An Analysis of the Scope of Copyright Protection for Application Programs

Peter S. Menell*

* Associate Professor of Law, Georgetown University Law Center. S.B., Massachusetts Institute of Technology, 1980; J.D., Harvard Law School, 1986; Ph.D., Stanford University, 1986. I am grateful to Doug Baird, John Barton, Tim Brennan, Jeff Cunard, Joe Farrell, Paul Goldstein, Gillian Hadfield, Dennis Karjala, Ken Leffler, Steven Lundberg, Steve Salop, Warren Schwartz, and participants in seminars at the University of Chicago Law School, Georgetown University Law Center, Stanford Law School, and Yale Law School for comments on an earlier draft. I also owe thanks to Renee Brooker and Peggy Kelsey for excellent research assistance. This research was supported in part by a grant from the John M. Olin Foundation to Stanford Law School. I am solely responsible for the shortcomings of this work.

Computer software, by its very nature as written work intended to serve utilitarian purposes, defies easy categorization within our intellectual property system. The copyright law has traditionally served as the principal source of legal protection for original literary work, while the patent system and trade secret law have been the primary means for protecting novel utilitarian works. Faced with the difficult challenge of fitting computer and other new information technologies under the existing umbrella of intellectual property protection, Congress in 1974 established the National Commission on New Technological Uses of Copyrighted Works (CONTU) to study the implications of the new technologies and recommend revisions to federal intellectual property law.[1] After conducting extensive hearings and receiving expert reports, a majority of the blue-ribbon panel of copyright authorities and interest group representatives comprising CONTU concluded that the intellectual work embodied in computer software should be protected princi-

1. Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201, 88 Stat. 1873. CONTU's mandate was to study and make recommendations concerning the implications of photocopying and computer technologies for the authorship, distribution, and use of copyrighted works. *See generally* NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1979) [hereinafter CONTU REPORT].

pally under copyright law,[2] notwithstanding the fundamental principle that copyright cannot protect "any idea, procedure, process, system, method of operation, concept, principle, or discovery."[3] Congress implemented CONTU's recommendation in its 1980 amendments to federal copyright law.[4]

In light of the computer software industry's relative youth and anticipated rapid growth,[5] CONTU's rough empirical judgment that copyright would best promote the invention, development, and diffusion of new and better software products was, by necessity, highly speculative. As CONTU recognized, it was impossible in 1978 to establish a precise line between copyrightable expression of computer programs and the uncopyrightable processes that they implement.[6] Yet the location of this line—referred to as the idea/expression distinction[7]—was critical to the rough and highly speculative cost-benefit analysis that guided CONTU's recommendation. Drawing the line too liberally in favor of copyright protection would bestow strong monopolies over specific applications upon the first to write programs performing those applications and would thereby inhibit other creators from developing

2. *See* CONTU REPORT, *supra* note 1, at 1. *But see id.* at 27-37 (Commissioner Hersey, dissenting) (arguing that "forcible wrenching" would be required to protect computer programs under the copyright law); *id.* at 37-38 (Commissioner Karpatkin, dissenting) (same); *cf. id.* at 26-27 (Commissioner Nimmer, concurring) (warning that CONTU recommendations might take copyright law "beyond the breaking point," converting it into a general misappropriation law). *See generally* Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form,* 1984 DUKE L.J. 663.

3. Copyright Act of 1976, 17 U.S.C. § 102(b) (1982).

4. Act of Dec. 12, 1980, Pub. L. No. 96-517, 94 Stat. 3007, 3028 (codified at 17 U.S.C. §§ 101, 117).

5. Just in the decade since CONTU issued its recommendations, revenues in the computer software industry have grown from $10 billion in 1978 to a projected $53.6 billion for 1988. *See* COMPUTER & BUSINESS EQUIPMENT MANUFACTURERS ASSOCIATION, THE COMPUTER BUSINESS EQUIPMENT, SOFTWARE, AND TELECOMMUNICATIONS INDUSTRY 1960-95, at 9 (1986) [hereinafter CBEMA DATA] (all figures in 1986 dollars).

6. *See* CONTU REPORT, *supra* note 1, at 22-23. Yet CONTU was aware that difficulties in drawing this line might lie ahead: "Most infringements, at least in the immediate future, are likely to involve simple copying. In the event future technology . . . permits future infringers to use an author's program without copying, difficult questions will arise." *Id.* at 22.

7. In order to prevent copyright from occupying the proper domain of patent law, the courts have limited copyright protection to the *expression* of an idea and not the underlying idea itself. The leading case establishing this principle is Baker v. Selden, 101 U.S. 99 (1879). In *Baker,* the plaintiff, Selden, owned a copyright in a book describing a new accounting system. The book contained blank forms for using the system. Baker subsequently published books that described a similar bookkeeping system. The Supreme Court held that Selden's copyright extended only to his expression of the system and not to the system itself. Therefore, the copyright did not extend to the blank forms which contained no expression beyond the basic organization of the accounting system. In the Court's view, "[t]o give the author of the book an exclusive property in the 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 copyright." *Id.* at 102. This principle is codified in section 102(b) of the Copyright Act of 1976. H.R. Rep. No. 1476, 94th Cong., 2d Sess. (1976), *reprinted in* 1976 U.S. CODE CONG. & ADMIN. NEWS 5659, 5667. Courts applying this principle have withheld copyright protection from systems of shorthand and rules for games. *See generally* 1 PAUL GOLDSTEIN, THE TOPOGRAPHY OF COPYRIGHT § 2.15 (1989) (forthcoming).

improved products. Drawing the line too conservatively would allow programmers' efforts to be copied easily, thus discouraging the creation of all but modest incremental advances. The wisdom of Congress's decision to bring computer programs within the scope of copyright law thus depends critically upon where the courts draw this line.[8]

The first generation of copyright infringement suits under the 1980 amendments focused on whether and to what extent literal copying of computer software violates copyright law. These cases generally interpreted the copyright law as prohibiting direct copying of all forms of computer software, whether written in human- or machine-readable form, and whether designed to perform specific data processing tasks for the user (application programs) or to manage the internal functions of computers (operating systems).[9] Because the coding of operating systems is critically important to hardware system compatibility, the principal economic effect of the first generation of cases was on competition among hardware manufacturers.[10]

A second generation of copyright infringement suits is now beginning to emerge, focusing on the extent to which nonliteral forms of copying constitute copyright infringement. In a leading case of this second generation, *Whelan Associates v. Jaslow Dental Laboratory*,[11] the Third Circuit held that copyright protection extends to the overall structure, sequence, and organization of an application program.[12] More recent cases have extended this doctrine to video screen displays, finding that copyright protection encompasses the "overall structure, sequence and arrangement of the screens, text, and artwork (i.e., the audiovisual display in general)."[13] Whereas the first generation of cases was most critical for the hardware sector of the computer industry, the second generation looms large for the software sector, particu-

8. *See* CONTU REPORT, *supra* note 1, at 22-23 (noting that "[s]hould a line need to be drawn to exclude certain manifestations of programs from copyright, that line should be drawn on a case-by-case basis by the institution designed to make fine distinctions—the federal judiciary").

9. *See, e.g.*, Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1251 (3d Cir. 1983), *cert. dismissed*, 464 U.S. 1033 (1984); Stern Elecs., Inc. v. Kaufman, 669 F.2d 852, 856-57 (2d Cir. 1982); Apple Computer, Inc. v. Formula Int'l, Inc., 562 F. Supp. 775, 781-82 (C.D. Cal. 1983), *aff'd*, 725 F.2d 521, 524-25 (9th Cir. 1984). Commentators have questioned the correctness of this line of cases. *See, e.g.*, Paul Goldstein, *Infringement of Copyright in Computer Programs*, 47 U. PITT. L. REV. 1119, 1126-27 (1986); Dennis S. Karjala, *Copyright, Computer Software, and the New Protectionism*, 28 JURIMETRICS J. 33, 64-67 (1987); Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329, 1362-63 (1987); Samuelson, *supra* note 2, at 742-49.

10. It is notable that CONTU did not recognize the potentially adverse effects of copyright protection for computer software on the adoption and improvement of compatible operating systems. *See* Menell, *supra* note 9, at 1330.

11. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 107 S. Ct. 877 (1987).

12. *Id.* at 1224-25.

13. *See* Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1135 (N.D. Cal. 1986); *see also* Digital Communications v. Softklone Distrib. Corp., 659 F. Supp. 449, 457-59 (N.D. Ga. 1987).

larly with regard to the coding of application programs and the design of computer-human interfaces.

Encouraged by expansive decisions in this second wave of litigation, two leading computer firms have shaken the U.S. computer industry recently by filing copyright infringement suits against their competitors. In January 1987, Lotus Development Corporation, a leading maker of "spreadsheet" application programs,[14] filed suit against Mosaic Software, Inc. and Paperback Software International, alleging that Mosaic's Twin™ and Paperback's VP-Planner™ infringe Lotus's popular 1-2-3™ program.[15] In March 1988, Apple Computer, Inc. brought an action against Microsoft Corp. and Hewlett-Packard Co. for selling software products that allegedly infringe the video displays of Apple's Macintosh™ computers.[16]

These suits require courts to draw the elusive line between copyrightable expression in an application program and an uncopyrightable "idea, procedure, process, system, method of operation, concept, principle, or discovery"[17] used to implement such a program. Drawing upon the rapidly developing science of application programming and key features of the market for application programs—in particular, the "public goods" nature of the intellectual work embodied in application programs and the importance of compatible user interfaces—this article critically analyzes the application of copyright law's idea/expression merger doctrine[18] in early application program cases and proposes an alternative approach that better promotes the invention, development, and diffusion of improved application programs.

The article describes how the first courts to address the scope of copyright protection for application programs have tended to view application programming as predominantly an exercise in creative expression and accordingly have interpreted the scope of copyright protection in this area quite broadly. As Part I explains, however, many design choices in writing application programs are made by applying the principles of the scientific fields of human factor analysis and software engineering. Thus, the tendency of courts to view application

14. A "spreadsheet" application program is an "electronic worksheet," useful for business calculations, that enables the user to generate quickly a new table of financial, production, accounting, or statistical information merely by changing any entry in the table or by modifying the assumptions that underlie the relationships between rows or columns of the table.

15. Lotus Dev. Corp. v. Paperback Software Int'l, No. 87-0074k (D. Mass. filed Jan. 1987); Lotus Dev. Corp. v. Mosaic Software Inc., No. 87-0076k (D. Mass. filed Jan. 1987); *see also* David Churbuck & Beth Freedman, *Suits Against 1-2-3 Imitators May Have Wide User Impact,* PC WEEK, Jan. 20, 1987, at 125, 130; Jerry Schwartz, *Software Industry Stalled by Suits,* Nat'l L.J., June 29, 1987, at 3, col. 1.

16. Apple Computer, Inc. v. Microsoft Corp., No. 88-20149 (N.D. Cal. filed Mar. 17, 1988); *see also* Brenton Schlender, Michael W. Miller & Paul B. Carroll, *Apple's Copyright Lawsuit Is Seen as Effort to Lock in Technical Lead,* Wall St. J., Mar. 21, 1988, at 21, col. 4.

17. 17 U.S.C. § 102(b).

18. *See* note 7 *supra* and text accompanying notes 154-156 *infra* (describing idea/expression merger doctrine).

programming as more akin to literary creativity than to scientific and engineering advancement threatens to give broad legal protection to basic principles of human factor analysis and software engineering without requiring the creators of the programs embodying those principles to satisfy the more exacting standards of patent law.[19] The article concludes that a careful application of the idea/expression merger doctrine, recognizing the importance of scientific considerations in application programming and the need to standardize computer-human interfaces, would both foster the invention, development, and diffusion of improved application programs and comport with basic copyright principles.[20]

As background to the analysis, Part I provides an overview of computer technology generally and the two principal aspects of application programming, the design of computer-human interfaces and software engineering. Part II presents an economic framework for analyzing legal protection of the intellectual work embodied in application programs. Parts III and IV, respectively, analyze the scope of copyright protection for application program code and computer-human interfaces.

I. An Overview of the Design of Application Programs

In order to evaluate legal rules protecting application programs, one must first have a working knowledge of computer technology. Section A presents an overview of modern computer technology and describes the function of application programs. Section B describes the essential features of an application program: the design of the computer-human interface and the layout and engineering processes used in structuring and coding the program's tasks.

A. *An Introduction to Computer Technology*[21]

To oversimplify a bit, a modern programmable computer consists of hardware—the physical computing machine—and software—the programs that operate the machine and perform data processing tasks for

19. *See* text accompanying notes 164-173 *infra*.

20. Though the proposed approach would be significantly better tailored to the attributes of software technology and market conditions than the direction of copyright doctrine suggested by the early cases, a further question remains: whether the unique attributes of computer software justify a *sui generis* form of legal protection. This article confirms earlier analyses arguing that existing modes of protection for intellectual work are not ideally suited to computer software. *See* Menell, *supra* note 9; Samuelson, *supra* note 2. Since this article is principally concerned with analyzing the application of traditional copyright principles to the protection of application programs, and because the design of a *sui generis* regime for computer software is discussed elsewhere, *see* Menell, *supra* note 9, at 1364-67, 1371, the article only indirectly addresses how a legislature might fashion a new system of legal protection for application programs.

21. *See generally* Susan Curran & Ray Curnow, Overcoming Computer Illiteracy: A Friendly Introduction to Computers (1984); Dennis Longley & Michael Shain, Dictionary of Information Technology (2d ed. 1986).

the user. A modern computer's basic hardware consists of the central processing unit (CPU) (which performs basic arithmetic operations such as addition and multiplication), internal memory storage, and a keyboard, disk drives, and/or other devices for transferring data and programs into and out of the internal memory.[22] In combination with the computer user's commands, computer software directs the hardware to perform data processing tasks.

The principal types of computer software are operating systems and application programs.[23] An operating system is a collection of programs that manages the computer's internal functions (*e.g.*, directs the flow of information among the computer's parts) and acts as an interface between the computer hardware, the application programs, and the computer user.[24] Application programs perform specific data processing tasks for the user. Bookkeeping, statistical and financial analysis, word processing, and video game programs are among the many types of application programs available today.

B. *"Writing" Application Programs*

There are several stages in the development of most application programs:[25] (1) *task definition*—describing the task to be accomplished; (2) *flowcharting*—depicting schematically the way in which the computer will accomplish the task;[26] (3) *coding*—converting the elements of the flowchart into machine-readable instructions;[27] (4) *debugging*—testing the program for accuracy, correcting programming errors, and verifying that the program functions properly; and (5) *documentation*—preparing materials that explain the functioning of the program.

In the first decades following the invention of computers, their us-

22. The complexity of computer hardware varies significantly across the range of computer sizes. At the high end of the hardware market are supercomputers and large-scale mainframes which offer the greatest processing speeds and data storage capacities. In the medium range are super minicomputers, medium mainframes, and minicomputers. At the low end are microcomputers.

23. A third important type of software is microcode. Microcode is a set of encoded instructions that can substitute for the hardwiring of certain basic operations within a computer's CPU. *See* David A. Patterson, *Microprogramming,* SCI. AM., Mar. 1983, at 50, 50.

24. *See* Rick Cook, *Operating Systems,* POPULAR COMPUTING, Aug. 1984, at 111; Hoo-Min D. Toong & Amar Gupta, *Personal Computers,* SCI. AM., Dec. 1982, at 87, 88.

25. *See generally* DAVID BENDER, COMPUTER LAW—SOFTWARE PROTECTION § 2.06(3) (1985); J.M. Yohe, *An Overview of Programming Practices,* 6 COMPUTING SURVS. 221 (1974).

26. Flowcharting is being replaced by the use of "design languages," which combine a rigid syntax with English text to enable a programmer to write a structured program which other programmers can more easily understand and review. *See generally* LAWRENCE J. PETERS, SOFTWARE DESIGN: METHODS & TECHNIQUES (1981).

27. The program's coding often proceeds through a number of stages. The flowchart typically is first translated into a higher level programming language (referred to as "source code"), such as FORTRAN, which uses arithmetic statements and English-like word statements. An assembler or compiler program then translates the source code program into "object code," strings of binary digits ("0"s and "1"s) that can be read by the computer. *See Copyright Protection of Computer Program Object Code,* 96 HARV. L. REV. 1723, 1724-25 (1983) (student author).

age was largely confined to persons with specific training in computer operation.[28] Because of the limited capabilities of the early machines and their users' familiarity with the technology, computer designers and programmers focused primarily on maximizing computing efficiency, rather than on accommodating the user.[29] While computing efficiency remains an important programming consideration, designing programs for easy use has become a high priority as computers have entered the lives of a large portion of the population, including many with little or no training in computer science.[30] This concern extends beyond merely teaching how to write and execute programs. The broader goal is to facilitate communication between computers and humans at the visual, audio, and tactile levels.[31]

As noted above, the first stage of designing an application program is identifying the user tasks that are to be accomplished. Because of the importance of accommodating users' needs, abilities, and limitations as well as the attributes of their work environments, a major aspect of the task definition process is understanding the users and determining how best to serve their needs. This will typically involve research into the nature of the task, the users' backgrounds and abilities, and the working environment. As the discussion below explains, the process by which user tasks are defined and interfaces designed has developed into a hybrid science drawing on a wide range of disciplines.

Once the tasks are clearly defined, the programmer turns to the tedious job of designing a program to serve the users as efficaciously as possible. Because the computer instructions serve no function other than to accomplish the defined tasks, the overriding concern in structuring the program is to meet the users' needs in the most efficient manner. The concept of efficiency in this context comprises: (1) code efficiency—maximizing the processing speed; (2) memory efficiency—using solution techniques and addressing methods to minimize the amount of memory needed to accomplish the desired tasks; and (3) input/output efficiency—maximizing the quality and speed of information transmission between the computer and the user or external

28. Ben Shneiderman, *The Future of Interactive Systems and the Emergence of Direct Manipulation*, in HUMAN FACTORS AND INTERACTIVE COMPUTER SYSTEMS 1, 2 (Y. Vassiliou ed. 1984) [hereinafter HUMAN FACTORS].

29. *See* Jack Grimes, Kate Ehrlich & Jerry Vaske, *User Interface Design: Are Human Factors Principles Used?*, SIGCHI BULL., Jan. 1986, at 22.

30. *See* BEN SHNEIDERMAN, DESIGNING THE USER INTERFACE: STRATEGIES FOR EFFECTIVE COMPUTER INTERACTION 15-18 (1987) (describing the wide range of computer applications and the diversity of people using computers); Shneiderman, *supra* note 28, at 1-3.

31. The means by which humans and computers communicate have been greatly expanded. In addition to the standard typewriter keyboard and the video screen, a variety of other physical devices, including function keypads, pointing devices such as light pens, joysticks, mice, touchscreens, track balls, and keypads, and speech recognition and generation devices have been developed to ease computer use. *See* B. SHNEIDERMAN, *supra* note 30, at 227-69.

hardware devices (*e.g.*, another computer).[32] The software engineering field strives to develop methods for writing more "efficient" and reliable programs at lower cost. The following sections describe the principal methods by which computer programmers seek to achieve the dual goals of maximizing user ease (or "user friendliness") and computer efficiency in designing application programs.

1. *The design of computer-human interfaces.*[33]

Over the past half century, great strides have been made in a number of disciplines—including education, graphic art, industrial design, industrial management, computer science, mechanical engineering, psychology, artificial intelligence, linguistics, information science, and sociology—in understanding human needs and designing products to serve these needs. During this time, there has been a gradual effort to unify this learning within a hybrid discipline generally referred to as the "study of human factors" or "human factor analysis." Recognizing the many ways that the study of human factors can aid computer system design, the computer industry has taken a particularly strong interest in the effort to synthesize and expand this learning.[34] As a result, the field of "computer-human interaction," a branch of human factor analysis, has become a vital force in most aspects of computer system design.[35] In no sector of the computer industry has the influence of computer-human interaction been more strongly felt than in the design of application programs.[36]

The developing field of computer-human interaction has changed profoundly the way in which application programs are both conceptualized and written.[37] The field has identified five human factor goals that

32. *See* ROGER S. PRESSMAN, SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH 284-86 (1982).

