ELSEVIER

Information & Management 34 (1998) 161–173

INFORMATION & MANAGEMENT

# Acceptable protection of software intellectual property: a survey of software developers and lawyers

Effy Oz*

*Management Division, Penn State University, Great Valley, 30 E. Swedesford Road, Malvern, PA 19355, USA*

Accepted 18 May 1998

**Abstract**

The article reports the results of a survey on the optimal legal way to protect developers' rights to their intellectual property in the US. Two groups were incorporated: software developers and attorneys. The majority of both groups favor copyright as the legal method, but attorneys prefer patenting with a longer protection period. There is no difference between the groups with respect to the desired length of copyright protection. Majorities in both groups prefer the current period of 75 years for corporations. By comparing information from 1992 and 1996, we find that software developers now are more in favor of protection of user interfaces. We found little differences between the groups regarding different categories of software. We also found high proportions of support for protection of source and object codes. Both groups prefer stronger protection for systems than application software. Attorneys, more than developers, favored greater protection for application software than for game software. We expected software developers to favor regulations that would force owners of systems software to offer their creation to any interested party for equal terms. To our surprise, there was significantly more support for this idea among attorneys than among the developers. Majorities in both groups support a special law for protection of software intellectual property. 

*Keywords:* Software; Software developers; Intellectual property; Copyright; Patent

## 1. Introduction

The software industry has become extremely powerful in recent years, and there is agreement that software is an important intellectual property. However, there are disagreements among people inside and outside the software industry on the necessary extent of the protection. Any proposed solution must be satisfactory to technical parties; i.e., corporate and individual developers of software. However, it must also be legally practical to follow and prosecute offenses.

Although the US Congress has considered the issue and the US Patent and Trademark Office has re-evaluated their legal approach toward protection of software, there have not been extensive studies that involved opinions of the software industry and the legal profession. In this study we polled members of these two groups about the issue.

To date, only one survey has been conducted to solicit opinions from software developers [4]. The

*Tel.: +1-610-648-3234; fax: +1-610-889-1334; e-mail: effyoz@psu.edu

0378-7206/98/$19.00 
*PII*: S-0378-7206(98)00049-4

purpose of our study was (1) to replicate that survey 4 years later; (2) to conduct a similar survey among attorneys who specialize in intellectual property and computer law, (3) to find the protection mechanism preferred by the two groups, and (4) to test the proposition that there are no significant differences between the two groups.

## 2. Legal and economic background

The growing dependence on software for business, scientific, educational, and entertainment purposes has created a very competitive software industry. Writing a computer program requires a substantial investment of time and money.

In a capitalist economy, very few are willing to make an investment unless there is reasonable grounds to believe that a profit can be made by selling it. The developer must, therefore, have the ability to prevent others from using a program without paying for it. Unfortunately for the developer, the inherent nature of software requires that a computer be able to copy the program before it can be used, but any program that can be copied with permission can be copied without it. To recoup the investment in a large program often requires the sale of millions of copies; this eliminates any possibility for a personal contract between the developer and the ultimate user.[1] Thus, the programmer must have some legal means of protection against unauthorized copying.

United States law provides the developer with three potential means of protecting against unauthorized copying and/or use: (1) trade secret; (2) patent; and (3) copyright. At present, the methods are often all used. Patent and copyright laws protect different aspects of the product. Both require publication of at least part of the software, which may interfere with trade secret rights. The law relating to all types of protection was developed long before the advent of computers and computer programs. It has not been changed radically.

The laws concerning trade secrets require that the subject matter be protected to remain secret, and that only persons who have a legal obligation not to reveal the information have access to that information. The moment the information comes into the hands of a person who has no such obligation, the information is no longer secret and the legal protection may be lost. The owner may have a cause of action against someone for breach of contract or fiduciary duty, but the proverbial cat is out of the bag, never to be recaptured.[2]

The US Constitution authorizes Congress to enact laws that grant a creator exclusive, but limited, rights to exploit his or her creation. Pursuant to that authorization, Congress has enacted laws granting 'patents' and 'copyrights.' The two types of legal protection protect different things. Patent law protects inventions, regardless of the means of expression. Copyright law protects the expression of ideas, but not the idea expressed. The requirements for obtaining patents and copyrights differ. To obtain a patent, the originator must show that the invention is new, unique, useful, and not merely an adaptation or application of an old idea.

