UIC John Marshall Journal of Information Technology & Privacy Law

Volume 13
Issue 2 *Journal of Computer & Information Law - Winter 1995*

Article 4

Winter 1995

Computer Software: Intellectual Property Protection in the United States and Japan, 13 J. Marshall J. Computer & Info. L. 245 (1995)

Jack M. Haynes

COMPUTER SOFTWARE: INTELLECTUAL PROPERTY PROTECTION IN THE UNITED STATES AND JAPAN

By JACK M. HAYNES†

I. INTRODUCTION

Since it is software, not hardware, that brings computers to life and allows them to serve their users in ways that those users find desirable, and since the ability to make a business out of software depends heavily on the existence of intellectual property rights in that software, it is becoming clear that the extent of the software protection offered in the developed world will determine the outcome of what may be the last great competitive war of the current international industrial economy. One does not have to be an MBA to be able to see that if software were deprived of all legal protection, money to fund the development of new commercial software products would dry up.[1]

The following statistics provide insight into what is at stake in this international competitive battle. A United States Trade Representative (USTR) report estimated that in 1986, five hundred and fifty-seven billion dollars of information intensive products in the United States were directly affected by intellectual property rights.[2] The highest concentration of sales was in computer software and entertainment.[3] In 1986, worldwide losses due to inadequate intellectual property protection were close to twenty-four billion dollars.[4] Two-hundred and seventy-one million dollars were spent in 1986 on identification and enforcement costs

* B.S. (CPT) Purdue University (1984), MBA University of Indianapolis (1992), J.D. Candidate Indiana University at Indianapolis; an Advanced System Engineer with 10 years of data processing experience; and a committee member ABA Intellectual Property Section on New Information Technology.

1. ANTHONY LAWRENCE CLAPES, SOFTWARS: THE LEGAL BATTLES FOR CONTROL OF THE GLOBAL SOFTWARE INDUSTRY 6 (1993).

2. *See* MEHEROO JUSSAWALLA, THE ECONOMICS OF INTELLECTUAL PROPERTY IN A WORLD WITHOUT FRONTIERS: A STUDY OF COMPUTER SOFTWARE 41-42 (1992).

3. *Id.* at 42.

4. *Id.*

for global protection.[5] While only about one percent of total sales infringed copyright, this contributed to a forty billion dollar balance of payments deficit for the U.S. in 1986.[6] Worldwide, losses of revenue resulting from inadequacies in intellectual property protection for computers and software were over four trillion dollars, and eighty-one percent of thirty-one firms responding in the report incurred some loss due to inadequate intellectual property protection.[7] Thus, the stakes are high, especially for the United States, since the United States controls seventy percent of the fifty billion dollar worldwide software market.[8]

The game is also expensive for the individual market participant. Developing the software costs millions of dollars in labor hours, but the software is easily copied in a matter of minutes or hours. Violating the rules is also expensive for the market participant. In 1988, Fujitsu paid IBM eight-hundred and thirty-three million dollars in fees to settle a dispute surrounding violations of an agreement that would allow Japanese software to run on IBM compatible hardware.[9]

The statistics reveal that the health of a major industry and the United States' dominance of that industry will be greatly impacted by the protection which software receives within national legislatures and international agreements. Therefore, the following will make recommendations to improve protection of software based on insights gained from an analysis of the software development process, analysis of software protection in the United States and Japan, and analysis of the major international agreements that protect computer software.

II. SOFTWARE DEVELOPMENT PROCESS

Before one can begin to understand intellectual property protection for computer software, it is necessary to understand the software development process. To understand software development, one must first understand the definition of a computer and its components. A computer is a machine that processes data according to a set of instructions.[10] It is comprised of hardware, consisting of all the physical equipment, and software, which is the set of instructions that tell it what to do.[11]

---

5. *Id.*

6. *Id.*

7. *See Id.* at 46.

8. John C. Phillips, *Sui Generis Intellectual Property Protection for Computer Software*, 60 GEO. WASH. L. REV. 997, 1001 (1992).

9. JUSSAWALLA, *supra* note 2, at 113.

10. ALAN FREEDMAN, THE COMPUTER GLOSSARY: THE COMPLETE ILLUSTRATED DESK REFERENCE 124 (5th ed. 1991).

11. *Id.*

A. HARDWARE

The basic hardware of the computer consists of a central processor unit (CPU), main memory, input devices, and output devices. The CPU is made up of the arithmetic unit (ALU) and control unit.[12] The ALU performs arithmetic calculations and compares numbers.[13] Main memory of the computer is typically made up of many random access memory (RAM) integrated circuits. The CPU reads program instructions from main memory and directs input into main memory before processing it.[14] Main memory is like a large checkerboard. Each square contains one character, and each square has a unique address, similar to a post office box. This enables the CPU to find information. Data is also output from memory before being sent to an output device.[15] A computer can have many input devices, but typically every system has at least a keyboard. Other types of input devices are CD/ROM drives, mice, diskette drives, and light or touch sensitive input screens. Typical computer output devices include printers or monitors. Some devices, such as disk drives, are both an input and an output device. Hardware systems vary in size from small personal computers to large computers called mainframes, but the basic hardware components are found on all sizes of computers.

B. SOFTWARE

A computer is useless without software. The two types of software typically found on a computer are operating system software and application software. Operating system software "provides interfaces [italics omitted] that make it easier to develop programs for the system by reducing the amount of code that must be written, and simplifying the exercise of certain [hardware] functions."[16] The 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, applications programs, and the computer user."[17] Application software consists of one or more computer programs that fulfill a specific function for the user like word processing, bookkeeping, or financial analysis. Thus, the generic term software refers to a collection of programs which can either be application software or operating system software.

---

12. *Id.* at 143.

13. *See Id.* at 15.

14. *See Id.* at 374.

15. *See Id.* at 375.