33. *See generally* B. SHNEIDERMAN, *supra* note 30.

34. *See Editorial,* 1 INT'L J. MAN-MACHINE STUD. i-ii (1969) (justifying publication of a new interdisciplinary journal focusing on the study of human-machine systems).

35. The number of persons with human factor analysis training at leading computer firms has grown dramatically during the past decade. Between 1975 and 1985, the number of members of the Human Factors Society working at 14 leading computer companies grew from 122 to 485. Grimes, Ehrlich & Vaske, *supra* note 29, at 22. Another sign of this field's importance is the proliferation of journals focusing on or regularly featuring articles dealing with the study of computer-human interaction: *Behaviour and Information Technology*; *International Journal of Man-Machine Studies*; *Human-Computer Interaction*; *ACM (Association for Computing Machinery) Computing Surveys*; *SIGCHI Bulletin* (quarterly newsletter of the ACM's Special Interest Group on Computer and Human Interaction); *Communications of the ACM, ACM Transactions on Office Information Systems*; *Ergonomics*; *Human Factors*; *IBM Systems Journal*; *IEEE Computer*; *IEEE Computer Graphics and Applications*; *IEEE Transactions on Systems, Man, and Cybernetics*; and *Journal of Applied Psychology*. In addition, a number of these organizations publish the papers and proceedings of their annual conferences.

36. *See generally* B. SHNEIDERMAN, *supra* note 30, at 4-8.

37. As an example of the impact of the field of computer-human interaction on application programming, Apple Computer, Inc. uses a "Human Interface Group," comprised of psychologists, technical writers, graphic designers, and computer scientists, in its software product development. Apple describes the function of this group as follows:

programmers should strive to achieve in designing application programs:[38] (1) minimize learning time, (2) maximize speed of performance, (3) minimize rate of user errors, (4) maximize user satisfaction, and (5) maximize users' retention of knowledge over time.

Application programmers attempt to utilize these objectives in designing computer-human interfaces. As a result, many of their design choices are guided by principles of human factor analysis that have evolved over years of empirical research. Among the design issues that have been studied are: the choice among command-based programs, menu-driven programs, and natural language programs;[39] menu design;[40] windowing versus scrolling;[41] the choice of abbreviations and command names;[42] the layout and scheme of graphical interfaces;[43] the color and highlighting of video displays;[44] the consistency of user inter-

---

> The Human Interface Group studies the aesthetic, psychological, graphical and ergonometric factors that influence the overall audiovisual appeal and success of the interface and applies the fruits of such studies to the revision of the interface and to the creation of new elements of the interface as new products are introduced or new ideas are added to the interface.
>
> Apple has a Human Interface Laboratory in which different interface types, styles and designs are systematically studied. A range of subjective and quantitative evaluations are conducted, including videotaping of user responses to collect verbal protocol data. The Human Interface Group also performs testing of the interface through interviews, questionnaires and performance metrics (such as speed and accuracy measures). The Group is also working to develop new interface paradigms in the use of color, sounds, input/output devices and visual presentation schemes.

Written Submission of Apple Computer, Inc., U.S. Copyright Office Public Hearing on Registration and Deposit of Computer Screen Displays 4-5 (Sept. 4, 1987) [hereinafter Apple Testimony] (on file with the *Stanford Law Review*).

38. *See* B. SHNEIDERMAN, *supra* note 30, at 14-15.

39. *See, e.g.*, Alexander G. Hauptmann & Bert F. Green, *A comparison of command, menu-selection, and natural language computer programs*, 2 BEHAV. & INFO. TECH. 163 (1983).

40. *See, e.g.*, B. SHNEIDERMAN, *supra* note 30, at 85-133; Patricia A. Billingsley, *Navigation Through Hierarchical Menu Structures: Does It Help to Have a Map?*, PROC. HUM. FACTORS SOC'Y, 26TH ANN. MEETING 103 (1982); Stuart K. Card, *User Perceptual Mechanisms in the Search of Computer Command Menus*, 1982 PROC.: HUM. FACTORS IN COMPUTER SYSTEMS 190 [hereinafter 1982 PROC.]; John I. Kiger, *The Depth/Breadth Trade-Off in the Design of Menu-Driven User Interfaces*, 20 INT'L J. MAN-MACHINE STUD. 201 (1984).

41. *See, e.g.*, Kevin F. Bury, James M. Boyle, R. James Evey & Alan S. Neal, *Windowing vs. Scrolling on a Visual Display Terminal*, 1982 PROC., *supra* note 40, at 41.

42. *See, e.g.*, P. Barnard, N. Hammond, A. MacLean & J. Morton, *Learning and Remembering Interactive Commands*, 1982 PROC., *supra* note 40, at 2; S.L. Ehrenreich & Theodora Porcu, *Abbreviations for Automated Systems: Teaching Operators the Rules*, in DIRECTIONS IN HUMAN/COMPUTER INTERACTION 111 (A. Badre & B. Shneiderman eds. 1982); Jarret Rosenberg, *Evaluating the Suggestiveness of Command Names*, 1982 PROC., *supra* note 40, at 12.

43. *See, e.g.*, B. SHNEIDERMAN, *supra* note 30, at 326-36; Christopher Herot, *Graphical User Interfaces*, in HUMAN FACTORS, *supra* note 28, at 83; Harvey Z. Kriloff, *Human Factor Consideration for Interactive Display Systems*, in ACM/SIGGRAPH WORKSHOP ON USER-ORIENTED DESIGN OF INTERACTIVE GRAPHICS SYSTEMS 45 (S. Trev ed. 1977); David E. Peterson, *Screen Design Guidelines*, in TUTORIAL: END USER FACILITIES IN THE 1980's 10 (J. Larson ed. 1982).

44. *See, e.g.*, B. SHNEIDERMAN, *supra* note 30, at 336-42; Izak Benbasat, Albert S. Dexter & Peter Todd, *The Influence of Color and Graphical Information Presentation in a Managerial Decision Simulation*, 2 HUMAN-COMPUTER INTERACTION 65 (1986); Beverly G. Knapp, Franklin L. Moses & Leon H. Gellman, *Information Highlighting on Complex Displays*, in DIRECTIONS IN HUMAN/COMPUTER INTERACTION, *supra* note 42, at 195.

faces;[45] and the design of direct manipulation[46] interfaces.[47] To aid in the implementation of these ideas, many firms engaged in application program development have compiled computer-interface guidelines and criteria to guide their programmers.[48]

2. *Software engineering.*

After the programming tasks have been identified and the methods for fostering computer-human interaction chosen, the next principal programming stage reduces these design considerations to a flow chart, which is eventually reduced to machine-readable code. In this stage, the programmer seeks to implement the program in the most efficient and reliable way.

The most common method for attacking complex programming problems is "top-down design," "a design strategy that breaks large, complex problems into smaller, less complex problems—and then decomposes each of those smaller problems, until the original problem has been expressed as some combination of many small, solvable problems."[49] Top-down design breaks down each task into a series of modules or subroutines, which in turn are broken down into their component parts. This process continues until the problem is represented by a flow chart that can be translated into a programming language (which in turn is translated into object code).[50]

---

45. *See, e.g.*, B. SHNEIDERMAN, *supra* note 30, at 69-71, 72-73; Fred H. Otte, *Consistent User Interface*, in HUMAN FACTORS, *supra* note 28, at 261.

46. Direct manipulation refers to means to enable users to manipulate a computer screen's contents directly, as in changing particular entries in a spreadsheet by moving the cursor to the cell and editing it, rather than using command languages.

47. *See, e.g.*, B. SHNEIDERMAN, *supra* note 30, at 179-223; Edwin L. Hutchins, James V. Hollan & Donald A. Norman, *Direct Manipulation Interfaces*, 1 HUMAN-COMPUTER INTERACTION 311 (1985).

48. *See, e.g.*, B. SHNEIDERMAN, *supra* note 30, at 60-62 (describing "eight golden rules of dialog design"); *see also id.* at 31. Apple Computer, Inc., for example, has developed a set of "Human Interface Guidelines" for its software developers. The Apple guidelines stress the following "fundamental principles" of user-friendly software design:

> 1. *Metaphors*—the interface should embody plain, simple metaphors for accomplishing tasks with audio and visual effects to support the metaphor.
> 2. *Direct Manipulation*—the interface should allow users to directly manipulate items or actions on the screen, rather than indirectly through abstract commands.
> 3. *See and Point*—the interface should allow the user to effect action by a see-and-point mechanism that is intuitive, rather than by a remember-and-type mechanism that can be intimidating, too abstract or unnatural.
> 4. *Consistency*—the symbols and mechanisms for accomplishing tasks should be consistent across all application programs through a consistent interface.
> 5. *WYSIWYG (What You See Is What You Get)*—the interface should present the information on the screen to the user in exactly the form that it will come out on the printer, removing the need to type in abstract formatting commands or to make mental calculations to envision how the screen translates to paper.

Apple Testimony, *supra* note 37, at 5-6.

49. EDWARD YOURDON, MANAGING THE STRUCTURED TECHNIQUES 61 (1986) (emphasis omitted).

50. In recent years, software engineers have developed more advanced design methodologies for particular types of programming tasks. These methodologies build upon the mod-

Though some programmers have distinctive programming styles,[51] what matters in the end (assuming that the assigned tasks have been accomplished) is the accuracy, efficiency, and reliability of the resulting program.[52] Program efficiency, measurable in such terms as processing speed, memory utilization, and quality and speed of input/output functions,[53] is determined by the internal structure of the subroutines, the algorithms,[54] and the code comprising the application program. In order to improve reliability (and ease of testability and correction) of programs, software engineers have developed guidelines for good programming practice.[55] Although there is some variation among sources regarding the correctness of particular guidelines,[56] and engineers constantly seek to develop better guidelines, programmers adhere widely to many conventions they consider to be good programming practice. These rules constrain and standardize the design choices regarding substance, structure, and form made in programming.[57]

---

ular programming concept used in the top-down design methodology. *See* R. PRESSMAN, *supra* note 32, at 230-39.

51. *See* Anthony L. Clapes, Patrick Lynch & Mark R. Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs,* 34 UCLA L. REV. 1493, 1535-36 (1987).

52. This proposition is stated clearly in Lotus Development Corporation's written statement for the U.S. Copyright Office:

> The user never sees the code. He or she couldn't care less whether it is in one computer language or another, or how it is expressed in the computer language it employs. All that user [sic] cares about is how he or she can use and interface with the program, with how the program operates and what it does for the user. . . . Well written code enables a program to run quickly, efficiently and reliably—all potentially important to a user.

Testimony of Thomas Lemberg, Vice President and General Counsel, and Ed Belove, Vice President, Research and Development, Lotus Development Corporation, Before the U.S. Copyright Office 6 (Sept. 9, 1987) (on file with the *Stanford Law Review*); *see also* BRIAN KERNIGHAN & P.J. PLAUGER, ELEMENTS OF PROGRAMMING STYLE 143-45 (2d ed. 1978); R. PRESSMAN, *supra* note 32, at 147-48; GERALD WEINBERG, THE PSYCHOLOGY OF COMPUTER PROGRAMMING 22-25 (1971); Eric Alderman, *Word 40: Fast at Last,* PC WORLD, Feb. 1988, at 138, 139 (noting the importance of processing speed for word processing programs); Duncan M. Davidson, *Protecting Computer Software: A Comprehensive Analysis,* 23 JURIMETRICS J. 339, 342 (1983); Karjala, *supra* note 9, at 38-39.

53. *See* R. PRESSMAN, *supra* note 32, at 147-48. The criteria for evaluating application program efficiency may vary depending on the types of tasks involved and the capacity of the hardware on which the software will be run.

54. An algorithm is a step-by-step procedure for achieving a particular result. Algorithms are typically expressed as formulas for solving mathematical problems. *Cf.* Allen Newell, *Response: The Models Are Broken, The Models Are Broken,* 47 U. PITT. L. REV. 1023 (1986) (discussing the difficulty of defining an algorithm).

55. *See* B. KERNIGHAN & P. PLAUGER, *supra* note 52, at 159-61 (listing 77 basic rules of good programming practice); *see also* MARTIN L. SHOOMAN, SOFTWARE ENGINEERING 116 (1983). Kernighan and Plauger's work is a respected reference in the software engineering field.

56. *See, e.g.,* M. SHOOMAN, *supra* note 55, at 116 n.2.

57. It should also be noted that there are more general efforts to standardize programming concepts and the reuse of proven program modules, though these efforts have been hindered by parochial interests. *See* M. SHOOMAN, *supra* note 55, at 16-17. One of this article's concerns is the effect of intellectual property rules on standardization of interfaces, though

3. *The process of technological advancement in application programming.*

The process of technological advancement in the application programming field, as in other areas of technology, is through rapid sequential improvements to the existing knowledge base.[58] One commentator notes that "[n]ew computer programs . . . often rely on existing programs, and this reliance will surely increase when programmers reach the stage of creating new programs by computer instead of human intellectual effort."[59] In addition, innovation in application programming can take the form of combining existing programs in a useful way.[60]

The development of the spreadsheet application program illustrates this process of rapid sequential innovation.[61] The idea for the spreadsheet was conceived in the spring of 1978 by Daniel Bricklin, a frustrated MBA student who disliked the tedium of performing repetitive mathematical calculations for homework assignments. An experienced computer programmer, Bricklin had little difficulty developing an electronic spreadsheet that would allow him to calculate entire tables of financial information simultaneously. By the spring of 1979, he had a working version of VisiCalc™ designed to run on the Apple II™ computer. The program quickly became a commercial success. Within a few years following the introduction of VisiCalc™, a number of new spreadsheets were introduced—such as Lotus 1-2-3™, Javeline™, and Supercalc™—offering a variety of improvements, including graphics capability, enhanced capacity, increased speed, and greater ease of use. More sophisticated programs offering even greater speed, enhanced graphics capability, and increased versatility are now challenging this second generation of spreadsheets.[62]

---

much of the general analysis would apply to standardization of software engineering methods as well.

58. *See* Karjala, *supra* note 9, at 39-41; INTERNATIONAL TRADE ADMINISTRATION, U.S. DEPARTMENT OF COMMERCE, A COMPETITIVE ASSESSMENT OF THE U.S. SOFTWARE INDUSTRY 10, 12, 15 (1984) [hereinafter SOFTWARE TRADE STUDY] (noting the fast obsolescence of microcomputer application programs due to the rapidly changing technology).

59. Dennis S. Karjala, *Lessons From the Computer Software Debate in Japan,* 1984 ARIZ. ST. L.J. 53, 67, 68 (noting further that "many new programs are developed through improvements or additions to existing programs").

60. *See* Michael J. Miller, *Software Integration,* POPULAR COMPUTING, Dec. 1983, at 106 (noting that the leading software companies are introducing sophisticated products that integrate standard business needs to permit data transfer, multiprocessing, and greater ease of use).

61. *See* Eric Bender, *VisiCalc Visionary,* PC WORLD, Jan. 1988, at 104-05; Churbuck & Freedman, *supra* note 15, at 1, col. 4; John P. Sumner & Steven W. Lundberg, *The Versatility of Software Patent Protection: From Subroutines to Look and Feel,* COMPUTER LAW., Jan. 1986, at 1, 1-3 (describing the spreadsheet's early development).

62. *See* Quattro advertisement, PC WORLD, Feb. 1988, at 1; Mike Hogan, *Product Outlook: Fresh from the Spreadsheet Oven,* PC WORLD, Feb. 1988, at 100-02 (describing the Quattro spreadsheet); Lawrence J. Magid, *'Surpass' Spreadsheet Program Lives Up to Name, Beats Lotus 1-2-3,* Wash. Post, Apr. 25, 1988, at 26.

II. AN ECONOMIC FRAMEWORK FOR ANALYZING LEGAL PROTECTION FOR APPLICATION PROGRAMS[63]

The natural starting point for analyzing legal protection for application programs is the stated constitutional purpose underlying Congress's power to grant patents and copyrights: "To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[64] The Supreme Court has interpreted the law implementing this language to mean that the author's benefit is "secondary"[65] to advancement of the arts and sciences for society's benefit. The "economic philosophy behind the clause . . . is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare."[66]

This economic philosophy, however, does not necessarily call for the provision of legal protection for intellectual work. A fundamental principle of neoclassical economic theory is the proposition that in the absence of market imperfections,[67] competition will ensure an efficient allocation of resources.[68] Thus, absent imperfections, the market will compensate authors sufficiently to induce the optimal rate of advancement in the arts and sciences. Even where market failures are present, government intervention is only justified (in an economic sense) if its costs do not outweigh the benefits of such remedial measures.[69] This section discusses the two principal market failures present in the application program market—the public goods and network externalities problems—and describes the design of government policies to remedy these market shortcomings. Both of the concerns are directly applica-

63. This section parallels the general theoretical discussion in Menell, *supra* note 9, at 1336-45, which emphasizes the analysis of legal protection for computer operating systems. The present section's discussion, by contrast, focuses on the analysis of legal protection for application programs.

64. U.S. CONST. art. I, § 8, cl. 8.

65. United States v. Paramount Pictures, Inc., 334 U.S. 131, 158 (1948); *see also* Fox Film Corp. v. Doyal, 286 U.S. 123, 127-28 (1932) (noting that "[t]he sole interest of the United States and the primary object in conferring the monopoly lie in the general benefits derived by the public from the labors of authors. A copyright, like a patent, is 'at once the equivalent given by the public for benefits bestowed by the genius and meditations and skill of individuals and the incentive to further efforts for the same important objects.' " (citation omitted)).

66. Mazer v. Stein, 347 U.S. 201, 219 (1954); *see also* Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs,* 84 HARV. L. REV. 281, 284-93 (1970); Robert C. Denicola, *Copyright in Collections of Facts: A Theory for the Protection of Nonfiction Literary Works,* 81 COLUM. L. REV. 516, 519 (1981).

67. Market imperfections arise in many forms, the most important of which are the presence of monopoly (single seller) or monopsony (single buyer) power, incomplete or incorrect information on the part of consumers or producers, and externalities—costs or benefits not accruing to the person making the decisions giving rise to such costs or benefits. *See generally* STEPHEN BREYER, REGULATION AND ITS REFORM 15-32 (1982).

68. *See generally* PAUL SAMUELSON & WILLIAM D. NORDHAUS, ECONOMICS 678 (12th ed. 1985).

69. *See* Arnold Plant, *The Economic Aspects of Copyright in Books,* 1 ECONOMICA (N.S.) 167 (1934). *See generally* Breyer, *supra* note 66.

ble to the design of legal protection for computer-human interfaces, while the public goods problem is the major concern with regard to the design of legal protection for application program code.[70]

A. *Innovation in Application Programming as a Public Good*

Markets for goods embodying intellectual work exhibit an externality commonly referred to as the "public goods" problem.[71] Public goods have two distinguishing features: (1) nonexcludability—it is difficult to exclude those who do not pay for the good from consuming it, and (2) nonrivalrous competition—additional consumers of the good do not deplete the quantity of the good available to others. Lighthouses, television signals, beautiful gardens, and national defense are classic examples of public goods. The private market will tend to undersupply these goods because producers cannot reap the marginal (incremental) value of their investment in providing such goods.[72]

As CONTU well recognized, the information comprising innovation in application programs is a prime example of a public good.[73] Given the ease and low cost of copying application programs, it is often impossible to exclude nonpurchasers from an application program's benefits once it is commercially available. Moreover, one person's use of the information does not detract from others' use of that same information. Since application program creators cannot reap the marginal value of their efforts under these circumstances, they will, in the absence of other incentives to innovate, tend to undersupply new and better application programs.