Applying copyright law to software presents numerous problems. At a minimum, copyright law was not created with computers or software in mind. Books, paintings, plays, and the like, reproduced by hand or printing press, are equally protected. Technological advances such as photography, radio and television, challenged copyright laws, but the problems were adequately met with rather minor changes. Computer software poses an entirely different type of challenge, one that has not yet been met in spite of some copyright law amendments specifically related to computers; this has engendered tremendous controversy.

One problem with applying copyright law to software is the multitude of levels involved. At a minimum, for any one program there are three levels: source code, object code, and what is produced on the computer screen. If the copyright is limited to either object code or source code, the program is effectively unprotected.

Protecting codes does not solve the problem either. Software is a tool. It is used to make a computer

[1] The normal method of product distribution in today's world also effectively precludes most contact between the producer and the ultimate consumer.

[2] Even if the developer is ultimately successful in a breach of duty action, the time and effort required to obtain a judgment and the problems with collecting on that judgment make having a cause of action a poor alternative to retaining the secret.

perform particular functions, and to produce a desired result. Different instructions can produce the same or a similar result. The 'user interface,' what appears on the screen, is therefore a significant aspect of any software. That is also the point at which many significant problems arise with copyright law. The US Copyright Office has declared that the program codes and the resulting screen display are covered by a single copyright (see 37 C.F.R.† 202.3(b) (1993); Copyright Office Announcement, 36 Pat. Trademark & Copyright J. (BNA) 152, 155 (1988)). This, of itself, produces a conceptual problem not raised by traditional means of expression. When a painter produces a work of art, the resulting work (the user interface) is protected by copyright, not the brush strokes and techniques used to produce the work (the software). On the other hand, if a person writes a set of instructions for constructing a model ship, the written instructions are protected by copyright, not a model ship produced by some person following the instructions. The Copyright Office's position is analogous to protecting both the written instructions *and* the resulting model. For software, the Copyright Office's position is probably appropriate, but the legal theory problems engendered are apparent.

The utilitarian (tool) nature of software presents other problems. Despite what advertising might imply, there are a limited number of operations that can be carried out using a computer program. The three most pervasive are electronic spreadsheet operations, word processing operations, and games. (Since programs performing those tasks are desired by millions of users, those types of programs are potentially very lucrative.) There are many different programs that produce spreadsheet applications. To be utilitarian, a spreadsheet program must perform specific functions, e.g., produce a grid of columns and rows, allow entry of numbers in the grid spaces, manipulate the numbers in the grid spaces, allow the results to be printed, etc. Producing a functional spreadsheet program is an idea, not protected by copyright law. However, at some point one person's copyrighted spreadsheet program may be infringed by another (see Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990)).

Another significant problem relates to the current practice of building on, or assuming the availability of, an existing program. This is particularly true with respect to operating system programs. An application program must be written so that it is compatible with the operating system program; otherwise it may not be effective. These programs necessarily emulate or copy portions of the related existing program. If developers of these subsidiary programs must obtain licenses from the existing program's owner, the additional cost may be prohibitive, and in any event will tend to stifle innovation.

While a number of articles have been published which address this problem, most of them have been written from the legal point of view, or the software professional point of view, not both (e.g., [1, 3], and several Legally Speaking articles by P. Samuelson in CACM issues). The purpose of our study was to address both software developers and legal professionals with the same questions on the issue and to try to glean opinions that highlight a convergence toward an acceptable legal method to protect intellectual rights in software, as well as points of disagreement.