16. HAROLD LORIN & HARVEY M. DEITEL, OPERATING SYSTEMS 1 (1981).

17. Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1051 (1989).

A computer program is defined under the copyright statute as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."[18] Computer programs are written in many different languages, each with their own unique syntax and structure. Originally, programs were written in machine language. As the machines and operating system software became more complicated, programmers began to write in more sophisticated symbolic languages. While these languages were much easier to code in than machine language, computers could not understand them. Therefore, the program goes through a step called compilation to turn the original source language into a language understood by the computer. The source language is called source code, and the language understood by the computer is called object code.[19] Many computers also require an additional step, called link editing, which turns the object module into a module the computer can execute, called a load module.[20] Today many personal computers and some mainframe software vendors provide only load modules or object code with purchased software. A person is thus prohibited from selling or marketing a slightly modified and disguised version of the original package. However, a software product called a decompiler will take object or executable code and produce a source module which can then be altered.[21] The ability to circumvent any attempt at securing the original source creation has been a big problem for many software manufacturers.

C. Software Development

The process of developing software is largely an artistic endeavor intermixed with a little science. The overall process is similar to designing and constructing a building and is generally guided by a project plan. Like the architect, the software developer initially attempts to determine the customer's software requirements. During this stage, the software developer models the overall processing flow, the data elements or fields to be manipulated, the structure of the data, and the design of the inputs and outputs. However, unlike the architect, the software developer cannot produce a model that replicates the look of the final product. The software developer, in fact, relies on various diagramming methods that provide a blueprint to aid the programmer in constructing the code. Unfortunately, the person paying for the product can rarely interpret these blueprints. To complicate matters, the written program looks nothing

18. 17 U.S.C. § 101 (1992).

19. *See* Freedman, *supra* note 10, at 123.

20. William S. Davis, Operating Systems: A Systematic View 21-22 (1977).

21. *See* Pamela Samuelson, A Case Study on Computer Programs, in Global Dimensions of Intellectual Property Rights in Science and Technology 284, 291 (1993).

like the blueprint. Therefore, during the construction stage of the project, the programmer can exercise much creativity in building the program to meet the specifications or blueprint. At this stage, the programmer faces a number of complicated puzzles. The first puzzle encountered is how to string together a series of instructions, using a language with a limited vocabulary, to accomplish the task according to the blueprint. Unlike the carpenter, the programmer was not given a blueprint of the final product and must begin to visualize what this program will look like. The ideas expressed in the blueprint must be converted into a series of program statements whose characteristics and syntax vary widely depending on the language in which the programmer chooses to write.[22] Programming is partly science because the programmer must adhere to the computer language syntax and work within the machine constraints as established in the hardware and operating system. Once the program is written, the programmer tests the code and resolves problems identified during testing. Resolving problems in software is often called debugging.[23] After testing is complete, the customer reviews the product. Because the initial design phase did not identify all the customer's requirements, the customer often requests additional changes. Once testing is complete, the system is implemented. Implementation involves turning over the software to the customer and freezing the source code, so that future problems with the software can be identified and fixed. Others can then pursue creating new features in the software.

D. Future of Software Development

The initial software design phase often results in specifications that do not match the customer's needs or expectations. Specification and design flaws account for half of the errors in applications, and because part of the system must be rebuilt these errors are costly to fix.[24] However, errors made in the construction or coding phase are inexpensive to fix.[25] The industry has responded by developing Computer Aided System Engineering (CASE) tools to remove the flaws created in the design phase.[26] CASE tools involve the end user more in the design process. Diagrams for planning, analysis, and design are drawn in a structured environment to ensure accuracy of the design.[27] Integrated CASE (I-CASE) tools automate the entire design and construction process, and

22. *See* Clapes, *supra* note 1, at 12-13.

23. *See* Freedman, *supra* note 10, at 172.

24. *See* James Martin, Information Engineering: Book III Design and Construction 97 (1990).

25. *Id.*

26. *See Id.* at 33.

27. *See Id.* at 30-31.

most of the actual coding process is completed by the computer.[28] However, I-CASE tools still require some coding because the tool typically cannot deliver all the required programs which meet all the customer's needs. Other tools automate all or part of the software development process. All the tools on the market attempt to create software which fulfills the customer's requirements at a reduced cost by generally decreasing program construction labor hours and increasing design labor hours.

Another emerging trend in software development is software which can run on any hardware platform. Initially software was developed for one machine. A program written for an IBM machine could not run on a Digital Electronics or Hewlett Packard machine. Standardized computer languages were developed, such as Beginners All Purpose Symbolic Instruction Code (BASIC), Formula Translation (FORTRAN), and Common Business Oriented Language (COBOL);[29] however, computer programs developed and compiled on a particular machine cannot run on another machine. Each machine understands a unique set of machine language instructions and is controlled by a unique operating system.

However, in an open system world, customers can choose products from a variety of hardware and software vendors.[30] Software specifications are open if they are generally proclaimed by public bodies, or the author does not have reserved rights. If the author has reserved rights, a license is available for a nominal fee.[31] In this world, there is a standard operating system which runs on many different hardware platforms of different sizes, and programs are freely portable across these platforms.[32] Intellectual property rights are usually not an issue because the software base for the open system is typically developed by a consortium of vendors. However, there is disagreement in the industry on whether the UNIX operating system developed by AT&T or the Disk Operating System (DOS) developed by Microsoft provides the open operating system solution.[33] In this country, progress toward open systems is further hindered by the instability of the software alliances which are building open systems, the accretion of proprietary versions of the open operating systems, and the failure of customers to build a sound business case for migrating to open systems.[34] However, the European commu-

28. *See Id.* at 29.

29. *See* FREEDMAN, *supra* note 10, at 47, 253, 112.

30. *See* CLAPES, *supra* note 1, at 263 (*quoting* Bill Gates of Microsoft Corporation).