1. *The role of legal protection in remedying the public goods problem.*

One means by which the government corrects the public goods problem inherent in the creation of intellectual work, and thus increases the public welfare, is to provide limited periods of legal protection to products embodying novel ideas and literary works containing original expression. Enhancing the scope of intellectual property protection (by increasing the duration or breadth of production) increases the expected reward to those engaged in intellectual work by enhancing the opportunity for monopolistic exploitation of any products created. These increased rewards encourage inventive activity, which in turn spurs the discovery of new and better products.[74]

---

70. The network externality problem has some relevance to application program code. *See* note 119 *infra*.

71. *See generally* Kenneth J. Arrow, *Economic Welfare and the Allocation of Resources for Invention*, in THE RATE AND DIRECTION OF INVENTIVE ACTIVITY: ECONOMIC AND SOCIAL FACTORS 609 (1962).

72. *See* P. SAMUELSON & W. NORDHAUS, *supra* note 68, at 48-49, 713-15; Paul A. Samuelson, *The Pure Theory of Public Expenditure*, 36 REV. ECON. & STATISTICS 387 (1954).

73. *See* CONTU REPORT, *supra* note 1, at 9-12.

74. *See* Louis Kaplow, *The Patent-Antitrust Intersection: A Reappraisal*, 97 HARV. L. REV. 1813, 1823-24 (1984).

This linkage between the scope of legal protection and the public welfare, however, is greatly complicated with respect to technologies, such as application programming, in which innovation proceeds through sequential improvements to the state of the art.[75] Protection must provide sufficient lead time to a product's first developer to reap an adequate reward for his or her efforts; but protection beyond that time threatens to delay design improvements, refinements, and adaptations of the product to different uses.[76]

2. *Assessing the need for legal protection.*

The above linkage establishes only the potential for benefits from legal protection for intellectual work. A complete analysis requires a comparison of the extent of the market failure to the costs of government policies to remedy such failure. It is also necessary to consider other policies to address the market failure.

The market itself often provides means by which producers of public goods can realize sufficient rewards to encourage them to produce such goods. The first to introduce a product can in many contexts earn substantial and long-lived advantages in the market.[77] In addition, producers of innovative products can internalize some of their research benefits by requiring purchasers to enter into long-term maintenance and updating contracts.[78] This option, however, typically is viable only with respect to application programs for mainframe computers, minicomputers, and higher end microcomputers.[79] Furthermore, this strategy does not completely solve the public goods problem because it does not entirely preclude access by nonpurchasers.

One way application program producers can attempt to overcome the public goods problem directly is by entering into licensing agree-

75. *See* notes 58-62 *supra* and accompanying text; *see also* Menell, *supra* note 9, at 1338.

76. *See* Karjala, *supra* note 9, at 95; Menell, *supra* note 9, at 1338.

77. *See* JOE BAIN, BARRIERS TO NEW COMPETITION 114-43 (1962); Richard Schmalensee, *Product Differentiation Advantages of Pioneering Brands*, 72 AM. ECON. REV. 349, 360 (June 1982). *See generally* RONALD BOND & DAVID LEAN, CONSUMER PREFERENCE, ADVERTISING, AND SALES: ON THE ADVANTAGE OF EARLY ENTRY (BUREAU OF ECONOMICS, U.S. FED. TRADE COMM'N, WORKING PAPER NO. 14, OCT. 1979); *cf.* Richard C. Levin, Alvin K. Klevorick, Richard P. Nelson & Sidney G. Winter, Appropriating the Returns from Industrial Research and Development, Brookings Papers on Economic Activity 783, 816 (1987) [hereinafter R. Levin] (finding in a cross-sectional survey of manufacturing industries that lead time, secrecy, learning advantages, and sales and service efforts are typically more important than patent protection as a means of appropriating rewards for research and development).

78. *See* SOFTWARE TRADE STUDY, *supra* note 58, at 12 (noting that software packages for high-end computers typically are not sold outright but licensed, accompanied by maintenance contracts).

79. *See id. But cf.* Michael Antonoff, *The New Spreadsheets*, PERS. COMPUTING, Nov. 1985, at 107 (noting that some spreadsheet manufacturers tie maintenance and program enhancement services to their programs). Application program manufacturers use a related strategy of tying their programs to documentation and other support material (independently copyrighted). This technique is imperfect to the extent that such supporting material can be photocopied without detection or that similar material prepared by others is available at low cost.

ments with their customers, prohibiting reproduction and dissemination of information embodied in their products. Most producers of application software packages for mainframes and minicomputers license their products on a monthly or yearly basis.[80] Although this strategy is also used by many microcomputer application program manufacturers (in the form of shrink-wrap licenses[81]), it is usually infeasible to police these agreements, particularly with respect to high-volume products.[82] Furthermore, questions remain about the enforceability of such agreements with regard to mass-marketed programs[83] and whether this form of protection is preempted by the Copyright Act of 1976.[84]

A second way application program producers have attempted to internalize the public goods problem is by installing anticopying devices in their programs that impede reproduction and disclosure of intellectual work embodied in such products.[85] In practice, however, these devices have not proven to be uncrackable. Additionally, and undesirably, they limit the customer's ability to make back-up copies and modify the program, and encourage wasteful expenditures in tearing down a wall the producer has built.[86]

Another solution to the public goods problem is the use of private arrangements such as research consortia to share the cost of funding "leaky" technologies. While such arrangements might not completely prevent nonpurchasers from eventually gaining access to the consortia's intellectual work, they do encourage research and development by spreading costs across a large group of potential beneficiaries of the research; and they allow earlier access to new technologies to those who do participate in the joint venture. The National Cooperative Research Act of 1984 (NCRA)[87] has significantly expanded firms' ability to engage in such consortia. The NCRA codifies a limited antitrust exclu-

80. *See* SOFTWARE TRADE STUDY, *supra* note 58, at 12.

81. A "shrink-wrap" license purports to bind the purchaser to the terms of a licensing agreement if the purchaser removes the heat-sealed plastic wrapping surrounding the documentation and/or floppy disk. *See* David D. Bahler, *Shrink-Wrapped Software Agreements,* 8 LICENSING L. & BUS. REP. 37, 39 (1985); John R. Harris, *A Market-Oriented Approach to the Use of Trade Secret or Copyright Protection (or Both?) for Software,* 25 JURIMETRICS J. 147, 154 (1985); Michael J. McNeil, *Trade Secret Protection for Mass Market Computer Software: Old Solutions for New Problems,* 51 ALB. L. REV. 293, 307-08 (1987).

82. David Bender, *Protection of Computer Programs: The Copyright/Trade Secret Interface,* 47 U. PITT. L. REV. 907, 922-23 (1986).

83. *See* Bahler, *supra* note 81, at 39-42; Bender, *supra* note 82, at 922-24; Miles R. Gilburne & Ronald L. Johnston, *Trade Secret Protection for Software Generally and in the Mass Market,* 3 COMPUTER/L.J. 211, 229-37 (1982); McNeil, *supra* note 81, at 308-09.

84. 17 U.S.C. § 301(a) (1977); *see also* Bender, *supra* note 82, at 924-39.

85. *See* Gilburne & Johnston, *supra* note 83, at 225-27.

86. *See* Jonathan Sacks, *To Copy Protect or Not to Copy-Protect?,* POPULAR COMPUTING, Oct. 1985, at 73.

87. 15 U.S.C. § 4301 (Supp. 1988). *See generally* Christopher O.B. Wright, *The National Cooperative Research Act of 1984: A New Antitrust Regime for Joint Research and Development Ventures,* 1 HIGH TECH. L.J. 133 (1986).

sion (based on a rule-of-reason standard)[88] for joint research and development ventures engaged in theoretical analysis and experimentation,[89] "the development or testing of basic engineering techniques,"[90] the production of test models and prototypes (including process systems),[91] and the collection and exchange of research information.[92] In addition to clarifying the antitrust law applicable to joint ventures of this kind, the NCRA encourages joint research and development ventures by dulling the incentive of competitors or the government to bring antitrust actions through eliminating the availability of treble damages where the venture has been duly registered with and approved by the Department of Justice and the Federal Trade Commission.[93]

In the first four years of the NCRA, more than 100 research consortia have registered under the Act.[94] A number of them were formed to conduct software-related research.[95] Among the most important are the Microelectronics and Computer Technology Corporation (MCC) and the Software Productivity Consortium. MCC, which as of early 1986 had more than twenty member companies, 400 employees (provided from member companies), and an annual research budget of $60 to $70 million, plans to study, among other high-technology fields, the following software-related areas: the creation of powerful new tools for advanced software development, problems in advanced computer architecture (such as parallel processing), artificial intelligence, and expert systems.[96] The Software Productivity Consortium brings together fifteen major defense contractors interested in developing advanced

88. 15 U.S.C. § 4302 (Supp. 1986).

89. *Id.* § 4301(a)(6)(A).

90. *Id.* § 4301(a)(6)(B).

91. *Id.* § 4301(a)(6)(C).

92. *Id.* § 4301(a)(6)(D).

93. *Id.* §§ 4303, 4305. In addition, the Act further discourages questionable private antitrust suits by providing for the award of attorney's fees to "a substantially prevailing party [plaintiff or defendant] . . . if the claim, or the claimant's conduct during the litigation of the claim was frivolous, unreasonable, without foundation, or in bad faith." *Id.* § 4304(a). Furthermore, a prevailing party's fee award is subject to an offset for such portions of the case that the court deems "frivolous, unreasonable, without foundation, or in bad faith." *Id.* § 4304(b).

94. National Cooperative Research Act of 1984 Federal Register Publications Listing.

95. *See, e.g.,* Consortium Notice, 50 Fed. Reg. 2633 (1985) (notice of joint venture pursuant to the NCRA by Microelectronics and Computer Technology Corp.); *id.* (notice by Software Productivity Consortium); *id.* at 3425-26 (notice by Computer Aided Manufacturing-International); *id.* at 41,232 (notice by Applied Information Technologies); 51 Fed. Reg. 21,260 (1986) (notice by Corp. for Open Systems); 52 Fed. Reg. 4065 (1987) (notice by Industry/University Cooperative Research Center for Software Engineering).

A number of research consortia have been formed in the semiconductor and telecommunications industries; these technologies are closely aligned with software technology. *See* 50 Fed. Reg. 4280, 4280-81 (notice by Bell Communications Research Inc.); *id.* at 26,850 (notice by Semiconductor Research Corp.); *id.* at 50,864 (notice by Intel Corp./Xicor Corp.); 53 Fed. Reg. 17,987 (1988) (notice by SEMATECH).

96. *See* KENNETH FLAMM, TARGETING THE COMPUTER 114-16 (1987).

software tools to reduce computer programming costs.[97] It plans to employ 500 persons at its software development facility.

A related approach to the public goods problem associated with computer software is for the industry's major firms to work together to develop standard operating systems and interfaces in the public domain. Such network software helps to define product markets, enabling firms to devote their individual research and development energies to improving the products designed to run on particular standards. A group of many of the world's leading computer manufacturers—including Digital Equipment Corporation, Hewlett-Packard, IBM, and Siemens—has recently established the Open Systems Foundation in order to develop an open software environment.[98] The venture's stated purpose is to make it easier for consumers to use computers and software from many vendors.[99]

Though the market might not be able to remedy the public goods problem completely, especially with regard to mass-marketed application programs, government subsidies provide important additional incentives for research and development of improved application programs.[100] In general, the federal government funds more than one-third of all research and development in the United States.[101] This support has been particularly important in the fields of mathematics and computer science.[102] The National Science Foundation's Computer and Information Science and Engineering Program (CISE) has funded projects in the following software-related areas: advanced scientific computing (which includes the development of innovative software, graphical techniques, and efforts to standardize software); computer and computational research (which includes new software approaches and languages for parallel computer architectures); networking and communications research and infrastructure; and cross-disciplinary activities (which includes development and evaluation of experimental research activities).[103]

---

97. *See id.* at 116.

98. *See* Open Software Foundation, New Foundation to Advance Software Standards, Develop and Provide Open Software Environment, Press Release (May 17, 1988).

99. *Id.* It is not clear, however, whether such efforts to date have promoted or hindered the adoption of industry standards. *See* Andrew Pollack, *Computer 'Gangs' Stake Out Turf,* N.Y. Times, Dec. 13, 1988, at D1, col. 3.

100. *See* MERTON J. PECK, GOVERNMENT RESEARCH AND DEVELOPMENT SUBSIDIES IN THE AMERICAN ECONOMY? 17, 18, 21, 44 (Economic Research Inst., Economic Planning Agency, Tokyo, Japan, Discussion Paper No. 35, 1985); SOFTWARE TRADE STUDY, *supra* note 58, at 56-57. *See generally* KENNETH FLAMM, CREATING THE COMPUTER 29-79, 251-55 (1988); K. FLAMM, *supra* note 96, at 42-124.

101. *See* CBEMA DATA, *supra* note 5, at 38.

102. In 1986, 8.7% of the National Science Foundation's research budget of $1.705 billion and 10.3% of the Department of Defense's research budget of $3.366 billion went toward research in mathematics and computer science. *See* NATIONAL SCIENCE FOUNDATION, FEDERAL FUNDS FOR RESEARCH AND DEVELOPMENT: FISCAL YEARS 1986, 1987, AND 1988, at 43-44, 65-66 (1987).

103. *See* NATIONAL SCIENCE FOUNDATION, FEDERAL R&D FUNDING BY BUDGET FUNCTION:

In addition, the Department of Defense currently has major projects to develop integrated and automated software design tools,[104] multiprocessing,[105] and artificial intelligence (speech and natural language understanding, vision, and expert systems).[106] The National Aeronautics and Space Administration is spending $8 billion over the period 1984-94 to develop network operating systems and software tools for the space station project.[107] Although these projects are directed toward military and space rather than commercial applications, they are expected to have some spillover benefits for the commercial sphere.[108]

Furthermore, universities have produced many of the significant advances in computer programming.[109] The funding for university research comes from direct university sources, the federal government,[110] and increasingly from private industry.[111] Universities are currently conducting important research in the fields of advanced programming languages, software engineering,[112] the design of computer-human interfaces,[113] and artificial intelligence,[114] areas of great importance for the advancement of application programming.

The foregoing discussion indicates that various market factors as

---

FISCAL YEARS 1987-1989, at 49-50 (1988). The CISE Program budget is estimated to grow to $127 million for 1989. *Id.* at 49.

104. *See* SOFTWARE TRADE STUDY, *supra* note 58, at 57 ($5 million spent on the Software Technology for Adaptable and Reliable Systems project in 1984).

105. *See id.* (Strategic Computing project costs estimated at $750 million to $1 billion over the period 1984-1994).

106. *See id.*

107. *See id.*

108. *See* M. PECK, *supra* note 100, at 17. The government's role in funding research and development in the computer field has shifted from a "pervasive, all-encompassing" influence to a dominant role only in the riskiest and most expensive products to enter the commercial marketplace. But as one expert on the evolution of the computer industry notes, "[T]oday's leading-edge product is tomorrow's mundane workhorse in th[e computer] industry!" K. Flamm, *supra* note 96, at 123-24.

109. *See, e.g.*, K. FLAMM, *supra* note 96, at 51-70, 117, 123. Although much university research has traditionally been in the public domain, *see, e.g.*, ARCHIE M. PALMER, UNIVERSITY RESEARCH AND PATENT POLICIES, PRACTICES, AND PROCEDURES 7 (1962), many universities have begun to seek intellectual property protection actively for work done on their campuses. *See* Phyllis S. Lachs, *University Patent Policy*, 10 J.C. & U.L. 263 n.1 (1983-1984); *cf.* K. FLAMM, *supra* note 96, at 120-21 (discussing the allocation of patent rights for research funded by the federal government).

110. In 1988, the federal government gave universities an estimated $95 million for research in mathematics and computer science. NATIONAL SCIENCE FOUNDATION, *supra* note 102, at 160.

111. In 1987, the Industry/University Cooperative Research Center for Software Engineering was formed and registered with the Department of Justice under the National Cooperative Research Act of 1984, 52 Fed. Reg. 4065 (1987); *see also* text accompanying notes 87-93 *supra* (discussing NCRA). The Center's objectives are: "To sponsor research designed to improve the productivity of computer software developers and the quality of computer software; to build models and prototypes of tools which are related to these objectives; and to train graduate level students in the field of software engineering." 52 Fed. Reg. 4065.

112. *See* 52 Fed. Reg. 4065.

113. *See, e.g.*, Ben Shneiderman, *Human-Computer Interaction Research at the University of Maryland*, SIGCHI BULL., Jan. 1986, at 27.

114. *See* SOFTWARE TRADE STUDY, *supra* note 58, at 60.

well as government and university support for research tend to alleviate the public goods problem associated with application program creation and development. In certain market sectors, such as application programming for mainframe computers and minicomputers where licensing is feasible, market mechanisms and government subsidies may largely remedy the appropriation problem.

3. *Balancing the costs and benefits of legal protection.*

Even if the market and nonmarket mechanisms discussed above do not completely remedy the public goods problem associated with the creation of application programs, providing intellectual property rights must still stand or fall on whether the benefits of such protection outweigh the costs.[115] It is important in considering this balance to keep in mind that the costs of developing microcomputer software are low relative to the costs of other factors of production.[116] One survey of microcomputer software firms found that research and development costs are approximately 15 percent of total revenues, as compared with 35 percent for marketing, 20 percent for management, and 15 percent for production.[117] The benefits of legal protection flow from the inventions that never would have occurred absent the availability of protection and the discovery of certain inventions sooner as a result of the availability of intellectual property rights. Against these benefits must be weighed the direct costs of research and development associated with increased inventive activity, the inhibiting effects of intellectual property rights on sequential innovation, the monopoly costs resulting from legal protection, the widespread costs of keeping abreast of the property rights of others, the increased transaction costs for those creators wishing to make use of others' inventions (*e.g.*, through licensing), and the administrative costs of the intellectual property systems (*e.g.*, the costs of determining whether a patent should be granted, and the social costs of using the legal system to enforce intellectual property rights).

At a minimum, the foregoing analysis suggests that the government should approach the provision of legal protection for intellectual work embodied in application programs with great caution. The feasibility of direct licensing agreements and service and updating contracts significantly alleviates the public goods problem associated with application programs for medium- and large-scale computers. And with regard to microcomputer application programs, the relatively low development costs, the rapid obsolescence rate, the market mechanisms for internal-

115. *See* FRITZ MACHLUP, AN ECONOMIC REVIEW OF THE PATENT SYSTEM 44-73 (Subcomm. on Patents, Trademarks, and Copyrights of the Senate Comm. on the Judiciary, 85th Cong., 2d Sess., Study No. 15, 1958); William F. Baxter, *Legal Restrictions on Exploitation of the Patent Monopoly: An Economic Analysis*, 76 YALE L.J. 267, 271 (1966).

116. *See* SOFTWARE TRADE STUDY, *supra* note 58, at 10.

117. *See id.* at 15-16 (the remaining 15% of total revenues is profit).

izing research and development costs, and the extent of public subsidies and public domain research with regard to basic research and development attenuate the public goods case for protecting application programs expansively. Finally, in light of the rapid sequential nature of technological advancement in the application programming field, overbroad protection could impose significant costs in the form of monopoly pricing and choking off later innovation.