### *2.1. Arguments for software copyright and patents*

The main argument for protecting software under copyright laws is that it is a creation no different in nature than literary, musical, or any other artistic fruit of intellectual effort. If we wish to encourage development in this important segment of the economy, we must grant the right to make copies to the author of new software. At the cost of monopolistic power to the author, society gains tools that increase productivity and the general well-being of society. There are also some compelling arguments for patent protection for software [2]:

1. Software is not different from other areas of technological innovation. Denial of patents to software developers will discourage inventive endeavors, which, eventually, will hurt society.
2. Often, investment in research and development is substantial. (IBM reportedly invested US$ 2.5 billion in its OS/2 operating system program.) So is the effort to market the new product. Patents protect the initial intellectual assets of a small startup company. This enables the company to evolve into an industry giant. Examples: GE, AT & T, Polaroid, Xerox, and Hewlett-Packard.

3. Because of the general success of the software industry, few inventors have sought patents for their computer programs. However, imitators benefitted from the efforts of the true inventors. Had patents been more widespread, the true innovators would enjoy a fairer share of the rewards.
4. More software patents would result in a greater diversity of categories and features. Without patents, developers settle for improving existing ideas, but avoid the risk of researching new ideas.
5. The argument that software should be nonpatentable because it is used in computers, which are facilitators of information flow, is invalid. The telephone, phonograph, and radio, served the same purpose, and nevertheless were patented.

### 2.2. Arguments against software copyright

As copyright is granted for user interface design, the lion share of resistance is to granting protection to user interfaces. The major arguments are [5]:

1. Until 1986 there was no copyright protection for the user interface. There was incentive for software companies to develop better interfaces although others imitated them. The commercial success of the original works was not hurt by imitations. The main incentive was in the originality of the program. Greater protection increases the price to consumers.
2. Copyrights do not protect small companies. Suppose a small company develops a new interface but can reach only a few thousand buyers. A larger company that can tap a market of a million users imitates the interface. Granting copyright to the small company would not change the result. It would force the large company to develop another interface, but, because of its greater marketing clout, new customers are likely to prefer the product of the large company.
3. Copyrights serve the public by encouraging diversity, but diversity in interfaces is undesirable. While we want to read a new novel, listen to a new piece of music, and watch a new movie, diversity does not serve the public when it comes to interfaces. All cars are equipped with similar dash boards, similar steering wheels, and similar transmission sticks. This eliminates the need to retrain a driver for driving a new car. The same applies to software interfaces. What is needed is similarities, not diversity.
4. Monopoly on an established interface may yield monopoly on the functions accessed through the interface. This would reduce competition in the area where competition best serves the public: the functionality of the program.
5. Copyright is a tool for extortion. Virtually every computer program needs an interface. It is difficult to design a new interface without some similarities to an existing one. The reality in the business world is such that to avoid long and expensive trials, a company likely to be sued will tend to pay royalties even if the grounds for the suit are dubious. Interface copyright encourages the holder to threaten another developer with a suit that the holder could not win.
6. Interface copyright inhibits innovation. Usually, user interfaces are not the fruit of original, isolated, intellectual effort. Rather, they are the result of refinement and adaptation of a previous idea. Such were the cases of the Macintosh interface, which drew on ideas developed at Xerox, 1–2–3 which adapted VisiCalc's idea of the electronic spreadsheet, and many other useful interfaces. Interfaces result from an evolutionary, not revolutionary, processes. Users often prefer small, incremental, improvements to totally new concepts.

### 2.3. Arguments against software patents

Many people who object to granting patents for software do so for the same reasons they object to software copyrights. However, there are some arguments against patents per se. The major ones are:

1. Usually, a computer program does not reflect a new idea. It only automates a widely known process. A patent for the program is bound to cover the underlying idea, too, which may be obvious.
2. In the software development arena, communication among researchers and developers has been remarkably open. The race for a patent diminishes the freedom of communication among those who work on a certain technique.
3. Experience shows that software designers almost always perfect an existing program. There is a

continuous process of building new programs on the foundations of older programs. This competitive perfection benefits the public. Granting a patent for a program may discourage further development.