31. *Id.* at 31.

32. *See Id.* at 269.

33. *See Id.* at 264.

34. *See Id.* at 268-269.

nity is moving much faster toward the adoption of open systems.[35]

Thus, after the various forms of protection for computer software are reviewed, and software protection in the U.S., Japan, and in international agreements are reviewed, recommendations can be made considering the emerging trends toward automated software development and open systems.

III. FORMS OF INTELLECTUAL PROPERTY PROTECTION FOR SOFTWARE

Property is generally thought of as land or personal belongings, property that occupies space or can be consumed, or tangible forms of property.[36] Property rights for certain forms of information or intangible property, such as computer programs, are protected under intellectual property law.[37] Although the intangible nature of this property makes protection difficult, software, as a form of intellectual property, is generally protected using the traditional legal mechanisms found in copyright, patent, trade secret, trademark, and licensing. Background information is provided on each area.

A. COPYRIGHT

Copyright protection provides relatively long term protection for the expression of the idea not the idea.[38] Copyright does not prohibit an author from independently coming up with an identical or similar text.[39] If there is one way to express an idea, the idea is not protected.[40] Copying for fair use is allowed for academic research or criticism.[41]

B. PATENT

A patent protects inventive advances in a technological process, a product, or a machine design. A patent is granted after completing an examination procedure by a government agency.[42] Property rights are conferred to fresh, useful, and nonobvious processes and goods.[43] The protection granted is for a shorter duration than copyright, but it affords more protection because it covers anyone selling, using, or making a

35. *See Id.* at 270.

36. Marshall A. Leaffer, *Protecting United States Intellectual Property Abroad: Toward a New Multilateralism*, 76 IOWA L. REV. 273, 279 (1991).

37. *Id.*

38. *See* SAMUELSON, *supra* note 21, at 285.

39. CLAPES, *supra* note 1, at 27.

40. *Id.*

41. *Id.*

42. SAMUELSON, *supra* note 21, at 287.

43. Leaffer, *supra* note 36, at 279 n.31.

claim of invention without the owner's permission.[44] This includes protection against people who independently develop an identical or highly similar invention.[45] Ideas can be protected with a patent. However, the procedure for granting a patent precludes using trade secret.[46]

C. Trade Secret

Trade secret requires that information be secret, be reasonably protected, and have value economically.[47] A trade secret may consist of "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."[48]

An owner does not apply to the government to get trade secret protection, and the owner is not granted absolute power to bar others from any activity.[49] The trade secret owner can prohibit acquisition by improper means such as bribery, breach of contract, or industrial espionage, but the trade secret owner cannot prohibit reverse engineering, independent development, or derivation from public sources.[50]

D. Trademark

Words, names, symbols, and other devices that distinguish goods are protected through trademark law.[51] In the U.S., trademark rights are acquired through the use of the mark, but in other countries, trademark rights are acquired by registration.[52] Trade names are important in computer software because consumers, looking for compatible products, are very conscious of brand names like Windows, Lotus 1-2-3, and Excel.[53]

E. License

To protect trade secrets in mass-marketed software, many developers of commercial software use shrink-wrap licensing agreements as a means of limiting the rights of consumers.[54] This license is typically on a printed form inserted between the shrink-wrap and the box containing

44. Samuelson, *supra* note 21, at 287.
45. *Id.*
46. *Id.*
47. Michael D. Stein, *The Importance of a Trade Secret as a Supplement to Copyright Protection of Computer Software*, 12 Intell. Prop. L. Newsl. 28, 29 (Fall 1993).
48. *Id.*
49. *Id.*
50. *Id.*
51. Leaffer, *supra* note 36, at 279 n. 30.
52. *Id.*
53. *See* Menell, *supra* note 17, at 1092.
54. Samuelson, *supra* note 21, at 291.

the software, and the license informs the owner that by removing the shrink-wrap the owner agrees to all terms and conditions of the license.[55] The purchaser of a shrink-wrap licensed product is typically informed that he is not an owner of a copy of the software. This "avoid[s] the provision[s] of 17 U.S.C. Section 117 that grant[s] certain rights to modify and make backup copies" to owners of software.[56] This license also typically prohibits decompilation.[57] However, shrink-wrap license enforcement is dubious.[58] Other licenses, such as the site license, grant the holder the right to run multiple copies of the software.

The software developer has a number of methods to protect his creations. However, depending on the current protection granted under each of these regimes, the developer will get varying results. If the developer is operating in several countries, the variation in protection is magnified. The following section reviews intellectual property protection for computer software in the U.S., in Japan, and in international agreements.

IV. INTELLECTUAL PROPERTY PROTECTION FOR COMPUTER SOFTWARE

A. U.S. INTELLECTUAL PROPERTY PROTECTION FOR COMPUTER SOFTWARE

The current state of software protection in the U.S. for software and the problems with this protection can only be understood after tracing the development of the law.

1. *Historical Development of Software Protection in the U. S.*

Software initially was created by academics and researchers without much regard for protection, and computer manufacturers often bundled software with the hardware.[59] In the 1960's, computer software protection was thought to fit best into the copyright area, since it could provide long-term protection against unauthorized copying without an expensive registration requirement.[60] In 1964, the U.S. Copyright Office decided to begin accepting computer programs for copyright.[61] However, the office accepted them under its "rule of doubt," which leaves the ultimate question whether the work qualifies for copyright protection up to the

55. *Id.* at 291 n.16.
56. *Id.*
57. *Id.* at 291.
58. *See* SAMUELSON, *supra* note 21 at 291; Menell, *supra* note 17 at 1077-78.
59. SAMUELSON, *supra* note 21, at 284.
60. *Id.* at 285.
61. CLAPES, *supra* note 1, at 25.