B. *Network Externalities—Fostering Standardized and User-Friendly Computer-Human Interfaces*

The second principal market failure in the application program market, one that CONTU entirely overlooked in its analysis and recommendations, relates to computer users' desire for application programs using standardized computer-human interfaces.[118] This feature of consumer demand for application programs is part of a more general phenomenon that economists refer to as the problem of network externalities.[119] The standardization concern with respect to application programs is akin to the classic network externality that flows from the prevalence of a standard typewriter keyboard.[120] Because almost all English language typewriters feature the same key configuration, commonly referred to as "QWERTY," typists need learn only one keyboard system. This has the advantage of increasing labor mobility and reducing retraining costs. Similarly, the greater the extent to which computer-human interfaces for particular application programs, such as word processing or spreadsheets, are standardized, the easier it will be for computer users to utilize their skills in different working environments.

Computer users also want the computer-human interface to be as easy to use as possible—*i.e.*, "user friendly."[121] In part, user friendli-

118. In a survey of 2000 organizations engaged in the fields of research, education, government, and finance regarding their criteria for purchasing software, 86.4% stated that compatibility with other software in use was "very important" or "important" to their decision. SOFTWARE TRADE STUDY, *supra* note 58, at 50 (citing *1984 Software User Survey*, SOFTWARE NEWS (1984)) [hereinafter *1984 Software User Survey*].

There is also a standardization concern with regard to aspects of software engineering. Standardized design methodologies, programming languages, coding conventions, and program modules can reduce the costs of writing application programs. *See* notes 55-57 *supra*. Although this section of the article's discussion focuses on the standardization concerns associated with computer-human interfaces, much of the analysis is applicable to standardization in software engineering.

119. More technically, the term "network externality" describes a class of goods for which the utility (or satisfaction) derived from the good's consumption increases with the number of other persons consuming the good. *See* Michael L. Katz & Carl Shapiro, *Network Externalities, Competition, and Compatibility*, 75 AM. ECON. REV. 424, 424 (June 1985). *See generally* DAVID HEMENWAY, INDUSTRYWIDE VOLUNTARY PRODUCT STANDARDS (1975).

120. *See* Paul A. David, *Clio and the Economics of QWERTY*, 75 AM. ECON. REV. 332 (May 1985).

121. In the survey cited above, 99.3% of respondents stated that ease of use was "very important" or "important" to their software purchasing decision. *1984 Software User Survey*, *supra* note 118.

ness is a function of what the user has already learned: Things that are familiar tend to be easier to use. More generally, user friendliness is determined by the extent to which the design is tailored to achieve human factor goals—such as maximizing performance speed, minimizing learning time, minimizing rate of errors, and maximizing retention over time.[122] The typewriter key configuration example aptly illustrates the possible discordant relationship between the standardization problem and the concern that software be user friendly. As noted above, typists benefit from a common key configuration on almost all English language typewriters. Since this configuration was designed primarily to avoid key jamming by "one finger" typists on early, mechanically primitive typewriters, it does not promote the fastest typing speed and therefore does not serve typists' objective of getting their typing tasks done as quickly and accurately as possible. U.S. Navy studies and world typing speed records attest that the Dvorak Simplex Keyboard, a typewriter key configuration patented in 1932 by August Dvorak and W.L. Dealey, is significantly more efficient than the QWERTY system.[123] In the absence of the costs of switching systems, notably equipment replacement costs and retraining, the preferred state of affairs would be for the Dvorak system to be the standard.[124] Though users value standardization, the standard adopted may not be the most user-friendly system possible.

The discussion below will elaborate on the dual and potentially conflicting concerns in designing legal protection for computer-human interfaces: to promote adoption of uniform interfaces while providing incentives for the development of better computer-human interfaces.

1. *The negative effects of legal protection for computer-human interface standards on the realization of network externalities.*

The extent to which standardized computer-human interfaces emerge depends in large part on whether application program producers have correct incentives to adopt compatible products, thereby enlarging existing networks. The presence of network externalities, however, might encourage firms to adopt noncompatible product standards, even though adopting compatible products would increase net social welfare.[125] The explanation for this behavior is that by adopting a compatible standard, a firm entering the market enlarges the size of a network comprising both its product and its rivals' products. This will have the effect of increasing the desirability of the rivals' products to

122. *See* B. SHNEIDERMAN, *supra* note 30, at 14-15.

123. *See* David, *supra* note 120, at 332.

124. The QWERTY system's persistence, despite the availability of less expensive means for switching equipment on microcomputers (though not without retraining costs), remains a conundrum. For an intriguing and delightful discussion of this issue, see David, *supra* note 120.

125. *See* Katz & Shapiro, *supra* note 119, at 435.

consumers, thereby reducing the adopter's market share (although of a larger market) relative to what it would have been had the firm adopted a noncompatible product standard. Thus, though the net social welfare of adopting its rivals' standard may exceed the net social welfare of introducing an incompatible standard, the entrant may nonetheless prefer to adopt the incompatible standard because the entrant cannot appropriate all the benefits of compatibility, some of which accrue to past and present purchasers of rivals' products.[126]

The following example illustrates this effect. Firm A develops the first application program for analyzing inventory for a widget factory. The program uses a particular function key configuration,[127] which we will assume for the purpose of this example is not in itself inherently functional. In other words, prior to widespread user learning of a particular program, human factor analysis would not prefer any particular configuration. Although present and future users of application programs performing this data processing task would benefit from widespread adoption of Firm A's particular key configuration, Firm B, in designing a competing application program, might decide to change the order of the function keys to prevent Firm A from being able to offer customers access to the larger network that includes Firm B's program. A principal advantage to Firm A of a larger network is being able to offer Firm B's customers related or enhanced programs without requiring Firm B's customers to incur retraining costs.

The availability of legal protection for computer-human interfaces strengthens this adverse incentive by allowing firms with brand recognition to reap increased rewards from developing noncompatible interfaces. In the absence of legal protection for such computer-human interfaces, the private benefits from introducing a noncompatible interface will be short-lived. As the interface gains acceptance in the marketplace, other firms will adopt it, thereby reducing the first producer's market share. By contrast, legal protection greatly increases the rewards that an application program producer can reap by successfully introducing a noncompatible interface. Legal protection allows the firm to introduce its program into the market without expanding the network of its rivals. Society, however, loses the benefit that would have resulted from the expanded network. In addition, the intellectual property right enables the producer to prevent other manufacturers from adopting the new interface. In this way, the developer of the noncompatible user interface can enjoy a long-term monopoly in the interface, thereby making the firm even less likely to make the socially optimal choice. Adopting a noncompatible interface, therefore, not only prevents realization of a social welfare increase from network ex-

126. *See id.*

127. Function keys are extra keys on computer keyboards that can be programmed to perform specific commands, such as moving blocks of text or changing the margins.

pansion, it also increases the social loss due to monopolization of network technologies.

The availability of legal protection for computer-human interfaces exacerbates the network externality problem not only by encouraging producers of new application programs to choose noncompatible standards; it also discourages firms seeking to introduce application programs featuring improvements or designed to serve distinct, but related, markets from using popular computer-human interfaces. Legal protection requires the new product manufacturer to obtain a license for the user interface from the intellectual property right holder. The transaction costs of licensing, including those resulting from strategic behavior,[128] might vitiate the deal.[129] And although in a static competition model there is little reason to believe that the interface owner would be unwilling to license the interface for a mutually agreeable price (assuming away transaction cost problems), the potential for inhibiting competitors' innovation efforts might lead the interface owner to prefer not to license, even at a price that gave the owner the full static surplus.[130]

Computer system hardware that can run a variety of application programs can significantly reduce the network externality problem. Such hardware enables computer users to bring their acquired knowledge with them merely by having their own software. For example, a person trained to use the WordPerfect™ software package will be able to transfer that skill to any work environment with a microcomputer system that can run WordPerfect™. If such computer users must share data files prepared for another system, however, the network externality problem will persist. The extent of the problem will depend on the amount of interdependency among computer users utilizing different application programs in the work environment and the costs of converting data and communicating across incompatible interfaces.[131]

---

128. Strategic behavior refers to the use of bargaining tactics, such as bluffs, for the purpose of obtaining a larger share of the gains from trade. *See* Robert Cooter, *The Cost of Coase,* 11 J. LEGAL STUD. 1, 20 (1982); A. Mitchell Polinsky, *Resolving Nuisance Disputes,* 32 STAN. L. REV. 1075, 1092 (1980).

129. *See* Cooter, *supra* note 128, at 20; *cf.* Robert Cooter, Stephen Marks & Robert Mnookin, *Bargaining in the Shadow of the Law: A Testable Model of Strategic Behavior,* 11 J. LEGAL STUD. 225 (1982).

130. *See* F.M. SCHERER, INDUSTRIAL MARKET STRUCTURE AND ECONOMIC PERFORMANCE 450-54 (2d ed. 1980); Nancy T. Gallini & Ralph A. Winter, *Licensing in the Theory of Innovation,* 16 RAND J. ECON. 237, 249 (1985); Louis Kaplow, *Extension of Monopoly Power Through Leverage,* 85 COLUM. L. REV. 515, 530-36 (1985); Michael L. Katz & Carl Shapiro, *On the Licensing of Innovations,* 16 RAND J. ECON. 504, 505 (1985).

131. The network externality problem is also alleviated to the extent low-cost conversion programs are available. *See generally* Joseph Farrell & Garth Saloner, The Economics of Converters (June 1988) (unpublished manuscript).

2. *The positive effects of legal protection for computer-human interface standards on innovation in and adoption of improved interfaces.*

While a widely adopted computer-human interface can offer important benefits to computer users, it can also "trap" the industry in an obsolete or inferior standard.[132] The installed base built upon the "old" standard—reflected in past purchases of application programs embodying the old standard and human capital investment (worker training)[133] specific to the old standard—can create excess inertia that makes it much more difficult for any one producer to break away from the old standard by introducing a noncompatible interface. This inertia can result even if the new interface offers a significant technological improvement over the current standard.[134] In this way, network externalities can retard innovation and slow or prevent adoption of improved interfaces.[135]

The availability of carefully tailored legal protection for product standards can alleviate the excess inertia effect by assuring innovators of better standards a limited monopoly in the event their standards break into the market. Without the availability of legal protection, innovators' profits would be competed away quickly as other firms introduced competing products embodying the improved standard. Because brand recognition is so influential in being able to introduce a new product quickly,[136] legal protection for improved interfaces is par-

132. *See* Joseph Farrell & Garth Saloner, *Standardization, Compatibility, and Innovation,* 16 RAND J. ECON. 70 (1985).

133. *See* Churbuck & Freedman, *supra* note 15, at 1 (noting that " '[t]he real cost of software is not in the package but in the price of training' " (quoting Wayne Maples, information-center consultant at the Federal Reserve Bank in Dallas)); *Interview: Apple Computer, Inc., President and Chairman John Sculley—On Fitting into the IBM World of Computing,* PERS. COMPUTING, Apr. 1986, at 145, 147 ("It's becoming apparent that the real cost is not the hardware or even the software. The real cost is teaching the user.").

134. *See* D. HEMENWAY, *supra* note 119, at 30, 39; Joseph Farrell & Garth Saloner, *Installed Base and Compatibility: Innovation, Product Preannouncements, and Predation,* 76 AM. ECON. REV. 940, 940 (1986); Farrell & Saloner, *supra* note 132, at 71-72, 75-79. As this literature points out, it is not necessarily efficient for all technological improvements in standards to be adopted immediately because of the costs of replacing the existing stock of products based upon the older standard. Thus, if the new product offers only minor advances but would require the obsolescence of a large stock of valuable installed capacity, it is better for society to stick with the older standard until a significant improvement is available or to replace existing stock only as it wears out. The process by which standards are supplanted depends largely on whether the transition can be made gradually or must occur all at once.

135. Investigators cite the persistence of the QWERTY typewriter keyboard as an example of this phenomenon. *See* notes 120-124 *supra* and accompanying text. Fearing the potentially adverse effects of entrenched standards on product innovation in the computer industry, the National Bureau of Standards declined to set interface standards for computer hardware in the early 1970s. *See* Lecture by Ruth Davis, Center for Computer Sciences and Technology, National Bureau of Standards, at Harvard University (April 1972), *cited in* D. HEMENWAY, *supra* note 119, at 39.

136. *Cf.* J. BAIN, *supra* note 77, at 216 (noting that "the advantage to established sellers accruing from buyer preferences for their products as opposed to potential-entrant products is on the average larger and more frequent . . . than any other barrier to entry").

ticularly important for smaller, less well-known firms than for established industry leaders.

These advantages of affording legal protection for computer-human interfaces must be balanced against the adverse effects of legal protection on the realization of network externalities,[137] the socially wasteful investment by competitors in efforts to develop functionally similar though noninfringing interfaces, and the other costs of legal protection noted earlier, including monopoly pricing and adverse effects on sequential innovation.[138]

III. Analysis of the Scope of Legal Protection for Application Program Code

As discussed in Part I, research and development of application programs center around two principal components: the computer-human interface and the program code. And as we saw in Part II, these two features of programs have different implications for the analysis of legal protection. The principal market failure relating to the provision of application program code is the public goods problem, whereas computer-human interfaces invoke both the public goods problem and network externalities. Because the intellectual property laws provide separate protection for application program code and the video displays that the code generates,[139] the distinction between application program code and the computer-human interface carries over to copyright doctrine.

This part analyzes the scope of legal protection for application program code. The first section summarizes the main intellectual property doctrines relating to application program code. The second section applies the economic framework developed in Part II to application program code and critically evaluates the current legal landscape in light of these policy considerations. The third section describes an alternative regime of legal protection which better accommodates policy concerns

137. *See* text accompanying notes 125-130 *supra*.

138. It should be noted that alternative government approaches are possible for addressing network externality concerns, such as having the government set standards directly, *but cf.* note 135 *supra,* relaxing antitrust restraints that limit industries' ability to set voluntary standards, and promoting standardization by using the government's market power as a buyer. This article limits its attention to the role of intellectual property protection in fostering (and undermining) the realization of network externalities.

139. Notwithstanding the availability of separate protection for application program code and screen displays, *see* 1 P. Goldstein, *supra* note 7, § 2.15. (1989) (forthcoming), the Copyright Office recently determined that all copyrightable expression owned by the same claimant and embodied in a single computer program should be registered on a single application form. 53 Fed. Reg. 21,817 (1988). Though recognizing that computer programs can feature both literary and audiovisual works of authorship, *see id.* at 21,817-18, the Copyright Office concluded that combined works must be registered under the class of authorship—*e.g.*, "literary" or "audiovisual"—that predominates in the work, *id.* at 21,818 (to be codified at 37 C.F.R. § 202.3(b)(2)).

and is more consonant with fundamental principles of intellectual property doctrine than the current direction of copyright protection.

A. *Modes of Legal Protection for Application Program Code*

1. *Copyright.*

By traditional views of copyright's proper place within the intellectual property landscape, it is somewhat surprising that CONTU found copyright the most suitable mode of legal protection for computer software.[140] Patent and trade secret law have traditionally been associated with the protection of utilitarian works.[141] Nonetheless, the majority of CONTU was concerned that the doctrinal and practical difficulties of obtaining patent protection[142] and the ease with which trade secret status could be lost[143] prevented these doctrines from providing adequate protection for intellectual work embodied in computer software. The CONTU majority believed that copyright, with its low requirements for protection[144] and lengthy duration,[145] was better suited for protecting software. Thus, the literary form of computer software rather than its utilitarian substance was thought to be the better guide for determining the appropriate form of legal protection.

Viewed in this light, the 1980 copyright law amendments implementing CONTU's recommendations[146] may be seen best as an instrumental decision—though the methods were not intellectually tidy, the results were thought to be worth the inconsistency.[147] Notwithstanding the mismatch, neither CONTU nor Congress intended that the 1980 amendments would change copyright's fundamental nature. In the re-

140. Professor Samuelson notes that numerous efforts during this century to extend copyright-type protection to designs of utilitarian objects have failed. *See* Samuelson, *supra* note 2, at 727-28.

141. *See* 1 P. GOLDSTEIN, *supra* note 7, § 2.15; Paul Goldstein, *Infringement of Copyright in Computer Programs,* 47 U. PITT. L. REV. 1119, 1122-23 (1986); Samuelson, *supra* note 2, at 732-36.

142. On the doctrinal side, the CONTU majority noted that the patent code's requirements of novelty, nonobviousness, and usefulness might be too high for computer software. On the practical side, the CONTU majority believed that the cost and time involved in obtaining patent protection augured against its efficacy. *See* CONTU REPORT, *supra* note 1, at 17.

143. *See id.* at 17-18.

144. Copyright extends to "original works of authorship fixed in [a] tangible medium of expression." 17 U.S.C. § 102(a) (1982). The term "original" has traditionally been interpreted as a minimal restriction on granting copyright protection. *See* 1 P. GOLDSTEIN, *supra* note 7, § 2.2. A software creator satisfies the "fixation" requirement of § 102(a) by storing the program on paper, magnetic tape, magnetic disk, or a semiconductor chip. *Id.* § 2.4.

145. Copyright extends for the life of the author plus 50 years. 17 U.S.C. § 302(a). For works created for hire, the term of protection is 75 years from the date of first publication. *Id.* § 302(c).

146. In 1980, Congress amended § 101 of the Copyright Act of 1976 to define a "computer program" as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." *Id.* § 101. Congress also rewrote § 117 to allow computer program owners to make additional copies (or adaptations) if necessary to use the program or for archival purposes. *Id.* § 117.

147. *Cf.* Karjala, *supra* note 9, at 87.

port explaining its recommendations, CONTU emphasized that only the expressive aspect of software was to be protected under copyright law[148] and that the section 102(b) limitation on protection—denying copyright protection to "any idea, procedure, process, system, method of operation, concept, principle, or discovery"—would be fully operative.[149] By merely amending section 101 of the Copyright Act to define "computer program" and rewriting section 117 to allow program purchasers a limited right to make copies,[150] the 1980 amendments did not in any way limit the operation of section 102(b).

Somewhat surprisingly in light of section 102(b)'s limitation on copyright's subject matter, courts have been extremely receptive, with rare exceptions,[151] to software developers' efforts to obtain broad copyright protection for their works. In the first generation of software infringement cases, which focused on whether and to what extent copyright prohibits literal copying of computer software, the courts determined that copyright protection extends to both source code and object code,[152] whether stored in read only memory (ROM), semiconductor chips, or on disks.[153] In *Apple Computer, Inc. v. Franklin Computer Corp.*,[154] perhaps the most important early case, the Third Circuit held that copyright law protects the precise coding of a computer operating system and that there is no merger of idea and expression as long as "other methods of expressing [the idea underlying the operating system program] are not foreclosed as a practical matter."[155] A critical attribute of the operating system program at issue in *Apple* was that its particular coding was essential to running a wide variety of application programs available on the market. In addressing the relevance of this fact, the court emphasized that "[the defendant] may wish to achieve total compatibility with independently developed application programs written for [the plaintiff's computer system], but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged."[156]

More recently, courts have begun to confront a second generation

148. *See* note 7 *supra* (discussing the idea/expression distinction).

149. *See* CONTU REPORT, *supra* note 1, at 18-23.

150. *See* note 146 *supra*.

151. *See* Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., 807 F.2d 1256 (5th Cir. 1987); Synercom Technology Inc. v. University Computing Co., 462 F. Supp. 1003 (N.D. Tex. 1978) (denying copyright protection of formats for inputting data on the ground that they are ideas and not expression).

152. *See* note 27 *supra*.

153. *See* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983); Williams Elec., Inc. v. Artic Int'l, Inc., 685 F.2d 870 (3d Cir. 1982).

154. 714 F.2d 1240 (3d Cir. 1983).

155. *Id.* at 1253.