4. A patent increases the cost, and risk, of work in the area to which the patent relates, and discourages the entrance of new players, particularly individuals.
5. The software industry has flourished with a relatively small number of patents. Patents do not encourage progress.
6. Computers are used as a means of expression and as facilitators of information flow. Software patents may restrict the generation and flow of information.
7. The wide use of a program makes it obvious. Nonetheless, in some cases, a company applied for, and received, a patent for such a program or for a program that incorporates some of the features used in the already distributed program. The users became infringers overnight. The previous use is difficult to prove.
8. One program may infringe many patents. Patent searches are tedious and expensive. The fear that the developer may infringe unknown patents deters small entrepreneurs who cannot afford the expensive patent search.
9. Many innovations are the by-products of solving problems in the course of software development. The innovations do not occur with the purpose of applying for patents. When developing software that involves the same type of problems, an 'invention' is often reinvented multiple times, independently, by other developers. Patents for such 'inventions' limit future development of better programs.

These arguments suggest some dissatisfaction with the current situation.

## 3. Method

### 3.1. Questionnaire

A six-section questionnaire was developed. The first section included three questions regarding the preferred type and length of protection.

The second section focused on what to protect. It replicated the table presented in the 1992 Samuelson et al.'s table. However, in our study, information was gathered not only from software developers, but also from attorneys. For comparison purposes, column a of Table 7 shows the results of the 1992 study. Columns b and c summarize the responses in the current study.

The third section presented three statements regarding protection of different types of software. The respondents were asked to indicate, on a seven-point Likert scale, the degree to which they agree or disagree with the statements. Major revisions in system software are less frequent than in application software. Therefore, there may be a reason to afford system software greater protection than that given to application software. Thus, the first statement is:

1. System software should be afforded greater protection than application software.

Business applications may be perceived as more economically important than game software because they improve business processes. Also, there seems to be a greater variety and more sophistication in business applications than in game programs. Thus, the second statement is:

2. Business application software should be afforded greater protection than game software.

The rise of Microsoft as a major purveyor of standard operating systems and its legal skirmish with the US Department of Justice was accompanied by many application developers that cried foul play by the software giant. Many developers claimed that they were treated unfairly by a software power house which practically holds a monopoly. Thus, although this issue only marginally touches the main issue of the study, we posed the following statement:

3. Developers of system software should be required to license their software to all interested application developers (with equal contract terms for all licensees).

The fourth section was titled 'Your Own Opinion' and provided space for the respondents to express their own ideas about protection of software as intellectual property. We hoped that a qualitative analysis of the comments would provide novel ideas.

The fifth section raised the idea suggested by some jurists and software developers: a *sui generis* law for protecting software. There have been claims that software is unlike any other type of creation, and therefore

cannot be protected by laws designed to protect mechanical inventions, books, graphical art, or music. Thus, there may be a need for a special law that protects software intellectual property. Some argue that the law should grant protection for periods shorter than guaranteed in copyright and patent laws because the life cycle of software is relatively short. Tables 9–11 present the three questions we posed to solicit opinions on this idea, and a summary of the responses.

The sixth section was formulated differently for software developers and attorneys. It solicited biographical information about the respondents. A summary of this information is presented in Tables 1 and 2.

The questionnaire was accompanied with a sheet containing definitions of the different types of software (see Appendix A). The purpose of the this information was to create a common reference base for the respondents.

### 3.2. Sample

Names and addresses of software developers were taken from the 1995 SPA Membership Directory. The SPA (Software Publishers Association) is the largest US association of organizations whose main business is to develop and sell software. The directory includes some businesses who are involved in the software industry but do not develop software. These organizations were removed from the target sample. The questionnaires were sent to 592 organizations. The addressee in each organization was the person whose name appears in the SPA directory, usually the Chief Executive Officer. Fifty-three organizations responded, for a response rate of 9%. Although it is reasonable to assume that many respondents were actively involved in the development of software at the time they responded, some might have been in management positions at that time. It was our intentions to solicit their response as well, as they represented the opinions of software-developing organizations.