courts.[62] The office based its decision on *White-Smith Music Co. v. Apollo*. In *White-Smith*, the Supreme Court determined that a piano roll used in a player piano did not infringe upon copyrighted music because the roll was part of a mechanical device.[63] Since a computer program is textual, like a book, yet mechanical, like the piano roll in *White-Smith*, the Copyright office granted copyright protection under the "rule of doubt."[64] However, few companies took advantage of the protection afforded computer programs because they were not mass-produced at this time, and the Copyright Office required deposit of the source code which precluded trade secret protection.[65]

The next significant development concerning U.S. intellectual property protection for software occurred in the patent area. The Patent Office initially viewed computer programs which were not part of a patentable industrial process as unpatentable.[66] In *Gottschalk v. Benson*, the position of the Patent Office was reiterated by the Court which held that a program which converted binary-coded decimal numbers to pure binary was not patentable because granting the patent would result in the monopolization of a fundamental building block of science.[67] Hence, at this point, the software industry in the United States continued to rely on licensing agreements and trade secret. Patents were largely unavailable for computer software. Furthermore, copyrights which were available remained unfavorable to market participants.[68]

In 1974, Congress created the National Commission on New Technological Uses (CONTU) to investigate whether the evolving computer technology field outpaced the existing copyright laws and to determine the extent of copyright protection for computer programs.[69] CONTU concluded that while copyright protection does "extend beyond the literal source code of a computer program," evolving case law should "determine the extent of protection."[70] CONTU also found that copyright was the best alternative among existing intellectual property protective mechanisms. Furthermore, CONTU rejected trade secret and patents as viable protective mechanisms.[71] The CONTU report resulted in the 1980 Computer Software Copyright Act, and the CONTU report now acts as an informal legislative history to aid the courts in interpreting the Act.[72]

62. SAMUELSON, *supra* note 21, at 285 n. 2.
63. 209 U.S. 1 (1908).
64. SAMUELSON, *supra* note 21, at 286.
65. *Id.*
66. *Id.* at 286-287.
67. 409 U.S. 163 (1972). *See* Phillips, *supra* note 8, at 1023.
68. *See* SAMUELSON, *supra* note 21, at 288.
69. Phillips, *supra* note 8, at 1011.
70. *Id.*
71. *See* SAMUELSON, *supra* note 21, at 289-90.
72. *See* Phillips, *supra* note 8, at 1012.

However, CONTU did not eliminate the industry's reliance on trade secret and protective licensing, as evidenced by the development of shrink-wrap licensing and the proliferation of machine language software distribution.[73] Additionally, CONTU did not eliminate patent as a protective mechanism due to the Patents Office interpretation of the Supreme Court's decision in *Diamond v. Diehr*. In *Diehr*, the court held that one element of a rubber cutting process, a computer program, was patentable.[74] The patent office interpreted the decision broadly and issued many patents to software developers. The only thing considered unpatentable were abstract mathematical algorithms.[75]

2. *Current State of Software Protection in the U.S.*

The stage is now set for exploration of the current state of protection. The software developer is afforded a rare position because he may select from the areas of patent, copyright, trade secret, and licensing to protect his commodity.[76] However, the developer's choice carries a price. Certain alternatives are risky due to controversies still blazing in certain areas.

a. Current Controversies in U.S. Software Protection

Current controversies exist regarding the extent of copyright protection afforded the non-literal elements of software, regarding the patent protection afforded other types of software developments, and regarding the enforceability of shrink-wrap licensing.[77] The following section explores these controversies and recent case law surrounding some of these controversies.

There are several cases which illustrate the controversy over the copyright protection afforded other software elements. Until 1992, most of the federal courts followed the decision in *Whelan Associates Inc. v. Jaslow Dental Laboratory, Inc.*[78] Whelan, a small software company, wrote an accounting program for Jaslow, a dental laboratory company, for an IBM Series/1 minicomputer. Jaslow rewrote the software to run on an IBM personal computer and proceeded to sell the product. Although the source code was different, the data structures, logic, and program structure were identical.[79] Jaslow argued the duplicated elements were part of the idea, not the expression; however, the court held

73. *See* SAMUELSON, *supra* note 21, at 290-91.

74. 450 U.S. 175 (1981).

75. *Id.*

76. *See* SAMUELSON, *supra* note 47, at 294 n. 25.

77. *Id.* at 295.

78. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987). *See* Stein, *supra* note 47, at 28.

79. CLAPES, *supra* note 1, at 30.

that the structure, sequence, and organization are afforded copyright protection just as the source code or literal elements are protected.[80] The court felt that there was a single function of a computer program which represented the idea and that everything not necessary to that function expresses the idea.[81] Thus, *Whelan* provided software developers with relatively strong protection to all elements of software.

However, in 1992, this protection was weakened by *Computer Associates v. Altai, Inc.*[82] Altai, a software developer, was accused of copying various modules of CA-SCHEDULER, a Computer Associates software package which controls the running of jobs (made up of multiple programs) on IBM mainframes.[83] The court rejected *Whelan's* premise that a computer program embodies one function because programs are made up of sub-routines that contain their own idea.[84] The court developed the abstraction-filtration process for determining whether one work or program is substantially similar to another. The process requires both works to be broken down into constituent elements to separate unprotected ideas from expression. This identifies the protected material. The protected material in the works is then examined to determine if it is substantially similar to warrant infringement.[85] Unprotected elements include components dictated by efficiency and required by factors external to the program. Efficiency is an industry goal that can result in two programmers writing substantially similar programs. Extrinsic considerations include the mechanical specifications of the computer, program compatibility required by other programs in a software package, manufacturer design standards, design standards of the industry being served, and widely accepted programming practices within the software industry.[86] The court recognized this process would narrow the scope of software copyright protection and found this in accordance with Congressional intent for protection of computer programs with copyright.[87] Thus, copyright protection afforded software currently is not as broad.