156. *Id.*; *see also* Apple Computer, Inc. v. Formula Int'l Inc., 725 F.2d 521 (9th Cir. 1984); *cf.* Menell, *supra* note 9, at 1362-63 (questioning the reasoning underlying this interpretation of the merger doctrine).

of software copyright cases alleging infringement of protected application program code on the basis of nonliteral emulation—duplication of a program's organization without literally copying any significant portions. In *Whelan Associates v. Jaslow Dental Laboratory*,[157] a leading case of this nascent generation, the Third Circuit held that "copyright protection of computer programs may extend beyond the programs' literal code to their structure, sequence and organization."[158] The most notable aspect of the Third Circuit's decision, however, was its extremely broad view of what should be considered expression within the structure, sequence, and organization of application program code. In addressing whether section 102(b)'s prohibition of copyright for ideas, procedures, processes, systems, methods of operation, concepts, principles, or discoveries[159] applied to the structure of the plaintiff's program, the court reasoned that "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression."[160] By defining the purpose of the plaintiff's program as the "efficient organization of a dental laboratory" and finding that "there are a variety of program structures through which that idea can be expressed,"[161] the court concluded that the particular means chosen were not necessary to the purpose and therefore were protectible expression and not merely an idea. While one court has, in dicta, questioned the *Whelan* formulation,[162] courts deciding the merits of second generation cases have adhered to the *Whelan* approach and extended it to application program screen displays.[163]

2. *Patent.*

Patent protection is available for "any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof"[164] that is not obvious.[165] Unlike copyright, which protects only against substantially similar copies of the copyrighted expression,[166] patent protection gives the inventor the right to exclude others from making, using, or selling the patented art.[167]

157. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987).
158. *Id.* at 1222.
159. *See* text accompanying note 149 *supra*.
160. 797 F.2d at 1236 (emphasis omitted).
161. *Id.* at 1240.
162. *See* Plains Cotton Coop. Ass'n v. Goodpasture Computer Serv., 807 F.2d 1256, 1262 (5th Cir. 1987).
163. *See* Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 461 (N.D. Ga. 1987); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986). These cases are discussed at text accompanying notes 246-252 *infra*.
164. 35 U.S.C. § 101 (1982).
165. *Id.* § 103; *see also* Graham v. John Deere Co., 383 U.S. 1 (1966).
166. *See* 3 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.01(b), at 13-6 (1988).
167. *See* Baker v. Selden, 101 U.S. 99, 104 (1879); Michael J. Kline, *Requiring an Election of*

Thus, a patent can be infringed by any device that accomplishes the same result in the same manner as the patented material,[168] or even by another inventor's independent creation of the same device.[169] So as not to hinder technological advancement unduly, the Supreme Court has excluded from the scope of patent protection certain fundamental building blocks of science such as laws of nature,[170] scientific principles,[171] and mathematical algorithms[172] on the ground that they are "too important to be subjected to private control."[173]

The Patent and Trademark Office initially took the view that computer programs were not patentable because they could be characterized as sequences of mental steps and/or mathematical algorithms.[174] Thus, the CONTU majority quite reasonably had grave doubts about the availability of patent protection for computer programs.[175] The Supreme Court's 1981 decisions in *Diamond v. Diehr*[176] and *Diamond v. Bradley*,[177] however, cleared the way for patent protection for computer programs.[178]

Since these decisions, there has been renewed interest in obtaining

---

Protection for Patentable/Copyrightable Computer Programs: Part II, 67 J. PAT. & TRADEMARK OFF. SOC'Y 339, 371 (1985).

168. *See* Graver Tank & Mfg. Co. v. Linde Air Prods. Co., 339 U.S. 605 (1950) (doctrine of equivalents).

169. Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 478 (1974); 4 DONALD S. CHISUM, PATENTS § 16.02(2), at 16-19 (1987).

170. *See* Dennis v. Pitner, 106 F.2d 142, 146 (7th Cir.), *cert. denied*, 308 U.S. 606 (1939); *cf.* Eibel Process Co. v. Minnesota & Ont. Paper Co., 261 U.S. 45, 52 (1923) (suggesting law of gravity is not patentable).

171. O'Reilly v. Morse, 56 U.S. (15 How.) 61, 117 (1853).

172. MacKay Radio & Tel. Co. v. Radio Corp. of Am., 306 U.S. 86, 94 (1939). *But cf.* Donald S. Chisum, *The Patentability of Algorithms*, 47 U. PITT. L. REV. 959 (1986) (arguing that novel and nonobvious algorithms should be patentable subject matter); Michael Gemignani, *Should Algorithms Be Patentable?*, 22 JURIMETRICS J. 326 (1982) (questioning the desirability of the traditional doctrine prohibiting patentability of algorithms). Notwithstanding the traditional doctrine, the Patent and Trademark Office has recently allowed patents for inventions comprised principally of new algorithms. *See* Edmund L. Andrews, *Equations Patented; Some See Danger*, N.Y. Times, Feb. 15, 1989, at D1, col. 3.

173. PAUL GOLDSTEIN, COPYRIGHT, PATENT, TRADEMARK AND RELATED STATE DOCTRINES 525 (2d ed. 1981).

174. *See* Nelson Moskowitz, *The Metamorphosis of Software-Related Invention Patentability*, 3 COMPUTER/L.J. 273, 282 (1982). It has also been suggested that bureaucratic concerns about processing a deluge of software applications, *see* Gregory J. Maier, *Software Protection—Integrating Patent, Copyright and Trade Secret Law*, 28 IDEA 13, 14-15 (1987), and political crossfire between vested interests, *see* Davidson, *supra* note 52, at 349, influenced the formulation of the Patent and Trademark Office's early policy disfavoring patentability of computer software.

175. *See* CONTU REPORT, *supra* note 1, at 37.

176. 450 U.S. 175 (1981).

177. 450 U.S. 381 (1981).

178. The Supreme Court had denied patent protection to a program that implemented a mathematical algorithm because it did not feature nonobvious post-solution activity (actual physical action following the execution of a mathematical formula). Parker v. Flook, 437 U.S. 584 (1978). This would seem to limit significantly the availability of patent protection for application programs because few of them feature nonobvious post-solution activity. The decision in *Diamond v. Diehr*, however, largely eroded this requirement. *See* 450 U.S. at 209-10 (Stevens, J., dissenting); Davidson, *supra* note 52, at 351-53; Maier, *supra* note 174, at 16-19.

patents for application programs.[179] The Patent and Trademark Office has implemented procedures for reviewing software applications[180] and has already issued many such patents. Among application program code patents the Office has issued are: a spell-checking routine,[181] a program that converts RPG programming language into COBOL programming language,[182] a program that implements a compiling technique,[183] a program that operates a brokerage cash management system,[184] and a system for detecting and correcting contextual errors in a text processing system.[185]

Nonetheless, patent law is not seen as a viable option for the protection of most application program code. Most programs, though products of significant efforts to define, outline, and implement a method for performing tasks on a computer, simply do not manifest sufficient novelty or nonobviousness to merit patent protection.[186] Even for those programs that are novel and nonobvious, the time and cost of obtaining protection may not be worth the effort, particularly if the product is not expected to have a long life cycle.[187] Furthermore, to obtain a patent, the inventor must disclose the art to the Patent Office and ultimately to the public.[188] The inventor might very well wish not to do this, for such disclosures would facilitate access by others to the inner workings of the patentee's invention and would destroy any trade secrecy claim.[189]

---

179. *See* Maier, *supra* note 174, at 19; John P. Sumner & Steven W. Lundberg, *The Versatility of Software Patent Protection: From Subroutines to Look and Feel,* COMPUTER LAW., June 1986, at 1, 5; *see also* ABA Comm. on Computer Software, Res. 406-3 (1986) (calling for the application of traditional patent standards to software-related patents).

180. PATENT & TRADEMARK OFFICE, U.S. DEPARTMENT OF COMMERCE, MANUAL OF PATENT EXAMINING PROCEDURE § 2106 (5th rev. ed. 1988).

181. 1023 Off. Gaz. Pat. Office 1075, U.S. Pat. No. 4,355,371 (1982) (assigned to IBM).

182. 1027 Off. Gaz. Pat. Office 1244, U.S. Pat. No. 4,374,408 (1983) (assigned to Burroughs).

183. 1024 Off. Gaz. Pat. Office 363, U.S. Pat. No. 4,309,756 (1982) (granted to Beckler).

184. 1021 Off. Gaz. Pat. Office 1498, U.S. Pat. No. 4,346,442 (1982); 1028 Off. Gaz. Pat. Office 646 (1983) (both assigned to Merrill Lynch); *see also* David J. Meyer, *Patentability of Business Methods Implemented by Computer,* COMPUTER LAW., Feb. 1985, at 12,13; David J. Meyer, Paine, Webber, Jackson and Curtis, Inc. v. Merrill Lynch, Pierce, Fenner & Smith: *Methods of Doing Business Held Patentable Because Implemented on a Computer,* 5 COMPUTER/L.J. 101 (1984) (student author); Lynne B. Allen, *The Patentability of Computer Programs: Merrill Lynch's Patent for a Financial Services System,* 59 IND. L.J. 633 (1984) (student author).

185. 1079 Off. Gaz. Pat. Office 1758, U.S. Pat. No. 4,674,065 (1987) (assigned to IBM).

186. *See, e.g.,* 1 DAVID BENDER, COMPUTER LAW § 3A.02, at 3A-6.1 (1988) (claiming that probably well over 90% of computer programs are not patentable); Davidson, *supra* note 52, at 357 (asserting that perhaps less than 1% of computer programs are patentable).

187. The uncertainty of obtaining protection—which is significant with regard to most software-related inventions—would also enter into the inventor's decision whether to seek patent protection.

188. *See* 35 U.S.C. §§ 111-112, 114, 154 (1982) (describing disclosure requirements); *see also* THORNE D. HARRIS, THE LEGAL GUIDE TO COMPUTER SOFTWARE PROTECTION 130 (1985).

189. *See* Q-Co Indus. v. Hoffman, 625 F. Supp. 608, 617 (S.D.N.Y. 1985).

3. *Other modes of legal protection.*

In addition to copyright and patent protection, a number of other forms of legal protection may enable computer software developers to appropriate the value of their efforts. In varying circumstances and to different degrees, trade secret law, misappropriation doctrine, trademark protection, and the Semiconductor Chip Protection Act aid software developers in preventing competitors from exploiting intellectual work embodied in computer software.

Trade secret law. As discussed in Part II, the public goods problem associated with intellectual work is alleviated to the extent inventors can prevent disclosure of their creations through contractual arrangements. Trade secret law, as developed in the common law and codified in state statutes,[190] governs the availability and extent of this type of protection. Although trade secret law varies from state to state, a widely adopted definition of a trade secret is "any formula, pattern, device or compilation of information which is used in one's business, and which gives him an opportunity to obtain an advantage over competitors who do not know or use it."[191] To obtain trade secret protection, a business must establish that its information is novel, valuable in the trade or business, and secret.[192]

There are a number of legal and practical impediments to establishing, maintaining, and enforcing trade secret protection for application program code. Although absolute secrecy is not required to obtain trade secret protection,[193] the secrecy element typically requires that contractual or other means be used to restrict dissemination of the valuable information.[194] Manufacturers of mass-marketed software products have attempted to satisfy this requirement by using "shrink-wrap"

190. *See, e.g.*, CAL. CIV. CODE §§ 3426-3426.10 (West Supp. 1989); CONN. GEN. STAT. §§ 35-50 to -58 (Supp. 1987); *see also* CAL. PENAL CODE § 499(c) (West 1988) (making trade secret theft a criminal offense).

191. RESTATEMENT OF TORTS § 757 comment b (1939) (withdrawn 1977). The reporters of the Second Restatement decided that trade secret law more properly belongs in a separate field of law relating to unfair competition and trade regulation. *See* 4 RESTATEMENT (SECOND) OF TORTS 1-2 (1979). The first Restatement's definition continues to be widely cited. *See* RESTATEMENT (SECOND) OF TORTS app. § 757 (1977 & Supp. 1987-88) (collecting cases).

192.

> The subject matter of a trade secret must be kept secret. . . . [A] substantial element of secrecy must exist, so that, except by the use of improper means, there would be difficulty in acquiring the information. . . . Some factors to be considered in determining whether given information is one's trade secret are: (1) the extent to which the information is known outside of his business; (2) the extent to which it is known by employees and others involved in his business; (3) the extent of measures taken by him to guard the secrecy of the information; (4) the value of the information to him and to his competitors; (5) the amount of effort or money expended by him in developing the information; (6) the ease or difficulty with which the information could be properly acquired or duplicated by others.

RESTATEMENT OF TORTS, *supra* note 191, § 757 comment b.

193. *See Hoffman*, 625 F. Supp. at 617.

194. *See, e.g., id.* (noting that the secrecy element requires "only that a 'substantial element of secrecy must exist and this means so much that' except by use of improper means,

and similar licenses, though there is substantial doubt as to the enforceability of such licenses.[195] Even if trade secret protection is legally effective, it is difficult for licensors of computer software, particularly mass-marketed programs, to monitor and enforce such agreements.[196] Consequently, trade secret protection typically is feasible only for programs written for mainframes and minicomputers.

Misappropriation doctrine. The misappropriation doctrine, a branch of unfair competition law, might also protect some aspects of intellectual work embodied in application program code. In the landmark case of *International News Service v. Associated Press*,[197] the Supreme Court held that the gatherer of valuable information has a limited right to prevent competitors from copying such information.[198] This "quasi-property" interest may provide limited protection for certain aspects of application program code, such as collections of information—*e.g.*, dictionaries, addresses—gathered by the creator and embodied in a program. Federal preemption doctrine, however, confines the scope of misappropriation doctrine to matters not within federal copyright and patent statutes.[199] Conversely, the scope of misappropriation doctrine is expanded to the extent that courts narrowly interpret the range of subject matter that falls within the copyright and patent statutes.

Trademark law. Trademark law, a different branch of unfair competition law, protects the marks manufacturers use to identify their products to help consumers avoid confusion regarding the products' source.[200] This form of protection is particularly significant in the computer software market because a reputation for reliability and quality facilitates customer acceptance of new products and because consumers value compatibility among complementary computer products.[201]

Semiconductor Chip Protection Act. In 1984, Congress enacted legislation establishing a new category of legal protection for original, noncommonplace computer programs embodied in semiconductor chips (mask works).[202] The Semiconductor Chip Protection Act

---

there would be difficulty in acquiring the information" (quoting A.H. Emery Co. v. Marcan Prods. Corp., 389 F.2d 11, 16 (2d Cir.), *cert. denied*, 393 U.S. 835 (1968)).

195. *See* note 83 *supra* and accompanying text. There is also concern that this form of protection is preempted by the Copyright Act of 1976. *See* Menell, *supra* note 9, at 1352; note 84 *supra* and accompanying text.

196. *See* Gilburne & Johnston, *supra* note 83, at 255-63.

197. 248 U.S. 215 (1918).

198. *See generally* Douglas S. Baird, *Common Law Intellectual Property and the Legacy of* International News Service v. Associated Press, 50 U. CHI. L. REV. 411 (1983).

199. *See, e.g.*, United States Golf Ass'n v. St. Andrews Sys., 749 F.2d 1028, 1034-41 (3d Cir. 1984); Addressograph-Multigraph Corp. v. American Expansion Bolt & Mfg. Co., 124 F.2d 706 (7th Cir. 1941), *cert. denied*, 316 U.S. 682 (1942); Synercom Technology, Inc. v. University Computing Co., 474 F. Supp. 37 (N.D. Tex. 1979); *see also* Baird, *supra* note 198, at 421-23.

200. *See* 15 U.S.C. §§ 1051-1127 (1982 & Supp. 1985). *See generally* J.T. MCCARTHY, TRADEMARKS AND UNFAIR COMPETITION (2d ed. 1984).

201. *See* note 118 *supra*.

202. Semiconductor Chip Protection Act of 1984, 17 U.S.C. §§ 901-914 (Supp. 1985).

(SCPA) supplements the various forms of legal protection described above (including copyright and patent) and protects application programs fixed in semiconductor chips[203] for a 10-year term.[204] To promote rapid sequential innovation in the computer software field, Congress expressly authorized competitors to engage in reverse engineering[205] for the purpose of developing improved versions of mask works.[206]

B. *Evaluation of Current Protection for Application Program Code*

This section first applies Part II's general economic framework in analyzing legal protection for application program code. On the basis of this analysis, it then evaluates the current regime of legal protection for application program code. As a result of the ease with which copyright protection can be obtained (relative to patent, trade secret, and other forms of legal protection) and the expansive interpretations of its scope, copyright has emerged as the predominant form of legal protection for application program code. The analysis below, therefore, focuses on evaluating the direction of copyright protection for application program code. It is important to keep in mind, however, that the other forms of legal protection discussed above remain available regardless of the scope of copyright protection.

1. *Economic analysis of legal protection for application program code.*

In terms of the economic framework for analyzing legal protection discussed in Part II, the principal market failure with regard to the provision of application program code is the public goods problem. There may also be network externalities attributable to the benefits of establishing standardized programming practices.[207] One can draw the following conclusions from applying this economic framework to legal protection for application program code. As discussed above, the case for granting intellectual property rights for application programs for mainframes and minicomputers is notably weak,[208] for manufacturers can recover research and development costs through updating and maintenance contracts and can preclude nonpurchasers from gaining access to their programs through licensing provisions prohibiting disclosure and copying. To the extent such mechanisms and the extensive public subsidies of computer science research[209] are inadequate to ad-

203. *See, e.g.*, E.F. Johnson v. Uniden Corp. of Am., 623 F. Supp. 1485, 1489 (D. Minn. 1985) (computer program for operating a two-way communication system fixed in a semiconductor chip).

204. 17 U.S.C. § 904(b).

205. Reverse engineering refers to the disassembling of a product for the purpose of discovering how it works.

206. 17 U.S.C. § 906(a).

207. *See* text accompanying notes 55-57 *supra*.

208. *See* text accompanying notes 78-79 *supra*.

209. *See* text accompanying notes 100-114 *supra*.

dress the public goods problem with regard to mainframe and minicomputer application program code, the general analysis developed below would apply.

The public goods problem is most troublesome with regard to application program code for microcomputers. The ease and low cost of copying programs, the infeasibility of licensing restrictions, and the drawbacks to technological anticopying devices give rise to the potential for significant "leakage" of intellectual work to nonpurchasers. Although first-mover advantages, licensing, and government and private subsidies for research and development offset this problem to some extent, intellectual property protection can still play a significant role in improving the provision of new and better application program code.

The dilemma in designing legal protection for application program code is finding the correct balance between providing programmers with sufficient lead time to exploit their programs in the market before imitators appear, and not inhibiting the rapid sequential process driving the technological advancement of application programming.[210] At a minimum, economic analysis calls for protection against literal copying of significant portions of application program code. Otherwise, an imitator could immediately enter the first programmer's market at much lower product development costs. This prospect would discourage innovation and reward waiting. On the other hand, protecting code much beyond prohibiting literal copying would impose significant monopoly costs and inhibit the development of subsequent innovations. When providing enhanced legal protection encourages important improvements that would be long in coming in its absence, the monopoly costs might be justified. But when the principal effect of furnishing legal protection is to exclude others from building on a readily apparent outgrowth of existing technology, the costs of protection clearly outweigh its benefits. Thus, the latter half of the dilemma calls for significant legal protection beyond literal copying only when valuable new programs would not be forthcoming in its absence. Since valuable means "useful" in this context[211] and obvious programs will be produced even in the absence of legal protection (to the extent demand exists for such products), the dilemma's resolution corresponds roughly to the basic threshold requirements of patent law—novelty, usefulness, and nonobviousness.

The discussion of the economics of microcomputer application pro-

210. *See* text accompanying notes 58-60 *supra*; *cf.* R. Levin, *supra* note 77, at 788 (noting that "[t]he semiconductor industry of the 1950s and 1960s provides an excellent example of rapid progress in a cumulative technology that might have been impossible under a regime that strongly protected intellectual property").

