for the development of prepackaged software; 33% were for custom-developed software and 37% for the development of software for internal use. Ibid. Custom and internal use development of software had also accounted for a very substantial portion of applications development in 1970 when Breyer assessed the need for copyright protection for programs and found it wanting. Breyer, supra note 4, at 345.
\textsuperscript{117} Thormahlen, supra note 111, at 14.
\textsuperscript{118} Justice Breyer noted this many years ago in his economic assessment of the case for copyrighting programs. See Breyer, supra note 4, at 344.
\textsuperscript{119} Thormahlen, supra note 111, at 16.
\textsuperscript{120} Cusamano, supra note 108, at 63–67.
\textsuperscript{121} A more extended discussion of these and other developments in the software industry as they impact the case for copyright for computer programs today can be found in Samuelson, Uneasy Case Revisited, supra note 13, Part IV.
Moreover, some of the most commercially valuable software developed in the past decade or so — search engines such as Google, social networking software such as Facebook and other Web 2.0 products and services — have found alternative ways to make money. If copyright is important to these firms, it is probably not because they need it to recoup their investments in developing this software.

Open source software has become a significant sector of this industry in the modern era. Developers of this software may have little need for copyright, except as a legal hook upon which to hang licence terms that support the openness preferences of their developers.

The segment of the software market for which copyright protection continues to matter the most is the market for pre-packaged general purpose application programs sold off the shelf (e.g., in computer stores or online). These programs are expensive to develop and cheap to copy, and because they are generally sold without other assets or services through which the costs of development might be recouped, their developers do rely on copyright, especially for protection against exact copying of code. Yet it should be noted that these types of application programs are often subject to mass-market licence restrictions and technical protection measures that inhibit software infringements. Still, it is fair to say that copyright infringement is nevertheless an ongoing source of concern for this part of the software industry.

### §10.05 CONCLUDING THOUGHTS

As early as the mid-1960s, it was apparent that computer programs were something of a square peg that would not fit comfortably in the round hole that copyright provided. This was mainly due to the functionality of machine-executable code and the fact that this code is generally not readable by humans, as traditional types of copyrighted works have been. Objections to copyrighting programs were, however, overcome by the rhetorical move of characterizing them as 'literary works' and assimilating them to the copyright regime as such.

When it became apparent that the many decades of precedents involving novels, plays and other conventional literary works provided scant guidance about how to apply copyright law to computer programs (except when the defendants made identical copies of protected works), courts had no choice but to fashion some sui generis protections for programs as functional writings that were eligible for only a thin scope of protection from copyright law.

122. Breyer anticipated that this kind of development might happen and asserted that the case for copyrighting programs would become easier if it did. Breyer, supra note 4, at 347.


126. Ibid.