Proponents of the use of patents as a method to protect computer software argue that patents can protect valuable assets not protected under copyright and promote progress in computer program innovation.[88] Opponents argue that the Patent Office is unable to do a good job

80. *See* STEIN, *supra* note 47, at 28.

81. *Id.*

82. 982 F.2d 693 (2d Cir. 1992).

83. *See* CLAPES, *supra* note 1, at 208-209.

84. *See* 982 F.2d at 705; *see also* Stein, *supra* note 47, at 28-29.

85. Michael D. Stein, *The Importance of a Trade Secret as a Supplement to Copyright Protection of Computer Software*, 12 INTELL. PROP. L. NEWSL. 29 (Fall 1993).

86. 982 F.2d at 709.

87. *Id.* at 712.

88. *See* SAMUELSON, *supra* note 21, at 301.

issuing software patents, and absent widespread patent protection, the software industry has grown rapidly. Furthermore, scientists and mathematicians are hesitant to issue patents for algorithms which they regard as fundamental truths.[89] While the controversy regarding the use of patents still rages, the courts and patent office continue to allow patenting of certain types of software.

Licensing software with shrink-wrap licensing agreements is used extensively in the industry, but the licenses have not been successful in deterring widespread copying. The legality of shrink-wrap licensing has not been put to the test in the courts. However, as profits in the industry become slimmer and competition increases, the probability of the issue arising increases.

3. *Conclusion and Recommendations for U.S. Software Protection*

Today, vendors are advised to protect their software by utilizing a strategy employing trade secret coupled with licensing and copyright.[90] Although standard procedures for protecting trade secrets do not exist, it is recommended that the software developer identify and inventory items which might constitute trade secrets, inform employees of what the trade secrets are and what restrictions are imposed on their distribution and use, utilize non-disclosure agreements to create confidential relationships with key employees, customers, and others, and take reasonable physical measures to protect the trade secret.[91] Trade secret protection claims are easier than copyright or patent infringement to explain because the plaintiff must only show the defendant took something improperly and is not required to explain the technical composition of the subject matter.[92]

U.S. law is arguably inadequate for protection of computer software. Several recommendations for improvement have been made: passing a sui generis statute which better addresses the problem in distinguishing the idea from the expression in software;[93] limiting copyright to protecting the elements of structure, sequence, and organization which are not related to enhancing computer efficiency[94] (which is similar to the *Computer Associates* approach); and adopting a sui generis legislative plan creating a separate administrative agency which would require software developers to file for protection similar to a patent, yet provide protection for only one year, while requiring filers to allow access to other industry

89. *Id.* at 301-303.

90. *See* Stein, *supra* note 47, at 30.

91. *Id.* at 29.

92. *Id.* at 30.

93. *See* Robert A. Arena, *A Proposal for the International Intellectual Property Protection of Computer Software*, 14 U. PA. J.INT'L BUS. L. 225-26 (1993).

94. *See* Menell, *supra* note 17, at 1085.

participants upon payment of a reasonable licensing fee.[95]

Software is constantly being created and modified. New tools are emerging and being adopted. An agency would need a large staff to monitor these changes and revise administrative procedures accordingly. Hence, the legislative plan for a separate agency is untenable, and any legislative enactment would be antiquated as soon as it passed. With the emerging use of automated software development tools, the growth in open systems, and the rapid pace of change in the industry, the judicial case-by-case approach used in the U.S. is preferred.

The U.S. has been very successful in software development and is the world leader in the industry. It is conceded that improvements can be made in the law which has grown more complex under *Computer Associates*, but the improvements are best made on a case-by-case basis within the judiciary. Occasional updating of the U.S. statutes as major trends in technological advancement occur is not prohibited under this recommendation; however, these changes should merely be minor increases in the scope of coverage afforded intellectual property under current statutes. The judicial approach recommendation is consonant with the advice a participant gave at a 1990 National Research Council workshop on software protection within the U.S. Improving upon an old military saying, the participant commented "If it ain't broke, don't break it."[96]

B. Japanese Intellectual Property Protection for Computer Software

To understand the protection afforded software in Japan under various intellectual property regimes, one must first understand the historical development of protection in Japan which, in turn, requires an understanding of the Japanese legal system.

1. *Japanese Legal System*

Japan's judiciary is an independent branch of government.[97] However, Japan is a civil law country and its courts are not bound by stare decisis.[98] Japan has an amount of litigation equal to one- twentieth per capita of the U.S., largely because the Japanese have a firmly rooted sense of *wa*, or civil harmony, that has lasted for two centuries.[99] Conservatism within the judiciary has been assured by a centrally controlled

95. *See* Phillips, *supra* note 8, at 1032-1033, 1036, 1038.

96. Clapes, *supra* note 1, at 294.

97. *Id.* at 172-173.

98. Wean Khing Wong, *Protecting American Software in Japan*, 8 Computer/L.J. 111, 115 (Spring 1988).

99. *See* Clapes, *supra* note 1, at 173.

education process which allows only two percent of applicants to become lawyers, a centrally controlled court system which allows the Supreme court to instruct lower courts on the decisions it expects them to make, and a judicial reappointment process which does not allow lifetime appointment. As a result, judicial decisions follow the bureaucracy.[100] Therefore, the size and conservatism of Japan's judiciary precludes resolution of all intellectual property issues related to software through the judiciary, and recommendations for improvement in Japan's treatment of the subject requires consideration of the cultural evolution of law in Japan.

2. *Historical Development of Legal Mechanisms for Software Protection*

The Japanese initially felt, similar to the U.S., that software protection was not important, and it was not until the late 1970's, when software was first mass produced for personal computers and video games, that the issue arose.[101] Debate concerning the appropriate legal device for software protection, however, arose in two governmental research studies conducted in the early 1970s.