211. As noted earlier, the user does not see application program code, except to the extent it contains messages displayed on a video screen. *See* note 52 *supra*. Thus, its value to the user is its usefulness—its ability to perform desired tasks.

gramming in Parts I and II furnishes guidance for determining how far intellectual property rights should extend beyond protection against literal copying. The relatively low cost of developing microcomputer application programs, the short product life cycle of most programs, and the advantages of introducing the product first suggest that the lead time needed to recover development costs (adjusted for risk) may be relatively short. Furthermore, trademark law protects indefinitely the reputation gains from introducing the product first. Although the actual lead time needed will depend upon how long it takes imitators to reverse engineer the original program, the complexity of most application programs, and the fact that programs are released only in object code[212] ensure that the lead time will be significant (compared to the product life) even if intellectual property rights protect only against literal copying.[213] Thus, the need for significant protection for application program code beyond protection against literal copying is not compelling.

The costs of extensive protection further suggest that the line should be drawn close to literal copying. In addition to the direct monopoly costs of extensive protection for application program code, it is likely that extensive protection would substantially inhibit the dynamic sequential innovation of the microcomputer application program mar-

212. *See* Glenn J. MacGrady, *Protection of Computer Software—An Update and Practical Synthesis,* 20 HOUS. L. REV. 1033, 1065 n.148 (1983) (discussing the complexity involved in decompiling object code into a form that enables analysis of a program's structure).

213. One commentator baldly asserts that reverse engineering of application programs can be done too quickly and inexpensively to enable an originator to recover his or her development costs if intellectual property rights do not prohibit others from inspecting the originator's program in designing their own. Duncan M. Davidson, *Common Law, Uncommon Software,* 47 U. PITT. L. REV. 1037, 1097-98 & n.152 (1986). Without citing any empirical data or other authority, he claims:

> The major cost [of software] is development, not production. A functional duplicator can immediately start selling with a marginal cost of production (the cost of the next copy sent to a customer) as low as that of the originator, and with development costs on the order of 1/10th the magnitude. Thus, the originator is suddenly behind, trying to catch up because it has to repay its higher development costs. Precluding reverse engineering and forcing a duplicator to write its program in its own way, from scratch, simply redresses an economic imbalance.

Id. at 1098 n.152. This reasoning, however, exhibits inaccuracy and analytical shortcomings. First, the data available indicate that research and development costs for microcomputer software are approximately the same as production costs (15% of total revenues). *See* text accompanying note 117 *supra.* Second, even though it might take less time to reverse engineer an application program than, say, an automobile, the relevant factor is how long it takes relative to the product life span. Pre-introduction product marketing plus a few months' lead time will enable the originator to recoup a significant portion of his or her development costs in the market. Whether this is enough to overcome the public goods concern *is* the relevant question. Third, there is little reason to believe that a complex application program can be reverse engineered immediately or at *de minimis* cost, particularly if the program is released only in object code. *See* note 212 *supra.* Fourth, by the time the duplicator enters the market, the originator will have already established a valuable reputation and may have made significant progress in developing a more advanced product that will supplant the market for the original product. Fifth, and possibly most important, it is always necessary to consider the costs of expansive intellectual property protection before concluding that a particular legal regime properly "redresses an economic imbalance."

ket. While licensing might overcome some of these inhibiting effects, the transaction costs of licensing as well as the potential for anticompetitive refusals to license[214] would choke off some socially desirable innovation. Consideration of the range of economic factors strongly suggests that legal protection for application program code should not extend much, if at all, beyond protection against literal copying, except for new, useful, and nonobvious improvements.

2. *Evaluation of the current direction of legal protection for application program code.*

While the proper location of the line between the two horns of the dilemma described above is not easy to determine precisely for all cases, it is apparent that the emerging interpretation of copyright protection for application program code is quite far off the mark. Under the *Whelan* test, copyright extends to the structure, sequence, and organization of application code as long as the overall purpose of the entire program, broadly defined, can be implemented in another manner. Since many aspects of application program code are dictated by basic principles of software engineering,[215] the *Whelan* rule makes it difficult for others wishing to market programs performing the same task as the first comer to perform it as effectively. In effect, the *Whelan* test enables first comers to "lock up" basic programming techniques as implemented in programs to perform particular tasks. This gives the first comer significant market power. In addition, since the programmer did not have to contribute novel and nonobvious programming techniques to our state of knowledge in order to get this legal protection, the monopoly power bestowed will not, in most cases, be justified on the basis of sound public policy analysis.

Furthermore, in light of the importance of sequential research in the application program industry, the *Whelan* test is likely to affect detrimentally the direction and costs of research and development. This can be seen clearly by analyzing the set of options open to an application programmer who wishes to improve on an existing program that employs nothing more than good programming practice. In a world protecting only useful, novel, and nonobvious programming elements, the programmer would be free to build upon the functional aspects of the first program. The *Whelan* test, however, prohibits the programmer from innovating in this way. The programmer would either have to: (1) secure a license for the original program (or its desired aspects) from the copyright owner; (2) circumvent the copyrighted material by expressing the desired functional aspects of the original program in a way that does not infringe its overall structure, sequence, and organization; or (3) independently develop the functional programming

214. *See* text accompanying notes 129-130 *supra*.
215. *See* text accompanying notes 51-57 *supra*.

techniques.[216]

Plainly, all three of these options substantially raise the costs of innovative activity. Under the first option, the programmer must expend costs trying to negotiate a licensing agreement with the first program's owner. Rights to research involving new technology can be costly to negotiate because of the difficulty of conveying the importance of an innovation without disclosing the content. Furthermore, the copyright owner might wish to discourage other firms from entering its market with improved products.[217] Under the second option, the programmer has the difficult task of specifically developing a program to perform a particular task in a perverse manner: The second comer is limited to using efficient structure where the original program used inefficient structure, and inefficient structure where the original program is efficient. The third option, independent development, will typically require the programmer to establish expensive "clean room" procedures so that she can later prove that she did not rely upon the original.[218]

In addition to these effects of broad copyright protection on the costs of innovation, the second and third options make clear that copyright redirects inventive activity away from productive research and development (improving the original program) toward wasteful, duplicative research (inventing around the original program's functional aspects or reinventing them in a complex and costly work environment).

By contrast, the patent system is less likely to extend protection to too many computer programs. The patent system's threshold requirements for protection—novelty, utility, and nonobviousness—are better tailored than the copyright standard to rewarding only those innovations that would not be forthcoming without protection. In light of the rapid advancement rate in the computer software field, however, the patent system's 17-year term of protection may prove to be too long.[219] On the other hand, because of costs and delay,[220] the patent system might not provide adequate protection for many application programs—notably those that require substantial effort but do not rise to the patent system's more exacting thresholds.

---

216. Independent creation is a complete defense to copyright infringement. Fred Fisher, Inc. v. Dillingham, 298 F. 145, 150-51 (S.D.N.Y. 1924) (Learned Hand, J.).

217. *See* note 130 *supra* and accompanying text.

218. *See* G. Gervaise Davis, *IBM PC Software and Hardware Compatibility,* COMPUTER LAW., 11, 16-17 (July 1984) (describing clean room procedures); Karjala, *supra* note 59, at 63 n.35 (noting that, in situations in which a second comer wants to ensure compatibility with an original program, independently developing a program might be more costly than developing the original system).

219. *See generally* YALE M. BRAUNSTEIN, DIETRICH M. FISCHER, JANUSZ A. ORDOVER & WILLIAM J. BAUMOL, ECONOMICS OF PROPERTY RIGHTS AS APPLIED TO COMPUTER SOFTWARE AND DATA BASES IV-1 to IV-58 (1977) (report prepared for CONTU analyzing the optimal duration of legal protection for computer software).

220. *See* text accompanying notes 186-188 *supra.*

C. *Toward a Better Standard of Copyright Protection for Application Program Code*

This section describes an alternative regime of legal protection for application program code that both accommodates the concerns brought out by the economic analysis and is more consonant with the fundamental principles of intellectual property protection than the *Whelan* approach. The *Whelan* court correctly recognized that drawing the line between idea and expression is informed by "the purpose of the copyright law . . . to create the most efficient and productive balance between protection (incentive) and dissemination of information, to promote learning, culture and development."[221] The court's approach did not, however, pay adequate attention to the implications of broad protection for the structure, sequence, and organization of application program code.[222] Perhaps more significantly with regard to fundamental copyright principles, the *Whelan* court naively reasoned that because a function could be performed in more than one way, its structure, sequence, and organization is expressive and therefore copyrightable. A few simple examples bring out the inconsistency of this reasoning with well-established copyright principles. Under the *Whelan* approach, a culinary writer would have an arguable claim that his cookbook's structure based upon the order of courses in a meal—*e.g.*, Chapter 1 - Appetizers, Chapter 2 - Soups, Chapter 3 - Salads,[223] Chapter 4 - Entrees, Chapter 5 - Desserts—is copyrightable because the function—a book expressing recipes—can be expressed in other ways. Similarly, a lexicographer could seek protection for a dictionary in alphabetical order because surely there are other ways of writing a book that defines words (though it is doubtful that anyone would want to own it);[224] and a historian could seek copyright protection for chronological presentation of a particular period.[225]

---

221. Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1235 (3d Cir. 1986). The Third Circuit noted further that "[a]chieving the proper incentive has been a longstanding task of courts." *Id.* at n.27. The court then quoted Lord Mansfield in Sayre v. Moore, 102 Eng. Rep. 138, 140 n.6 (1785):

> [W]e must take care to guard against two extremes equally prejudicial; the one, that men of ability, who have employed their time for the service of the community, may not be deprived of their just merits, and the reward for their ingenuity and labour; the other, that the world may not be deprived of improvements, nor the progress of the arts be retarded.

Id.

222. *Cf.* Goldstein, *supra* note 9, at 1125-26 (suggesting that the *Whelan* court interpreted the copyright concept of "idea" too literally); David Ladd & Bruce G. Joseph, *Expanding Computer Software Protection by Limiting the Idea,* 2 J.L. & TECH. 5, 10 (1987) (criticizing the *Whelan* court for failing to consider the costs of a broad rule).

223. Apologies to continental diners.

224. *See* Denicola, *supra* note 66, at 527. *But cf.* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1252-53 (3d Cir. 1983) (finding that in order for an idea to be merged with its expression, other methods of expressing the idea must be "foreclosed as a practical matter" and that a mere "commercial and competitive objective" would not satisfy this test).

225. This argument has, of course, been soundly rejected. "[S]ince the narration of

While these examples are admittedly hyperbolic, they highlight the key flaw in the *Whelan* court's test. Merely asking the question, "Does another way exist to perform a particular function?," does not resolve the idea/expression puzzle. Some ways are better than others. And with respect to computer programs, "better" typically means more efficient—*e.g.*, faster execution speed, less memory utilization, compatibility with data storage methods. The *Whelan* test freely protects efficient organization whenever other means of accomplishing a task exist, even if they are inferior (in terms of various efficiency measures). Ordinarily, such protection is available only after an inventor has met the patent system's exacting standards. Even then, the protection lasts only seventeen years rather than the life of the inventor plus fifty years (or seventy-five years in the case of works for hire).[226]

The better approach is to limit copyright protection to the expressive aspects of structure, sequence, and organization, where "expressive" means "not related to enhancing computing efficiency" or, more generally, not serving the program's functional attributes. Stated inversely, this approach assumes—quite plausibly, as Part I explains—that an implicit aspect of all application program code is computing efficiency. While efficiency measures may vary across programs depending on the particular programming objectives, the range of efficiency goals is generally clear: faster processing speed, good programming practice (as a means of minimizing debugging problems), efficient memory capacity utilization, and rapid, accurate information transmission across interfaces. By carefully dissecting a computer program's structure, a court (with the aid of experts) would be able to separate the program's functional (or efficiency-enhancing) aspects from its nonfunctional or expressive aspects.

This approach is not chosen as a means of encouraging the creation of expressive application program code. As discussed earlier,[227] application program users do not care how the code underlying the programs is expressed. Rather, the proposed approach makes sense only by reference to the instrumental purposes underlying CONTU's recommendation to extend copyright protection to computer software. Copyright was thought most appropriate because its threshold requirements are better suited to the nature of innovation in the software field.[228] Even though the "value" of software is in its functional attributes, CONTU was clear that copyright protection for software should

history must proceed chronologically—or at least, such is the convention—the order in which the facts are reported must be the same in the case of a second supposed author. There cannot be any such thing as copyright in the order of presentation of the facts . . . ." Myers v. Mail Express Co., 39 Copy. Dec. 478 (S.D.N.Y. 1919) (Hand, J.), *quoted in* 1 M. NIMMER & D. NIMMER, *supra* note 166, § 2.11(D) n.26; *see also* Denicola, *supra* note 66, at 527.

226. There is some question whether even 17 years is too long a term of protection. *See* note 219 *supra* and accompanying text.

227. *See* note 52 *supra* and accompanying text.

228. *See* notes 146-150 *supra* and accompanying text.

not extend beyond the traditional limits established by section 102(b) of the Copyright Act.[229] By protecting expressive aspects of software, copyright law indirectly protects the functional aspects of software by forcing second comers to exercise great care in writing software where others have toiled. At the same time, copyright law, as delimited by section 102(b), does not prevent second comers from making use of functional advances in application programming. Thus, the copyright law's protection of expressive aspects of program code is but a means—by extending lead time—to the end of providing appropriate incentives for creating new and better application programs.[230]

Under this approach to nonliteral copying, the plaintiff would have to prove that the defendant copied nonfunctional aspects of the overall structure, sequence, or organization of the plaintiff's application program code.[231] In addition to establishing substantial similarity with an aspect of plaintiff's overall structure,[232] the plaintiff would have to establish that its program's structure was inefficient or otherwise did not reflect good programming practice at the time defendant produced its program.[233] In assessing the plaintiff's evidence, a court would need to consider the purpose underlying the copyright law.[234] Because of the complexity of deciding how to weigh multiple efficiency criteria, it would probably be sufficient for a court to require a defendant to explain convincingly why the plaintiff's overall structure, sequence, or organization was the most efficient design for achieving the defendant's programming objectives.[235]

---

229. *See* note 149 & text accompanying notes 148-150 *supra*.

230. *Cf.* Denicola, *supra* note 66 (applying a similar approach in analyzing copyright protection of nonfiction literary works).

231. *Cf. In re* Registration and Deposit of Computer Screen Displays, 52 Fed. Reg. 28, 311 (1987) (testimony of William F. Patry before the U.S. Copyright Office) [hereinafter Patry Testimony] (noting "disagreement with the approach taken . . . in *Whelan* . . . under which the [court] found that if there was more than one way to produce the same result (i.e., there was no merger) the work was ipso facto copyrightable." In Patry's opinion, "this approach omits a critical step; whether the nonmerged material is protectible, for what remains may be de minimis, a method of operation, a procedure, a process, a system, or other uncopyrightable material.") (citations omitted).

The proposed test, by limiting the scope of copyright protection for computer programs so as to promote free access to technology, is consistent with the traditional copyright test for predominantly functional works such as architectural plans, business forms, and game rules. *See generally* 1 P. GOLDSTEIN, *supra* note 7, § 2.15.2.

232. *See* 2 P. GOLDSTEIN, *supra* note 7, § 8.5.1.2; Howard Root, *Copyright Infringement of Computer Programs: A Modification of the Substantial Similarity Test,* 68 MINN. L. REV. 1264 (1984) (student author); *see also* note 166 *supra*.

233. A programmer might try to set "traps" for infringers by inserting nonfunctional structural elements into a program, though the costs of such a strategy—in terms of less efficient computing—would have to be weighed against the benefits of identifying infringement.

234. *See* text accompanying notes 140-150 *supra*.

235. The proposed test thus has elements of misappropriation doctrine. *See* text accompanying notes 197-199 *supra*. It requires that second comers make significant efforts to accomplish the program's tasks in the most efficient manner. In this regard, it should not be an excuse that a second comer did not have adequate resources to conduct a reasonable effort to distinguish functional aspects from nonfunctional (expressive) or inefficient aspects of a program.

Perversely, this test requires the plaintiff to present evidence that aspects of its program's organization are inefficient. At the same time, the defendant will seek to show that those aspects of the plaintiff's program are efficient. In addition, this exercise might strain the ability of courts to sort out the complex technological evidence offered.[236] This problem would be lessened somewhat by allowing the defendant some leeway in showing that he or she strove to achieve a particular efficient result, *e.g.*, by finding for the defendant if the defendant could prove that its programmers made reasonable efforts to use efficient functional attributes in their program.

Although the court in *E.F. Johnson Co. v. Uniden Corp. of America*[237] based its analysis on infringement rather than the copyrightability of a particular subject matter, the opinion illustrates how the proposed test could be applied. Plaintiff, E.F. Johnson, owned a copyright in computer software used to operate a logic-trunked radio (LTR) system.[238] To compete in the LTR system market, Uniden developed an LTR system that was compatible with E.F. Johnson's LTR system. This required some similarity in software design;[239] but as the court carefully points out, Uniden went well beyond what was necessary to achieve compatibility. For example, expert evidence showed that E.F. Johnson's computer program used a particular technical parameter (a sampling rate) because of limitations imposed by the microprocessor in its system. Uniden chose the same sampling rate, even though its system used a different microprocessor which would have performed better

236. *See* Karjala, *supra* note 9, at 67. As a way of avoiding the difficult evidentiary problems associated with distinguishing between an application program code's expressive and functional aspects, Professor Karjala recommends limiting the scope of copyright protection for application program code to protection against slavish copying of substantial portions of a program in order to compete directly against the originator's product. *See id.* at 87-88; *see also id.* at 88 (noting that a second comer should be able to translate a program for use with a different hardware system with impunity). Though this approach has some appeal from a policy standpoint, its desirability has not been fully demonstrated. It would be necessary to determine whether direct copying of a program's organization, structure, and sequence would unduly reduce the time required to reverse engineer programs. *Cf.* note 213 *supra* and accompanying text. Furthermore, with regard to translations, such a rule might delay introduction of new programs for one hardware system in order to allow the originator to develop versions for other formats. By contrast, the proposed scheme features many of the attributes of Professor Karjala's scheme while providing greater inducements for second comers to improve upon existing technology, *see* last paragraph of Part III *infra*, without departing from such traditional copyright principles as protecting original, nonfunctional expression.

Professor Karjala's proposal is perhaps better directed toward the legislature than the courts. Since Congress has not specifically identified piracy as the principal basis for legal protection of computer software, as it did for sound recordings, *see* Sound Recording Act of 1971, Pub. L. No. 92-140, 85 Stat. 391; H.R. REP. No. 487, 92d Cong., 1st Sess. (1971), *reprinted in* 1971 U.S. CODE CONG. & ADMIN. NEWS 1566, there does not presently appear to be an adequate statutory basis for applying a pure piracy test for copyright infringement of computer programs.

237. 623 F. Supp. 1485 (D. Minn. 1985).

238. A logic-trunked radio system consists of mobile radio units, typically installed in motor vehicles like taxis and police cars, and base stations which receive and transmit signals to the mobile units. *Id.* at 1487.

239. *Id.* at 1494.

(without any loss of compatibility) with a higher sampling rate.[240] Uniden also copied a mistake that E.F. Johnson's engineer had made in its program.[241] The prevalence of such inefficient design choices led the court to conclude that the plaintiff had a substantial likelihood of success in establishing copyright infringement.[242]

Though the suggested test would no doubt engender evidentiary complexities, it is generally consistent with the conclusions of the economic analysis. The proposed copyright standard is in essence a literal copying test combined with carefully limited protection for structure, sequence, and organization: Structure, sequence, and organization is protected under copyright only to the extent that it is expressive and not functional. A programmer wishing to protect efficiency aspects of application program code, therefore, must turn to the patent system and meet a rigorous standard to ensure that legal protection extends only to useful, novel, and nonobvious elements of program structure. In this way, the proposed test would channel intellectual work into the most appropriate scheme of intellectual property protection.