For the attorneys sample, two address lists of the ABA (American Bar Association) were used: intellectual property lawyers, and computer law lawyers. In all, 2000 attorneys were sent questionnaires. A total of 194 responded, for a response rate of 9.7%. The reason for the low response rate may be (1) the ubiquitous reluctance to fill out questionnaires, (2) the feeling that a response will not make any difference even if the individual disagrees with the current situation, (3) lack of interest in the subject, etc.

As is evident from Table 1, a great majority of software developers who participated in the survey are senior managers and the highest ranking officers in the firms. The opinions of such people are very important because of the important role that they play in the software industry and the weight that their

Table 1
Software developer profiles

| *Occupation* | % |
|---|---|
| Programmer | 0 |
| Systems analyst | 1 |
| Project leader | 4 |
| Junior manager | 2 |
| Senior manager | 33 |
| Highest-ranking officer | 60 |
| *Type of software developed* | |
| Systems software | 8 |
| Software development tools | 12 |
| Business | 27 |
| Entertainment | 17 |
| Education | 21 |
| Other | 15 |
| *Organization size* | |
| Independent consultant | 6 |
| Employed by organization with less than 20 employees | 30 |
| Employed by organization with 21–500 employees | 46 |
| Employed by organization with over 500 employees | 18 |

Table 2
Attorney profiles

| *Current legal specialty* | %[a] |
|---|---|
| Computer law | 36 |
| Copyright law | 22 |
| Patent law | 45 |
| Trade secret law | 16 |
| *Organization* | |
| Sole proprietor | 11 |
| A partner in a law firm | 39 |
| Employed by a law firm | 25 |
| Employed by a software-developing organization | 9 |
| Employed by a non-software-developing organization | 16 |

[a] Total $\neq$ 100 because some respondents indicated more than one category.

positions carry in public debates on the issue. From a perspective of formal training, this might be a group similar to the group that Samuelson et al. surveyed. But from a perspective of position in the organization hierarchy and influence in the software industry, this group is representative of the top echelon.[3] Table 2 provides a profile of the attorneys.

The software developers were also asked to report the approximate dollar amounts of software sales by their organizations, and the dollar amounts of their organization's software purchases. Thirty-five of them reported sales amounts, and thirty-six reported purchase amounts. Sales ranged US$ 100,000–800,000,000 with a mean of US$ 76 million and a coefficient of variation of 2.3. Purchases ranged US$ 2000–500,000,000 with a mean of US$ 14.8 million and a coefficient of variation of 5.6.

## 4. Analysis and discussion

### 4.1. Preferred type of protection

The first section of the questionnaire solicited opinions regarding the existing legal protection of software (Table 3). The first question was: ''If all software were to be protected under one of the current laws, which would you prefer?''

Clearly, software developers would prefer copyright as the single protection avenue. While a majority of attorneys would prefer copyright as well, over a third prefer patents. As expected, neither group regards trade secrets as a practical method of protecting software.

Table 3
''If all software were to be protected under one of the current laws, which would you prefer?'' (%)

| | Software developers | Attorneys | Entire sample |
|---|---|---|---|
| Copyright | 86 | 57 | 64 |
| Patent | 10 | 36 | 30 |
| Trade Secret | 4 | 7 | 6 |
| *n* | 52 | 188 | 240 |

[3]Samuelson et al. reported that they surveyed 'programmers' attending a professional conference.

The reasons for copyright preference by software developers may be the low cost, simple procedure, and short time involved in this avenue. Patents involve high legal costs and long periods of time (two to four years) to conclusion. The somewhat stronger support for patents in the attorneys group may stem from the fact that patents provide stronger protection. There might also be an element of financial benefit involved in this preference, because prosecution of patents involves significantly higher legal fees than registration of copyrights.

Patents are granted for twenty years. If the Patent and Trademark Office continues to grant patents to software developers, the great majority of attorneys would prefer the duration to stay the same, thus the high mean of 16.1 years, as shown in Table 4. Software developers would like to see the period shortened. Their mean response was 13.7 years. The difference between the groups is statistically significant at $P=0.0527$.[4] Examining the variance in both groups, one cannot ignore the much greater standard deviation among software developers (11.4) than among attorneys (6.1). It seems that, on this issue, software developers are not a cohesive group.