The Ministry of International Trade and Industry (MITI) Study Committee on Legal Protection of Software issued a report in 1972 on the current state of protection for computer software and recommended several changes. The study found that programs qualified for copyright, but the copyright protection afforded programs was inadequate.[102] The inadequacies found in existing copyright law included failure to prohibit unauthorized use, failure to discourage duplicative development due to the absence of a public disclosure system, and long-term protection under copyright was inappropriate for computer programs.[103] Several recommendations were made, among them adopting new legislation after a worldwide consensus on the issue, requiring public disclosure or registration of software, and using arbitration to resolve disputes.[104]

The Second Subcommittee within the Copyright Council published a report on computer problems which examined the nature and extent of copyright protection. Two recommendations were made to revise the copyright law: to grant the copyright owner distribution rights, and to develop a registration process which would facilitate distribution of work

---

100. *Id.*

101. *See* Teruo Doi, *Computer Technology and Copyright-A Review of Legislative and Judicial Developments in Japan*, 8 MICH. Y.B. INT'L LEGAL STUD. 3, 8 (1987).

102. Torhu Nakajima, *Legal Protection of Computer Programs in Japan: The Conflict Between Economic and Artistic Goals*, 27 COLUM. J. TRANSNAT'L L. 143, 144-45 (1988).

103. *Id.* at 144.

104. *Id.* at 145 (footnote 13-14).

and avoid duplication of effort.[105] The report authors concluded that voluntary registration was sufficient to promote economic interests of program developers, and that there was little justification for shortening the period of computer protection.[106]

Although the government did not move forward with these recommendations, a 1982 decision of the Tokyo district court clarified the issues. In *Taite K.K. v. K.K.ING Enterprises*, the defendant converted its customers' video games, stored as object programs on Read Only Memory chips (ROM), into the plaintiff's "Space Invader Part II" machines.[107] For the first time in Japan, the court recognized the plaintiff's computer object program as a creative expression protected by copyright. This decision motivated the two agencies to reconsider computer software protection.[108]

In 1983, MITI, departing from the earlier report, categorized computer software as economic assets which are fundamentally different from other copyrighted material such as novels, paintings, and music.[109] The report recommended six principles for any system of legal protection for software. The system should consider the special characteristics of software (ease of copying and the need to maintain confidentiality after sale), promote industrial use of software and protect various rights, remove impediments to improving existing software by balancing rights of new and old software, protect economic rights of users and developers, be flexible enough to adapt to rapid change, and take into account the international nature of the legal protection of computer software.[110] The report concluded that the existing laws failed to provide adequate protection and recommended the enactment of a sui generis approach to protect computer software called the Program Rights Law (*Puroguramukenho*).[111] MITI outlined the Program Right Law in 1984. The provisions called for a term of protection of fifteen years, with right of use, and compulsory licensing; however, there was no protection for the moral rights of the creators as called for in the Berne Convention.[112]

The same year that MITI recommended a sui generis approach, the Copyright Council released a report recommending that Japan follow the worldwide trend of using copyright law as the primary mechanism of protecting software.[113] Although MITI had more power than the Copy-

105. Doi, *supra* note 101, at 6.
106. Nakajima, *supra* note 102, at 145.
107. *See* Doi, *supra* note 101, at 10.
108. *See* Nakajima, *supra* note 102, at 147.
109. *Id.* at 148.
110. *Id.*
111. *See Id.* at 148-149.
112. *See Id.* at 149-151.
113. Wong, *supra* note 98, at 113.

right Council, the Copyright Council's recommendation was followed due to external pressure from the European Economic Community (EEC) and the U.S.[114]

The Japan Copyright Law was amended in 1985 and became effective in 1986. The major provision of the amendments defined a computer program (*puroguramu*) as "an expression of combined instructions given to a computer so as to make it function and obtain a certain result,"[115] added program works to the list of works of authorship, extended copyright protection only to the form of expression, allowed authorship to be extended to either the natural person or the legal person with the natural person receiving extended protection of life plus fifty years, and established a registration system for computer programs.[116] Because Japan follows the Berne Convention, registration is not required to enjoy the protection of copyright.[117]

The Copyright Law was amended again in 1987 to extend protection to a database (*databesu*), which is defined as "a collection of theses, numerical figures, drawings and other pieces of information organized systematically so that these pieces can be searched (*kensaku suru*) by the aid of a computer."[118]

Japan also provides protection for computer programs under patent law and the Unfair Competition Prevention law.[119] Under patent law, computer programs are protected if they have some industrial application and are based on some natural law as opposed to human mental activity or mathematics.[120] Patent protection granted for computer programs in Japan is similar to the U.S. approach, and many of the computer programs which would be patentable under *Diehr* would be patentable in Japan.[121] However, under Japanese patent law, the guidelines are rather stringent, and many computer programs would not be protected under patent law.[122] Patent law in Japan is also criticized heavily by U.S. companies because of the time it takes to receive a patent. In Japan, it takes six to seven years to get a patent, while in the U.S. it takes an average of nineteen months.[123] However, when U.S. companies experience difficulties in Japan, it is often because U.S. com-

114. *Id.*
115. Doi, *supra* note 101, at 13.
116. *See Id.* at 13-15.
117. *Id.* at 15.
118. *Id.* at 17.
119. Nakajima, *supra* note 102, at 164.
120. Howard G. Pollack, *The Gordian Algorithm: An Attempt to Untangle the International Dilemma over the Protection of Computer Software*, 22 LAW & POL'Y INT'L BUS. 815, 825 (1991).
121. *Id.*
122. Nakajima, *supra* note 102, at 164.
123. Lawrence Matis et al., *Japan IP Update*, 9 J. PROPRIETARY RTS. 31, 32 (1993).

panies do not allow enough time for Japanese patent attorneys to translate patent applications which may affect the scope of protection granted the patent holder.[124]

The Unfair Competition Prevention Law prohibits copying of product indications such as trademarks or brand names. However, a claim cannot be made unless the product is widely known. A Tokyo District Court applied the Unfair Competition Prevention Law to protect a video game machine, finding the pictures projected and movement of the pictures were product indications.[125]

After exploring the historical development of software protection, the current state of protection afforded Japanese software can be explored.