The proposed standard will have additional desirable effects on research and development of application programs. It encourages firms in the software industry wishing to build upon others' programs to determine which aspects of those programs are efficient and which are not. In effect, this test forces those wishing to use the copyrighted programs of others to refine their structure, sequence, and organization. In this way the proposed test also lengthens the time it takes to reverse engineer a program, giving the first innovator somewhat more lead time to recoup his or her development costs than under a simple literal copying standard.

IV. Analysis of the Scope of Legal Protection for Computer-Human Interfaces

This part analyzes the scope of legal protection for computer-human interfaces. It focuses principally upon legal protection for video displays because of the importance of this mode of computer-human interaction in the present state of interface technology. The first sec-

240. *Id.* at 1494-96.

241. *Id.* at 1495.

242. *Id.* at 1504. The court's analysis of the scope of copyright protection is also interesting. Although it cites the Third Circuit's opinion in Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983), for its analysis of the idea/expression issue, it appears to have viewed system compatibility as idea rather than expression:

> The record amply demonstrates that an LTR-compatible software program could have been written without verbatim duplication of [E.F. Johnson's] version .... Thus, exact duplication ... was not the "only and essential" means of achieving compatibility. ...
>
> [T]he mere fact that defendant set out with the objective of creating an LTR-compatible radio does not, without more, excuse its copying of plaintiff's code.

E.F. Johnson, 623 F. Supp. at 1502-03. *But see Franklin Computer*, 714 F.2d at 1253 (discussed at text accompanying notes 154-156 *supra*).

tion describes the distinctive aspects of legal protection for interfaces. The second section applies the economic framework developed in Part II to computer-human interfaces and critically evaluates the current body of legal protection. The third section suggests a future direction for intellectual property protection of computer-human interfaces that would provide appropriate incentives for the development of improved interfaces while at the same time promoting interface standardization.

A. *Modes of Legal Protection for Computer-Human Interfaces*

Although the basic intellectual property doctrines for computer-human interfaces parallel those for application program code, there are a number of notable differences reflecting the nature of intellectual property law and the unique technological aspects of computer-human interfaces. This section will highlight these differences.

1. *Copyright.*

The principal aspect of computer-human interfaces that comes within the scope of copyright is the video display. Since different application programs can produce the same screen display,[243] copyright protection for the program will not ensure protection of the screen that it generates.[244] Video displays, however, may be protectible separately from the underlying application program code as literary works, pictorial or graphic works, or audiovisual works, depending on their content.[245] Common types of screen displays are menu status screens (which, for example, inform the user of available commands or parameter settings), video game displays, and mathematical tables with designated columns and rows (for example, spreadsheets).

As with application program code, the critical issue in determining the scope of copyright protection for video displays is whether their content is protectible expression or uncopyrightable idea. The principal cases addressing this issue, *Broderbund Software, Inc. v. Unison World, Inc.*[246] and *Digital Communications Associates v. Softklone Distributing Corp.*,[247] have extended the *Whelan* approach to video displays.

The plaintiff in *Broderbund* owned a copyright in an application pro-

243. *See* Stern Elecs., Inc. v. Kaufman, 669 F.2d 852, 855 (2d Cir. 1982) (noting that "many different computer programs can produce the same 'results,' whether those results are an analysis of financial records or a sequence of images and sounds").

244. *See* Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 455 (N.D. Ga. 1987); 1 P. GOLDSTEIN, *supra* note 7, § 2.15.3. *But see* Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1133 (N.D. Cal. 1986) (holding that copyright protection for a computer program "extends to the overall structure of the program, including its audiovisual displays").

245. *See* Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870, 875 (3d Cir. 1982); *Stern Elecs.*, 669 F.2d at 857; 1 P. GOLDSTEIN, *supra* note 7, § 2.15.3; Patry Testimony, *supra* note 231, at 4-9.

246. 648 F. Supp. 1127 (N.D. Cal. 1986).

247. 659 F. Supp. 449 (N.D. Ga. 1987).

gram for designing greeting cards, posters, banners, and signs. The program featured a series of menu-driven video displays that guided the user through the various design steps. The defendant's program, written to run on a different computer system, copied the plaintiff's menu screens, input formats, and screen sequencing. Applying *Whelan*, the *Broderbund* court concluded that since other means of expressing the underlying idea of the plaintiff's program—a system for creating greeting cards, banners, posters, and signs—existed, the "overall structure, sequencing, and arrangement of screens" in the plaintiff's program was protected.[248]

In *Digital Communications*, the plaintiff had written a popular microcomputer communications program[249] called "Crosstalk XVI." The main menu status screen for Crosstalk displays the relevant communications parameters and key settings, as well as a list of the many commands available to the user.[250] To make the rather cluttered screen more comprehensible, the first two letters of each command are capitalized and highlighted (*e.g.*, *SP*eed). In addition, the screen places the most essential information (communications parameters) near the upper left portion of the screen and the less pertinent information (numerous commands) at the bottom. The defendant marketed a microcomputer communications program using a similar status screen configuration. Although the court found that certain aspects of the interface—such as the use of commands, a status screen, particular command terms or symbols, and the highlighting and capitalizing of each command's first two letters—were unprotected ideas,[251] it had little trouble concluding that "the arrangement of the status screen involves considerable stylistic creativity and authorship above and beyond the ideas embodied in the status screen."[252]

248. *Broderbund*, 648 F. Supp. at 1133. The *Broderbund* decision has been roundly criticized for inferring from *Whelan* that the copyright in the application program code extends to the video displays. *See* Digital Communications Assocs. v. Softklone Distrib. Corp., 659 F. Supp. 449, 455 (N.D. Ga. 1987); J. Scott MacKay, Broderbund Software Inc. v. Unison World, Inc.: *"Look and Feel" Copyright Protection for the Display Screens of an Application Microcomputer Program*, 13 RUTGERS COMPUTER & TECH. L.J. 105, 126 (1987) (student author). Notwithstanding this apparent error, the substance of the *Broderbund* court's approach—applying *Whelan*'s idea/expression test to the overall structure, sequence, and organization of screen displays—has been followed. *See Digital Communications*, 659 F. Supp. 449.

The *Broderbund* court also rejected the defendant's argument that plaintiff's screens are uncopyrightable under the "rules and instructions" doctrine. *Broderbund*, 648 F. Supp. at 1134; *see also* Decorative Aides Corp. v. Staple Sewing Aides Corp., 497 F. Supp. 154, 158 (S.D.N.Y. 1980) (applying the doctrine).

249. *See* Jerry Schwartz, *Court Short-Circuits "Clone" Software*, Nat'l L.J., Apr. 27, 1987, at 3, col. 1.

250. The purpose of a communications program like Crosstalk is to allow users to transmit data between different computers. To enable this transmission to occur, the program must ensure that various technical parameters are properly set. The status screen for Crosstalk informs the user of these parameters and lists the numerous commands available for altering the transmission.

251. *Digital Communications*, 659 F. Supp. at 459, 462 (1987).

252. *Id.* at 460. The court rejected the defendant's argument that the status screen was

2. *Patent.*

Patent law is the principal source of protection for the physical devices implementing a user interface. In the past few years, the Patent and Trademark Office has become more receptive to software patents relating to user interfaces. Among the interface utility patents it has issued are a menu system for a word processing program,[253] a system of windowing,[254] a method for assembling documents from an inventory of "parts" accessed from a master screen,[255] a data entry screen for an interactive data entry system,[256] and a touchscreen that emulates three-dimensional objects on a two-dimensional computer screen.[257]

Relying upon a line of cases holding that printed matter on an object is patentable if it is functionally related to the object,[258] some practitioners have argued that a greater variety of interface features are eligible for utility patents.[259] They suggest that "[p]rogram functions or improvements such as editing, control functions, and other user interface features would also be protectible to varying degrees depending upon the novelty of the feature."[260]

More recently, programmers have obtained design patents for aspects of computer-human interfaces.[261] Design patents, which last for fourteen years,[262] are available for a "new, original and ornamental design for an article of manufacture."[263] To be ornamental, a design must be aesthetically appealing and not dictated primarily by functional considerations.[264] In May and June of 1988, the Patent and Trademark

---

not copyrightable because it was merely a "blank form." *Id.* at 460-62. The court also rejected the defendant's argument that the concern for standardization in the computer industry justified the similarity between the plaintiff's and the defendant's status screens. *Id.* at 462.

253. 1013 Off. Gaz. Pat. Office 1936, U.S. Pat. No. 4,308,582 (Dec. 29, 1981) (assigned to IBM).

254. 1060 Off. Gaz. Pat. Office 1845, U.S. Pat. No. 4,555,775 (Nov. 26, 1985) (assigned to AT&T Bell Laboratories).

255. 1088 Off. Gaz. Pat. Office 928, U.S. Pat. No. 4,730,252 (Mar. 8, 1988) (assigned to IBM).

256. 1075 Off. Gaz. Pat. Office 2234, U.S. Pat. No. 4,646,250 (Feb. 24, 1987) (assigned to IBM).

257. 1076 Off. Gaz. Pat. Office 1099-1100, U.S. Pat. No. 4,646,499 (Mar. 10, 1987) (assigned to Hewlett-Packard).

258. *See, e.g., In re* Gulak, 703 F.2d 1381, 1385 (Fed. Cir. 1983).

259. *See* John P. Sumner & Steven W. Lundberg, *The Versatility of Software Patent Protection: From Subroutines to Look and Feel,* COMPUTER LAW., June 1986, at 1, 1, 5.

260. *Id.* at 5.

261. *See generally* Daniel J. Kluth & Steven W. Lundberg, *Design Patents,* COMPUTER LAW., Aug. 1988, at 1, 1.

262. 35 U.S.C. § 173 (1982).

263. 35 U.S.C. § 171. A design must also meet the requirement of nonobviousness. 35 U.S.C. § 103. *See In re* Nalbandian, 661 F.2d 1214 (C.C.P.A. 1981).

264. Power Controls Corp. v. Hybrinetics, Inc., 806 F.2d 234, 238-39 (Fed. Cir. 1986); *see also* D. CHISUM, *supra* note 169, § 1.04(2)(d). There is some question regarding whether a designer must elect between copyright and design patent protection. *See, e.g., In re* Blood, 23 F.2d 772 (D.C. Cir. 1927). More recently, courts have ruled that design patent protection is available for a design that has been registered under the Copyright Act. *See In re* Yardley, 493 F.2d 1389 (C.C.P.A. 1974).

Office issued what are thought to be the first design patents for graphical computer screen displays.[265] The patented designs include icons for a wastebasket, file divider, physical floppy disk drive, telephone, and "softkey" menu display.

3. *Other modes of legal protection.*

Because consumers in search of compatible user interfaces are conscious of trade names and the interface "standards" for which they stand (*e.g.*, Lotus 1-2-3™ spreadsheet),[266] trademark is a particularly important form of legal protection in the computer-human interface area. Trademark protection gives the designer of a computer-human interface long-term benefits from having its product be first on the market, even without copyright or patent protection. By attaching a distinctive trademark or trade name to the interface, the manufacturer encourages consumers to associate the interface with that manufacturer. Furthermore, to the extent that consumers value compatibility, this gives the manufacturer an advantage in marketing complementary products in the future.

Trade secret law might also have limited value in protecting the research that goes into designing computer-human interfaces. To the extent that researchers studying human factors can maintain secrecy over their work, they can stay ahead of competitors in the race to introduce better products.

B. *Evaluation of Current Protection for Computer-Human Interfaces*

Although legal protection for computer-human interfaces is still in a relatively early stage of development, a number of patterns seem to be emerging. The early copyright cases concerning screen displays have favored a liberal interpretation of what constitutes protectible expression. As with application program code, therefore, copyright appears to be an important form of legal protection for at least the video display portion of computer-human interfaces. As noted, patent protection is available for useful, novel, and nonobvious aspects of interfaces and ornamental designs and is becoming a more widely sought form of legal protection. In addition, trademark serves as an important form of legal protection in the many software markets in which consumers value standardized interfaces and reputations for quality.

This section first applies the general economic framework of Part II to the analysis of legal protection for computer-human interfaces. On

265. *See* 1090 Off. Gaz. Pat. Office 1090, U.S. Design Pat. Nos. 295,631, 295,632, (May 10, 1988) (assigned to Xerox Corp.); 1090 Off. Gaz. Pat. Office 1626, U.S. Design Pat. Nos. 295, 762, 295, 764 May 17, 1988) (assigned to Xerox Corp.); 1091 Off. Gaz. Pat. Office 1073, U.S. Design Pat. No. 296, 218 (June 14, 1988) (assigned to Xerox Corp.); 1091 Off. Gaz. Pat. Office 1516, U.S. Design Pat. No. 296, 339 (June 21, 1988) (assigned to Xerox Corp.); Kluth & Lundberg, *supra* note 261, at 1 & n.2.

266. *See* note 118 *supra*.

the basis of this analysis, the second subsection evaluates the current regime of legal protection for computer-human interfaces, focusing principally on the direction of copyright law.

1. *Economic analysis of legal protection for computer-human interfaces.*

The development of new and better computer-human interfaces is affected by two market failures—the public goods problem and network externalities—each with particular implications for the design of legal protection. The first subsection analyzes the public goods problem with respect to computer-human interfaces. The second analyzes the implications of the fact that computer-human interfaces generate network externalities (as a result of consumer preferences for standardized interfaces).

Public goods problem. In the absence of legal protection for the design of computer-human interfaces, designers of screen displays and other interfaces might not be able to appropriate sufficient reward for their intellectual efforts because others can reproduce such interfaces at relatively low cost. To assess the need for legal protection of such efforts, we must examine the costs of designing computer-human interfaces, the availability of market mechanisms for appropriating value for creative effort, and nonmarket incentives or rewards available for such efforts. We must also assess the costs of legal protection, including the social costs of monopoly pricing and the effects on innovation.

Most interface design efforts involve applying established computer-human interface principles[267] to a particular problem; they do not involve significant original research.[268] For example, in designing the screen display for a new business inventory program, the programmer does not need to conduct human factor experiments for each aspect of the interface. He or she typically will need only to apply a number of established principles of screen design—*e.g.*, whether to use a menu-driven or command-driven program for a particular task,[269] how to sequence the tasks involved,[270] how to abbreviate information,[271] and where to place information on the screen.[272] This is not to

267. *See* notes 33-48 *supra* and accompanying text.

268. Many of the large-scale interface research efforts are conducted at universities, *see, e.g.*, Shneiderman, *supra* note 113, at 27, and through government-funded research projects, *see, e.g.*, SOFTWARE TRADE STUDY, *supra* note 58, at 57 (describing Department of Defense research project involving speech and natural language understanding, and vision); *cf.* Grimes, Ehrlich & Vaske, *supra* note 29, at 22-23 (presenting data showing that a number of Human Factors Society members work in government and defense-related organizations).

Even Apple Computer's major effort in developing the "desktop" interface for its Macintosh computer system borrowed interface principles from the public domain. The main ideas underlying the desktop interface were developed in the 1970s by Xerox Corporation at its Palo Alto, California research center. *See* Schlender, Miller & Carroll, *supra* note 16, at 21, col. 5.

269. *See, e.g.*, Hauptmann & Green, *supra* note 39.

270. *See, e.g.*, note 41 *supra*.

271. *See, e.g.*, note 42 *supra*.

say that the interface designer's task is trivial. Rather, it is to highlight that interface design is an applied field in which product development costs are relatively low for many commercial products.[273]

The public goods question is whether program developers have an adequate opportunity to recover these costs in the marketplace. A programmer has a certain amount of lead time because of the time it takes others to reverse engineer or rewrite complex application code implementing the interface.[274] Whether this amount of time is adequate is a difficult question; but, as noted in the analysis of application program code,[275] a variety of factors—including the advantage of being the first to introduce a product, the opportunity to begin marketing prior to a product's introduction, and the relatively short product cycle for most application programs—suggests that the timing is close. In markets for interfaces capable of generating significant network externalities, first-mover advantages are likely to be particularly important because of the potential for becoming a de facto industry standard. Much of this value is captured in a trademark's goodwill, which is a long-lasting asset.[276]

For some particularly complex interfaces, however, the design effort might involve conducting computer-human interface experiments to determine which ways of accomplishing tasks are most efficient—for example, fastest or easiest to remember. Although more expensive than simply hiring a programmer well-trained in the science of computer-human interaction, such original research is typically conducted only for products targeted for large markets, in which the rewards for designing a successful product are correspondingly large. Nonetheless, there is a risk that some of this research might not take place if competitors could quickly make use of the innovator's work. This suggests that legal protection should play some role in helping innovators recoup development costs.

The costs of legal protection must be considered in assessing how extensive such protection should be. The owner of a de facto industry standard could reap the value consumers place on standardization through monopoly pricing.[277] This would cause a significant welfare loss if many consumers were priced out of the market. Broad and/or long-lasting legal protection might also have detrimental effects on re-

272. *See, e.g.*, note 44 *supra*.

273. *Cf.* SOFTWARE TRADE STUDY, *supra* note 58, at 10.

274. *See* notes 212-213 *supra* and accompanying text.

275. *Id.*

276. A further value in being first to develop an interface for an important task comes from being a step ahead of competitors and having the know-how to make compatible/consistent interfaces. Thus, in markets with network externalities, there is an important "prospect" opportunity. *Cf.* Edmund W. Kitch, *The Nature and Function of the Patent System*, 20 J.L. & ECON. 265, 267-71 (1977).

277. *Cf.* Karjala, *supra* note 9, at 44-48 (discussing the problem of consumers becoming effectively locked in to interface standards as a result of investments in training).

search and development in application programming. The inability of programmers to gain access to desirable interfaces—for example, because the owner of intellectual property rights in that interface refused to license—would have a significant chilling effect on the development of complementary computer programs. Furthermore, protecting fundamental interface concepts would hinder their diffusion into different types of application programs.

The public goods rationale for protecting interfaces, therefore, justifies protecting only those interface designs that go beyond the state of the art. Thus, as in the area of application program code, the threshold requirements of patent law—novelty, usefulness, and nonobviousness—strike the most appropriate balance between incentive and access with regard to the protection of interface design.

Network externalities. Because standardized interfaces can generate external benefits for computer users, the economic analysis must also consider the effects of legal protection on the realization of network externalities. Computer-human interfaces generate network externalities to the extent that standardization produces larger networks and reduces retraining and mobility costs. As discussed in Part II, providing legal protection for products generating network externalities has two opposing effects: Though it discourages the realization of network benefits (by inhibiting widespread adoption of a protected standard), the reward to innovators of better standards can overcome the tendency of markets featuring network externalities toward being trapped in an inefficient standard.

While no system of legal protection can perfectly resolve this conflict, legal protection can be tailored to achieve a desirable balance. Two principal supply-side costs are associated with introducing a new interface—the cost of developing the interface and the cost of marketing it (*i.e.*, overcoming negative inertia). In addition, there are demand-side costs associated with switching to a new interface, including the costs of retraining operators and replacing obsolete equipment. Because it is often costly to change standards, it is not efficient to switch whenever small improvements become available. Yet, obsolete interfaces should not be retained because the producer of the new standard cannot appropriate a sufficient share of the innovation's value to justify a successful marketing effort.