The frequency distribution of the responses drew our attention to the fact that attorneys who prosecuted patents tended to opt for longer patent periods. To further investigate this assumption, we performed an analysis of variance (ANOVA) of two groups: attorneys who classified themselves as practicing only patent law or patent law along with other classes, and attorneys who did not practice patent law at all. As is evident from Table 5, the difference in support for

Table 4
ANOVA: ''If the PTO continues to grant software patents, for how many years should the patent be granted?''

| | Mean | Std. Dev. | | DF | *P*-value |
|---|---|---|---|---|---|
| Attorneys | 16.1 | 6.1 | Attorney/software developer | 1 | 0.0527 |
| Software developers | 13.7 | 11.4 | Residual | 226 | |

[4]We acknowledge that the acceptable maximum *P*-value in social research is 0.05. By this criterion, the difference is not statistically significant.

Table 5
ANOVA: "If the PTO continues to grant software patents, for how many years should the patent be granted?" Patent vs. Non-Patent Attorneys

| | Count | Mean | Std. Dev. | | DF | *P*-value |
|---|---|---|---|---|---|---|
| Patent attorneys | 77 | 16.6 | 4.9 | Patent/non-patent attorney | 1 | 0.0002 |
| Non-patent attorneys | 96 | 13.3 | 6.1 | Residual | 172 | |

patent periods is significant ($p$=0.0002). Indeed, patent attorneys would like a longer period for patents (mean: 16.6 years) than non-patent attorneys (mean: 13.3 years).A similar investigation about the desired length of software copyrights reveals that a great majority in both groups favor the current period of 75 years for corporate copyright. Nobody in either group suggested periods longer than guaranteed in the current law, but a few suggested shorter periods; e.g., zero, five, ten, or twenty years. Remarkably, the response means (46.3 and 46 years) and standard deviations were almost exactly the same (30.2 and 31). Clearly, the groups were almost unanimous ($p$=0.9508) in their support of long periods of copyright protection for software (Table 6).

### *4.2. What to protect?*

The study by Samuelson et al. revealed that software developers had different preferences for legal protection of different software features. We adopted the same list of features as their survey. However, we decided to drop 'look and feel' because this term seems to be misunderstood and subject to different interpretations. In the legal literature it is usually interpreted as one or more of the user interface features. These features are already included in the list.

#### *4.2.1. Comparison of software developers: 1992 vs 1996*

To see if there have been any differences in the opinions of software developers over the years between the studies, we compared columns a and b of Table 7.

The comparison reveals an interesting difference: a smaller percentage of software developers selected 'Neither' protection. Apparently, the trend among software developers is to desire more protection for software. Except for Computer Generated Images and Source Code, support for protection of every software category has increased. In general, the 'no support' turned into support for protection under the copyright law.

There is also greater support for protection by patent of module design and algorithms. The latter is interesting, because for many years the Patent and Trademark Office refused to issue patents for any kind of algorithm due to a Supreme Court decision. The PTO now does grant patents for algorithms.

Of special interest are the User Interface categories, a major category around which legal battles revolved. It seems that the opinions of software developers are converging with court decisions (e.g., the case of Apple Computers against Microsoft). While an overwhelming majority of the 1992 audience advocated no protection at all for user interface commands, layout, sequence, and functionality, only about half of our sample concurred.

Since images are quite a clear case of art work which traditionally has been easily protected by copyright, it was surprising to see that a smaller proportion of our respondents supported any protection at all in our study than the proportion that supported protection in 1992.