3. *Current State of Software Protection in Japan*

Copyright is the pervasive protective mechanism for software in Japan, but patent and the Unfair Competition Prevention Law may supplement when applicable. However, this scheme has resulted in controversies in several areas, including the scope of protection for software elements, the reverse engineering process, and the scope of protection for microcode and operating systems.[126] These areas will be explored and recommendations for improvements made considering the nature and evolution of Japanese law.

a. Current Controversies in Japanese Software Protection

Japan and the U.S. have difficulty determining the scope of protection for software because the idea in the software is not easily distinguished from the expression. The U.S. currently uses the *Computer Associates* abstraction-filtration process to delineate idea from expression. Japan's view in this area can best be determined by analyzing a 1987 decision in the Chisai District Court.[127] In *Microsoft v. Shuuwa Trading*, the Japanese court was considering, *inter alia*, whether the overall structure, the composition of routines and data where protectable nonliteral elements.[128] The court concluded, as did the U.S. court in *Whelan*, that the nonliteral elements exhibited creativity deserving of expressive protection under copyright.[129] Some commentators, suggesting Japan would interpret *Whelan* differently, argue that the defini-

124. *Id.*

125. Nakajima, *supra* note 102, at 164.

126. Dennis S. Karjala, *The Limitations on the Protection of Program Works Under Japanese Copyright Law*, 8 MICH. Y.B. INT'L LEGAL STUD. 25, 26 (1986).

127. *See* Pollack *supra* note 120, 824 n. 70; *See* CLAPES, *supra* note 1, at 174.

128. *See* CLAPES, *supra* note 1, at 174.

129. *Id.*

tion of an algorithm in Japanese copyright law would deny protection for the overall structure of a program.[130] In comparing the extent of coverage of Japanese and U.S. copyright protection for software, varying conclusions have been reached. Some commentators conclude that in Japan, coverage for non-literal software elements would be less broad than in the U.S.,[131] other commentators conclude that the coverage would be very similar,[132] and some conclude that the issue is largely unsettled.[133] With the advent of *Computer Associates*, which narrowed the scope of protection in the U.S., the scope of protection is similar in both countries.

The Japanese position regarding reverse engineering can best be described by analyzing *Microsoft*. In this case, Microsoft claimed that Shuuwa, a Japanese firm which marketed computer publications, infringed Microsoft's copyright of a BASIC interpreter (similar to a compiler) by printing a reverse assembled or decompiled listing of the interpreter.[134] The court held that two infringements occurred. The process of creating a reverse engineered listing was an infringement, and the publication of the listing was an infringement.[135] Based on the *Microsoft* case and provisions within the copyright law, commentators have concluded that "reverse engineering might be inhibited if the Copyright Law were literally interpreted."[136] However, controversy exists within Japan's civil law system because the Copyright Law neither explicitly allows nor prohibits reverse engineering.

Microcode software controls the functions of a computer and is written to simplify the logic circuitry within a computer.[137] In the U.S., copyright protection was extended to microcode in *Allen Myland, Inc. v. International Business Machine*.[138] However, in Japan, whether the Copyright Law covers microcode is not clear. Microcode may not be protected because it does not fit the definition of either a computer program or a creative work.[139] But microcode might be protected limited by the necessity of achieving compatibility with a competing chip.[140] Because of ambiguity in the Copyright Law, it is not clear whether microcode is protected. Protection of operating system software is also uncertain. If operating systems are determined to be a program language under Japa-

130. Karjala, *supra* note 126, at 44; *see* Wong, *supra* note 98, at 120-121.
131. *See* Karjala, *supra* note 126, at 44; *see* Wong, *supra* note 98, at 122-123.
132. *See* CLAPES, *supra* note 1, at 174.
133. Pollack *supra* note 120, 824.
134. *See* CLAPES, *supra* note 1, at 171.
135. *Id.* at 175.
136. Nakajima, *supra* note 102, at 166; Judith J. Welch & Wayne L. Anderson, *Copyright Protection of Computer Software in Japan*, 11 COMPUTER/L.J. 287, 295 (1991).
137. Karjala, *supra* note 126, at 38 (note 32).
138. 746 F. Supp. 520 (E.D. Pa. 1990). *See* CLAPES, *supra* note 1, at 190.
139. Karjala, *supra* note 126, at 42.
140. *Id.*

nese law, operating systems are not protected.[141] If operating systems are determined to be part of the machine, operating systems are not protected.[142] Operating systems may be protected if the copyright law is interpreted differently. Thus, the Copyright Law does not make it clear whether operating systems and microcode are protectable works.

4. *Conclusions and Recommendations for Japanese Software Protection*

The Japanese treatment of software is quite similar to the U.S. system. The primary mechanism of protecting software is copyright which is supplemented by patent and other forms of protection, and each system generates similar legal issues. However, solutions must be tailored in light of each country's culture and legal system. Consideration must be given to Japan's civil law system and aversion to litigation. The recommendations should also consider the industries adoption of open systems and automated software development tools. The following considers adoption of a sui generis approach for software protection and revising Japanese Copyright Law.

MITI initially proposed adoption of a sui generis approach to software, but the EEC and U.S. dissuaded Japan from adopting this approach. The U.S. disliked the MITI proposal because of the term of protection and compulsory licensing requirements.[143] A sui generis approach could be adopted which would not have these objectionable features and maintain compatibility with industry trends, Japanese culture, and Japanese law. Copyright inadequacies could also be addressed. However, the development of a separate system of protection would be costly to industry participants, difficult to draft, and result in a whole new set of problems.