For new interfaces that merely apply established interface design principles, development costs are modest. Thus, as discussed in the analysis of the public goods problem, the market system will perform reasonably well in generating good interfaces for most application programs without extensive intellectual property rights. In light of the network externalities flowing from standardization, however, the market system will tend to underreward manufacturers' marketing efforts to establish their products as the standard. Consumers will lose out to the

extent that numerous products performing the same task but using different interfaces proliferate. They will also lose out if better standards are not created and/or marketed because the manufacturer cannot appropriate a sufficient financial reward. In the absence of governmental policies or industry efforts to promote standardization,[278] this suggests that developers of products capable of generating network externalities should have some legal protection to encourage investment in innovation and marketing.

The difficult questions are how long and how broad this protection should be. In order to have an incentive to develop and aggressively market a new interface standard, the manufacturer must be assured that others will be unable to compete with identical interfaces and substantially lower costs until the original manufacturer has recouped its innovation and marketing costs.[279] This is particularly important where the inertia of an existing standard is significant. Because of the importance of promoting standardization, however, the intellectual property regime should promote access to an interface as soon as the product embodying it achieves wide diffusion in the market.

2. *Evaluation of the current direction of legal protection for computer-human interfaces.*

The first cases addressing the copyrightability of computer-human interfaces have taken the view that most of the effort going into the design of video screens (and their sequencing) is artistic expression rather than the use of utilitarian ideas. The test used by these courts to draw the line between expression and idea indicates that the scope of copyright protection is broad. This approach is likely to inhibit the innovation and diffusion of improved computer-human interfaces.

With regard to the diffusion of programs embodying visual interfaces, the prevailing interpretation of copyright protection results in substantial loss due to unjustifiable monopoly (that is, unneeded in order to generate an optimal level of research, development, and marketing). In light of the large body of accumulated learning in the human factor analysis field,[280] most improvements in computer-human interface design would not be viewed as novel and nonobvious as these terms are used in patent law. Patent examiners skilled in the computer-human interaction field would view much of the commercially oriented

278. *See* Comments of IEEE Computer Society Concerning Registration and Deposit of Computer Screen Displays in Response to Public Notice Issued by U.S. Copyright Office (Sept. 8, 1987), *reprinted in* 6 COMPUTER L. REP. 538, 549 (1987) [hereinafter IEEE Testimony] (referring to computer industry efforts to promote standardization of user-friendly interfaces); note 135 *supra* (noting the government's passive role with regard to interface standards for computers).

279. This period of protection could be reduced to the extent that the first comer obtains long-term benefits from being first, such as a good reputation (protected through trademark) or an advantage in developing the next generation of products.

280. *See* text accompanying notes 37-48 *supra*.

work being done as merely applied work. Yet because the first interface for a particular task apparently is copyrightable under the analysis of the early cases, the creator of that interface has significant market power.

The effects of this expansive approach to copyrightability on innovation are striking. Because the first effort to implement a task via a screen display (or sequence of displays) will likely employ many user-friendly concepts, the availability of copyright protection for most aspects of the screen display makes it extremely difficult for others to offer similarly "friendly" competing products.[281] Unlike with application program code,[282] an equally good computer-human interface cannot be independently developed (*e.g.*, through clean-room procedures) because "equally good" in the context of a standardized interface means exact duplication. This is impossible unless the programmer has access to the original. As a consequence of the sequential nature of innovation in application programming, this expansive approach can have detrimental effects on the advancement of application programming generally by discouraging the whole range of innovations that might depend upon the use of good, basic interface design features for their success.[283]

Broad copyright protection for video displays also discourages the adoption of interface standards, preventing the full realization of network externalities. The *Digital Communications* case illustrates this problem. Because computer users involved in data transfer often come in contact with many different computer systems, there are large benefits from having a uniform visual representation of the various parameters for transmitting data among computers. But since the Crosstalk interface is proprietary, other firms wishing to improve upon or expand its capability are forced to incur the costs of negotiating a license with Digital Communications Associates or designing another interface. Furthermore, as the IEEE Computer Society notes, expansive copyrightability of user interfaces could undermine industry efforts to standardize essential elements of good screen display design.[284]

---

281. These concerns are reflected in the comments by the Computer Society of the IEEE before the Copyright Office:

> It is important not to lock up by copyright the only ways of presenting information for a particular type of computer program that are user-friendly or reflect sound human factor analysis. . . . To do that would prevent legitimate competition. That result could occur if the idea/expression borderline were so drawn that the best techniques of promoting user-friendliness were considered expression.

IEEE Testimony, *supra* note 278, at 549.

282. *See* note 216 *supra* and accompanying text.

283. *Cf.* Schlender, Miller & Carroll, *supra* note 16, at 21, col. 4; Schwartz, *supra* note 15, at 3, col. 1; Schwartz, *supra* note 249, at 3, col. 1; Peter Waldman, *Software-Copyright Laws Are in State of Confusion,* Wall St. J., Mar. 21, 1988, at 21, col. 4.

284. *See* IEEE Testimony, *supra* note 278, at 549 (noting more generally that overbroad copyright protection of screen displays "could hinder widespread public acceptance of computers and computer programs, by causing a negative public reaction to any resulting coerced use of unpreempted, but unfriendly, screen displays, or by imposing excessive training costs

Patent disputes regarding utility and design patents on interfaces have not yet been brought to the fore, so we can only conjecture about the ability of patent protection to address the public goods and network externality problems. As noted in the economic analysis section, threshold requirements for patent protection—novelty, utility, and nonobviousness—are roughly consistent with rewarding only those innovations that would not be forthcoming in the absence of protection. The efficacy of the utility patent system will depend critically upon how carefully the threshold criteria are adhered to in issuing patents, the extent to which the exclusivity of patent rights discourages realization of network externalities, and whether the patent protection period is longer than necessary to encourage the optimal level of investment in research and development.[285]

The efficacy of design patent protection of screen display features depends critically upon how carefully the functionality limitation is applied. A further problem arises to the extent that protected, nonfunctional designs become functional because of consumer learning. In such circumstances, design patent protection could discourage realization of network externalities.

C. *Toward a Better Standard of Copyright Protection for Computer-Human Interfaces*

Because of its expressive form, its inherent utilitarian purpose, and the importance of its standardization, computer-human interfaces pose difficult problems for the intellectual property system. The previous section showed that the current direction of copyright protection does not bode well for the innovation and diffusion of improved computer-human interfaces. This section outlines an alternate course that responds to the complex policy concerns raised by computer-human interfaces and is consonant with fundamental principles of intellectual property law.

A clear implication of the economic analysis is that the functional aspects of computer-human interfaces should be protected only if they would not be readily forthcoming in the absence of legal protection. For functional works such as utilitarian screen displays, this amounts to requiring a threshold of novelty and nonobviousness. Despite contrary implications in recent cases, this proposition is not only consistent with but fundamental to the copyright and patent systems. Thus, courts addressing the copyrightability of computer-human interfaces must replace the *Whelan*-type test with more careful adherence to the purpose of section 102(b) of the Copyright Act. This will require courts to delve into the science of human factor analysis.

---

on business because of lack of standardization"); text accompanying note 98 *supra* (discussing industry standardization efforts).

285. *See* note 219 *supra*.

The approach advocated here would be fairly straightforward in cases involving "user-friendly" interfaces. Aspects of interface design derived from human factor analysis would not be protected by copyright law. Thus, in cases such as *Broderbund*,[286] in which the principal aspect of interface design is display screen sequencing, the critical issue would be whether display screen order is dictated by logical, dexterity, or cognitive skills of users that can be ascertained through the study of human factors. To be consistent with section 102(b) of the Copyright Act, the test applied in *Broderbund*—whether any means of writing a program for designing greeting cards, posters, banners, and signs other than the plaintiff's exists—implicitly assumes that other such means of implementing the task are equally useful; otherwise, copyright was permitted to extend to protection of an "idea, procedure, process, system, method of operation, concept, principle, or discovery."[287] As the discussion of interface design in Part I highlighted, however, many aspects of interface design are derived from human factor analysis. Therefore, copyright protection should at most be limited to those aspects of interface design that are *not* "user friendly."

Implementation of this test in copyright cases would not mean that functional aspects of interfaces are not eligible for legal protection. Rather, such innovations would have to meet the more exacting threshold requirements of the patent law.[288]

The more complex question concerns treatment of those aspects of an interface that did not initially reflect human factor principles—such as purely stylistic features of screen layouts or arbitrary labeling conventions—but have since become functional because the program has been widely accepted in the marketplace and users have grown accustomed to such features. This might occur for no other reason than that users become familiar with the particular screen layout or labeling conventions of the first program on the market performing a particular task. Alternatively, because aspects of the application program code are patented, other firms might have taken a long time to develop competing products; and as a result, consumers might have become accustomed to the pioneering product's interface.

Such arbitrary or purely expressive (as opposed to "functional" in the sense used in Part III) aspects of the interface would be protected by copyright law when the interface is introduced. But as users learn

286. *See* note 248 *supra* and accompanying text.

287. 17 U.S.C. § 102(b) (1982).

288. In addition to the delay and cost of seeking protection, *see* notes 142, 186-187 *supra* and accompanying texts, the patent system might have some drawbacks as a means of protecting computer-human interfaces. In particular, the patent holder's right to exclude others from making, using, or selling the invention for 17 years, 35 U.S.C. § 154 (1982), would inhibit realization of network externalities. It might be desirable, therefore, to have a compulsory licensing provision applicable to patents on interfaces capable of generating network externalities. *Cf.* Menell, *supra* note 9, at 1365-66.

such features, they become relevant to human factor analysis[289]—they become functional to the extent that user knowledge is an important human factor. In such situations, the ideal legal regime protects the particular aspects of the interface for long enough to allow the creator to recover his or her development and marketing costs (adjusted for risk) and then to allow others access to the "standard" so as to foster standardization and realize resulting network externalities.[290] Although it is difficult to know precisely how long this period needs to be in each case, granting the creator exclusive rights to the interface's arbitrary aspects until it is purchased by a large enough body of users to constitute a network would seem to provide sufficient lead time to encourage the introduction of improved standards.[291] In light of the nature of the public goods and network externality concerns, this approach surely seems better than the direction taken in recent copyright cases.

During the interim period before enough products have been sold to constitute a network, copyright protection for the arbitrary aspects of screen design serves much the same purpose as copyrightability for nonfunctional aspects of application program code structure, sequence, and organization—it provides lead time for program manufacturers to recover the costs of developing and marketing aspects of a program that do not rise to the patent law threshold. As before, choosing copyright protection is best seen not as a means of encouraging the development of nonfunctional aspects of a program but rather as an effective means of providing creators of application programs with sufficient lead time to exploit their products in the market. In light of the long-term benefits of being associated with the industry standard, allowing protection for the time necessary to establish a network would seem to come close to providing sufficient lead time.

As in the analysis of legal protection for application program code, the idea/expression merger doctrine provides the key to implementing the proposed approach for copyright protection of interfaces. In the ordinary application of the doctrine, the subject matter at issue can be evaluated as idea or expression at the moment of creation. What has not been addressed by the cases is how to evaluate material that, although nonfunctional at its creation, becomes functional as a result of

289. *See* text accompanying note 38 *supra*.

290. One could argue that the legal system should not foster standardization of interfaces embodying nonfunctional (or inefficient) features because such standardization will, because of network externalities, create inertia that discourages the adoption of better standards should they be created. *See* text accompanying notes 132-135 *supra*. While this argument has some merit, it must be balanced against the value placed on standardization and the efficacy of the incentives created by the intellectual property system for overcoming inertia should a major interface breakthrough be made.

291. This type of protection might cause innovators to delay widespread introduction of a product in order to prolong invention. This effect is limited, however, by innovators' incentive to make sales quickly in markets for products with short life cycles. *Cf.* notes 58-62 *supra* and accompanying text.

user learning. This situation seems to reflect the reality of many computer-human interfaces. While not functional at their creation, they become so because of the consumers' training (human capital investment) in using them. As the discussion of interface design technology described, human learning is an important element in computer-human interface design. In these situations, granting protection that excludes competitors from such a significant product market is inconsistent with section 102(b) and the nature of copyright. Whereas in the general case (such as novels and other traditional literary works) copyright protection for expression will not unduly hinder access to a market because exact expression is usually not critical to function,[292] in the context of standardized interfaces exact duplication is key.[293] In effect, an interface's widespread adoption in a market that values standardization transforms an otherwise nonfunctional interface into a functional one. Thus, the idea/expression merger doctrine, if applied with due attention to the dynamic nature of markets in which standardization is an important consideration, is consonant with the proposed standard.[294]

In applying the idea/expression merger doctrine to computer-human interface cases, courts should be sensitive to the extent to which interface standardization is an important feature for the particular application program task and the extent to which the first comer's interface has become established as a market standard. In essence, the court should invoke the merger doctrine only when the first comer's interface has become a de facto industry standard.[295]

Although the proposed standard does not identify a bright line, its workability is suggested by the efficacy of the genericness doctrine of trademark law,[296] which resembles the proposed standard. Under the genericness doctrine, trademark protection is lost if a mark becomes

---

292. *See* William M. Landes & Richard A. Posner, An Economic Analysis of Copyright Law 41-47 (Apr. 13, 1988) (unpublished manuscript).

293. *Cf.* Menell, *supra* note 9, at 1360-63 (discussing a similar problem in the context of computer operating systems).

294. An alternative approach would be to deny copyrightability to nonfunctional interfaces that could be expected to become standardized interfaces. Under this approach, a court, recognizing that a new program would have broad appeal and that its potential users would value a standardized interface, could deny copyrightability at the outset regardless of whether the interface is functional. In essence, programs of this variety would be presumed functional because of the consumer demand for standardization. This approach, however, would have a number of drawbacks in comparison to the proposed standard. It would not allow the creator any reward for developing the interface other than that which could be reaped during the time it took competitors to develop a competing version of the underlying application program. In addition, it would create an anomaly in copyright law by not permitting any protection for nonfunctional images on video displays that would otherwise be eligible for protection.

295. In addition, courts should allow copying of only those aspects of an interface that are essential to being on the network. Thus, the use of distinctive artistic patterns to decorate a video display should continue to be protected to the extent that they are *not* related to function.

296. *See* William M. Landes & Richard A. Posner, *Trademark Law: An Economic Perspective*, 30 J.L. & ECON. 265, 291-96 (1987). *See generally* 1 J. MCCARTHY, *supra* note 200, § 12.

generic.[297] Similarly, under the proposed test, once nonfunctional aspects of an interface become generally accepted in the marketplace, the creator can no longer prevent others from using it. In developing standards for determining when an interface has become "generic," as in the trademark area courts would look to the extent of consumer learning and the breadth of the market. It would also be possible for courts to weigh on a case-by-case basis other factors directly related to the intellectual property balance, such as the extent to which the interface designer has recouped his or her development costs (adjusted for risk of failure)[298] and the cost to consumers of multiple interface standards.

Even if a court did not view the idea/expression merger doctrine as flexible enough to implement this standard, the fair use doctrine[299] might allow limited access to interfaces that become de facto industry standards.[300] Although the fair use doctrine is typically unavailing when the alleged infringing use directly reduces the creator's potential commercial market,[301] the importance of interface standardization to competition in the application program market would seem to alter the standard fair-use calculus. As one commentator has suggested in a related context, the fair use doctrine is flexible enough to recognize the importance of standardization in computer products and should be applied so as to discourage copyright owners from refusing to license de facto industry standards protected by copyright.[302]

---

297. *See, e.g.*, Miller Brewing Co. v. Falstaff Brewing Corp., 655 F.2d 5 (1st Cir. 1981) ("Lite" beer); King-Seely Thermos Co. v. Aladdin Indus., Inc. 321 F.2d 577 (2d Cir. 1963) ("Thermos" bottle).

298. In most cases, the designer of an interface that becomes a de facto industry standard will have more than recouped the costs of his or her efforts (including an adjustment for the risk). It is possible, nonetheless, that the loss of legal protection for an interface standard after it becomes a de facto industry standard will reduce the incentive to develop better standards. In light of the other incentives for innovation in standards—including the advantages from being first (with resulting trademark benefits) and government subsidies of interface research, *see* note 268 *supra*—this adverse effect is not likely to be large.

299. The fair use doctrine is defined in the Copyright Act as follows:

> Notwithstanding the provisions of section 106 [exclusive rights in copyrighted works], the fair use of a copyrighted work . . . is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—
>
> (1) the purpose and character of the use, including whether such use is of commercial nature or is for nonprofit educational purposes;
>
> (2) the nature of the copyrighted work;
>
> (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and
>
> (4) the effect of the use upon the potential market for or value of the copyrighted work.

17 U.S.C. § 107 (1982).

300. *See* Goldstein, *supra* note 9, at 1129-30; Leo J. Raskind, *The Uncertain Case for Special Legislation Protecting Computer Software,* 47 U. PITT. L. REV. 1131, 1175-82 (1986).

301. *See* Harper & Row, Publishers v. Nation Enters., 471 U.S. 539, 566 (1985) (noting that the use's effect upon the potential market for the copyrighted work is the most important factor).

302. *See* Goldstein, *supra* note 9, at 1130. An alternative approach to implementing the proposed standard would be for courts to develop a copyright misuse doctrine along the lines

V. CONCLUSION

Although the intellectual work embodied in new technologies does not always fit easily within the traditional modes of legal protection, the basic principles underlying the intellectual property system often provide the flexibility necessary to promote the development of new technologies. For this to occur, however, the courts must, in delineating the scope of each mode of intellectual property protection, adhere closely to these basic principles; otherwise they risk undermining the greater goals of the intellectual property system.

In the nascent area of copyright protection for computer software, the courts regrettably have failed to recognize, much less avoid, this danger. Because computer software is in essence a utilitarian device in literary form, the idea/expression merger doctrine—which restricts copyright from the province of patent—takes on critical importance in determining the proper scope of copyright protection. The fundamental flaw in the courts' efforts to draw the line between idea and expression has been to view application programs and the video screens that they generate principally as creative (in the artistic sense) expression. In so doing, the courts have effectively given broad legal protection to basic principles of application program design without requiring that the creators of programs embodying those principles satisfy the exacting standards of patent law. Such a result threatens to impose grave monopoly costs on consumers, undermine efforts to standardize interfaces, and inhibit innovation in application programming generally.

This article's analysis has shown that the idea/expression merger doctrine, if applied carefully, can promote the development and diffusion of new and better application programs. The critical steps in achieving this goal are recognizing, first, that the principal components of application programming—the computer-human interface design and program layout and coding—are guided largely by functional considerations, and second, that standardization of computer-human interfaces is a legitimate functional goal.

This is not to say that intellectual property protection for computer software could not be significantly improved by *sui generis* forms of protection tailored to the unique technological and economic attributes of application programs.[303] Rather, the purpose of this article has been

---

of the patent misuse doctrine. Under the patent misuse doctrine, courts refuse to enforce an otherwise valid patent if the owner exploits it by violating the antitrust laws or by extending the patent beyond its lawful scope. *See generally* 4 D. CHISUM, *supra* note 169, § 19.04. In light of the predominantly functional nature of computer programs and their interfaces, the considerations underlying the patent misuse doctrine are fully applicable to cases involving copyrights covering aspects of computer-human interfaces. Therefore, courts should develop a copyright misuse doctrine to discourage owners of copyrights in user interfaces from engaging in discriminatory or unreasonable licensing practices. *See* Goldstein, *supra* note 9, at 1127-29.

303. *See* Karjala, *supra* note 9, at 95 (questioning whether an adequate public policy rationale has yet been articulated for granting computer software "the full panoply of copyright

to demonstrate that careful interpretation of the existing law can go a long way toward attaining the societal goal of properly promoting the invention and diffusion of new and better application programs.

protection"). Even if the standard and scope of patent protection were thought to be appropriate for protecting certain aspects of the intellectual work embodied in application programs, the question would still remain whether patent protection's duration was justified. *See* note 219 *supra*. For a general analysis of tailoring legal protection for computer software, see Menell, *supra* note 9, at 1364-67, 1371.