Table 6
ANOVA: "For how many years should copyrights be granted for software?"

| | Mean | Std. Dev. | | DF | *P*-value |
|---|---|---|---|---|---|
| Attorneys | 46.3 | 30.2 | Attorney/software developer | 1 | 0.9508 |
| Software developers | 46.0 | 31.0 | Residual | 243 | |

Table 7
Support for Software Protection by Copyright or Patent: (a) Software Developers 1992, (b) Software Developers 1996, and (c) Attorneys 1996

| Support for | Copyright % | | | Patent % | | | Both % | | | Neither % | | | Number of Respondents | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | a | b | c | a | b | c | a | b | c | a | b | c |
| Source code | 86 | 66 | 52 | 2 | 4 | 16 | 3 | 23 | 24 | 8 | 7 | 8 | 318 | 53 | 194 |
| Object code | 65 | 72 | 54 | 2 | 4 | 17 | 3 | 13 | 22 | 27 | 11 | 7 | 293 | 53 | 194 |
| Pseudocode | 37 | 60 | 51 | 1 | 10 | 11 | 1 | 2 | 8 | 61 | 28 | 30 | 278 | 53 | 194 |
| Module design | 18 | 51 | 38 | 9 | 11 | 24 | 1 | 13 | 14 | 72 | 27 | 24 | 269 | 53 | 194 |
| Algorithms | 9 | 34 | 20 | 12 | 28 | 30 | 1 | 15 | 11 | 79 | 23 | 39 | 303 | 53 | 194 |
| User interface commands | 6 | 47 | 38 | 1 | 0 | 8 | 0 | 6 | 8 | 92 | 47 | 46 | 294 | 53 | 194 |
| Icons | 43 | 72 | 62 | 0 | 0 | 3 | 1 | 2 | 6 | 56 | 26 | 29 | 307 | 53 | 194 |
| User interface layout | 19 | 53 | 57 | 1 | 2 | 7 | 1 | 2 | 8 | 79 | 43 | 28 | 302 | 53 | 194 |
| User interface sequence | 9 | 38 | 31 | 1 | 2 | 20 | 0 | 9 | 9 | 90 | 51 | 41 | 295 | 53 | 194 |
| Look and feel | 5 | – | – | 0 | – | – | 0 | – | – | 94 | – | – | 312 | – | – |
| User interface functionality | 5 | 42 | 16 | 4 | 6 | 28 | 0 | 5 | 10 | 91 | 47 | 45 | 300 | 53 | 194 |
| Computer generated images | 81 | 66 | 72 | 1 | 0 | 2 | 0 | 4 | 8 | 18 | 30 | 18 | 316 | 53 | 194 |

Source of a columns: Samuelson, P., Denber, M., Glushko, R.J., 'Developments on the Intellectual Property Front,' 35 Communications of the ACM, No. 6 (June 1992). Percentages are rounded.

#### 4.2.2. Comparison of software developers and attorneys

Ostensibly, attorneys should be interested in strong protection of any software intellectual property, because this would involve them in the process of registration and prosecution. Software developers, on the other hand, must consider both sides of the issue: they benefit when their own creation is protected, but they are at a disadvantage when they use other developers' programs. So, either in the same person, or at least within a group of developers, these two tendencies are expected to 'cancel' each other. Thus, we expected to observe a greater overall support for protection on the part of attorneys; this is not what we found.

Considering protection (Copyright+Patent+Both) against no protection (Neither), there is not a great difference between software developers and attorneys. In both groups, a high percentage (developers – 93%, attorneys – 92%) supports protection of source code. A relatively high percentage supports protection of the object code of programs as well.

### 4.3. Protection by software purpose

Respondents were first asked to react to the statement: ''System software should be afforded greater protection than application software.'' The respondents circled a number on a seven-point scale where 1='Strongly Agree' and 7='Strongly Disagree.'

Overall, both attorneys and software developers agreed with the statement. As shown in Table 8, mean responses were in the fives with similar standard deviations. As the *P*- value is greater than the acceptable 0.05, we conclude that there is no difference between the groups.

Within the application software category, business software is sometimes perceived as more useful to the economy because it helps create new business. The respondents differed on the statement ''Business application software should be afforded greater protection than game software.'' Attorneys highly agreed with a mean response of 6.0, and they were quite cohesive with their opinion (standard deviation of 1.6), while software developers expressed a lower level of agreement, and the variance of their response was greater. With $p=0.042$, we may conclude that attorneys supported greater protection for business application than for game software.