Japanese Copyright Law could be redrafted to address the current ambiguities and inadequacies. This change would have less of an impact upon industry participants, it could embrace new development technologies, and fewer problems would arise. The vast majority of countries also use copyright to protect software. While changes might be difficult to draft and result in some new problems, this recommendation is preferred because it generates fewer problems.

C. Software Protection in International Agreements

The Berne Convention and the Universal Copyright Convention (UCC) are international agreements which deal with intellectual prop-

141. *Id.* at 33.
142. *Id.* at 32.
143. *See* Nakajima, *supra* note 102, at 155-156.

erty protection for computer software.[144] The following describes protection under the two agreements and problems which have arisen under the agreements.

1. *The Berne Convention*

The Berne Convention does not explicitly grant protection for computer software,[145] but "the absence of limits in Article 2 can be taken as confirmation that the machine-readable computer program is protected."[146] Signatories to the agreement, among them the U.S., receive protection consistent with U.S. copyright law.[147] However, American copyright law is different in two respects. First, the moral rights of authors are not protected under U.S. law.[148] Moral rights are exclusive rights granted to the author which allow the author "to object to the use of [his] name" or "destroy the original" copies of the works.[149] "Second, [U.S.] copyright law is more stringent [in] deposit, registration, and notice.[150] Under the Berne Convention, only the author's name need appear on the work to receive protection. This difference is why the U.S. initially refused to join the Berne Convention.[151] Despite the initial refusal to join the Berne Convention, intellectual works of U.S. authors could still receive protection if published first in a member country or "simultaneously in a non-member and member country."[152] The Berne Convention grants protection for the "life of the author plus fifty years."[153] "National treatment" is provided for under the Convention, and this requires that "member nations treat non-nationals the same as nationals."[154]

2. *The Universal Copyright Convention*

"The UCC, to which the U.S. is a signatory," does not explicitly mention software in its list of protectable works, but the list has not been

144. Max W. Laun, *Improving the International Framework for the Protection of Computer Software*, 48 U. PITT. L. REV. 1151, 1154 (1987); Arena, *supra* note 93, at 218.

145. Arena, *supra* note 93, at 218.

146. Laun, *supra* note 144, at 1155.

147. *Id.*

148. *Id.* at 1156.

149. *See Id.* at 1151.

150. *Id.*

151. *Id.*

152. *Id.*

153. Arena, *supra* note 93, at 230 (citing The Berne Convention for the Protection of Literacy and Artistic Works of Sept. 9, 1886, completed at Paris on May 4, 1896, revised at Berlin on Nov. 13, 1908, completed at Berne on Mar. 20, 1914, revised at Rome on June 2, 1928, revised at Brussels on June 26, 1948, revised at Stockholm on July 14, 1967, and revised at Paris on July 24, 1971, 828 U.N.T.S. 221) [hereinafter Berne Convention]).

154. *Id.* at 228-29 (citing Berne Convention, art. 3(1)(a), art. 5(1)).

interpreted exclusively.[155] The author receives exclusive rights to authorize reproduction, as under the Berne Convention, but the rights lapse after seven years if the author has not granted a reproduction.[156] The term of protection shall not be less than the life of the author plus twenty-five years.[157] Unlike the Berne Convention, the UCC requires the author to affix the copyright sign (c), the author's name, and the year of publication to receive protection.[158] National treatment is also provided in the UCC.

3. *Problems Under the Berne Convention and UCC*

There are several problems created by these agreements. First, national treatment under the two agreements is problematic.[159] National treatment applies the law of the state where remedies are sought, and this provision creates uncertainty in the amount of protection granted by a given country.[160] Second, the granting of moral rights is incompatible with computer software which is utilitarian in nature.[161] Allowing a programmer to destroy a work is devastating to those who are dependent on the program.[162] The term of protection is also a problem because it far exceeds the useful life of software.[163]

Various recommendations for remedying the problems include adopting a sui generis computer software protection law in the General Agreement on Tariffs and Trade (GATT),[164] embracing the World International Property Organization (WIPO) proposal to eliminate national treatment and grant explicit rights to creators of software,[165] and amending existing national patent laws, locally and abroad, to cover computer programs.[166]

V. CONCLUSION

Because there is much to lose in the international competitive battle for dominance of the computer software industry, the U.S. has been

155. Laun, *supra* note 144, at 1151, 1157 (citing Raskind, *The Uncertain Case for Special Legislation Protecting Computer Software*, 47 U. PITT. L. REV. 1131, 1153-54 (1986)).

156. *Id.* at 1158.

157. Arena, *supra* note 93, at 230 (citing Universal Copyright Convention, revised on July 24, 1971, 25 U.S.T. 1341, 943 U.S.T. 1978, art. IV.(2)(a). The original draft was signed in Geneva on Sept. 6. 1952, 6 U.S.T. 2731, 216 U.N.T.S. 132.).

158. Laun, *supra* note 144, at 1158.

159. *See* Laun, *supra* note 144, at 1151, *see* Arena, *supra* note 93, at 219.

160. Arena, *supra* note 93, at 219.

161. Laun, *supra* note 144, at 1152.

162. Arena, *supra* note 93, at 219.

163. Laun, *supra* note 144, at 1152.

164. Arena, *supra* note 93, at 220.

165. *See* Laun, *supra* note 144, at 1159.

166. Pollack, *supra* note 120, at 820.

quick to exert pressure nationally and internationally to see that others adopt similar protective mechanisms. However, there are existing weaknesses in the protection of computer software locally and internationally. Further complicating matters are emerging trends toward open systems and automated software development. The U.S. needs to analyze the problem and make recommendations to improve the protection of software nationally and internationally. Moreover, the problem must be defined properly. The U.S. must consider dominant industry trends, inadequacies in existing U.S. copyright law, and inadequacies in existing international agreements, while giving special consideration to alternative legal systems.