The probe by the US Department of Justice of Microsoft's possible unfair trade practices raised a controversial issue: should a monopolist owner of system software and many applications be allowed to use its power to discriminate among other application developers who use its system software as a platform?

Table 8
Protection of software by purpose

| | Mean | Std. Dev. | | DF | *P*-value |
|---|---|---|---|---|---|
| ANOVA: ''Systems software should be afforded greater protection than application software'' (7-point scale) | | | | | |
| Attorneys | 5.3 | 1.9 | Attorney/software developer | 1 | 0.4851 |
| Software developers | 5.1 | 2.1 | Residual | 245 | |
| ANOVA: ''Business application software should be afforded greater protection than game software'' (7-point scale) | | | | | |
| Attorneys | 6.0 | 1.6 | Attorney/software developer | 1 | 0.0042 |
| Software developers | 5.2 | 2.1 | Residual | 245 | |
| ANOVA: ''Developers of system software should be required to license their software to all interested application developer (with equal contract terms for all licensees)'' (7-point scale) | | | | | |
| Attorneys | 5.1 | 2.1 | Attorney/software developer | 1 | <0.0001 |
| Software developers | 3.4 | 2.4 | Residual | 246 | |

The statement we posed in this regard was: ''Developers of systems software should be required to license their software to all interested application developers (with equal contract terms for all licensees).'' Since a great majority of software developers are in the business of creating application rather than system software, we expected to see strong agreement from developers. To our surprise, a majority of software developers disagreed with the policy, while attorneys supported it. At *P*-value<.0001, the difference between the groups is statistically significant. The objection to this idea in the software developers groups came despite the fact that only eight percent of them were involved in development of systems software.

### *4.4. Special law for protection of software intellectual property*

Some software developers and attorneys who have become frustrated with the current laws have raised the notion of a special law for protection of software. Software developers often complain that the legal profession, and especially the courts, lag behind technological development, and thus use legal doctrines that are often inappropriate. Many software professionals bitterly criticized court decisions like the infamous 'look and feel.' Therefore, we expected strong support for a special law among software developers and a conservative attitude among attorneys. Surprisingly, a majority in both groups supported the idea (see Table 9).

Proponents of a 'software protection law' often suggested that it should afford protection narrower than patent law but broader than copyright law with a relatively short period of protection to allow others to improve existing software. Table 10 indicates similar attitudes in the two groups: a very small minority for protection broader than patent; a small minority for narrower than copyright protection; and a large, absolute majority for protection broader than copyright but narrower than patent. Should a special law be adopted by the legislature, this convergence of opinions is important in giving a direction. The mere fact that the statement received such support may indicate the need for such a law.

Table 9
''Should congress pass a special law to protect software intellectual property?'' (%)

| | Software developers | Attorneys | Entire sample |
|---|---|---|---|
| Yes | 67 | 58 | 60 |
| No | 33 | 42 | 40 |
| *n* | 51 | 189 | 240 |

Proponents of a special law raised the period of protection as a major reason. An analysis of variance

Table 10
The scope of 'the software protection law' should be... (%)

| | Software developers | Attorneys | Entire sample |
|---|---|---|---|
| Narrower than copyright | 26 | 20 | 21 |
| Broader than copyright but narrower than patent | 66 | 73 | 72 |
| Broader than patent | 8 | 7 | 7 |
| *n* | 47 | 158 | 205 |

Table 11
ANOVA: "For how many years should protection under 'the software protection law' be granted?"

| | Mean | Std. Dev. | | DF | *P*-value |
|---|---|---|---|---|---|
| Attorneys | 13.7 | 12.6 | Attorney/software developer | 1 | 0.0006 |
| Software developers | 22.2 | 23.0 | Residual | 246 | |

yields a statistically significant difference between software developers and attorneys with regard to the number of years to protect software with a special 'software protection law.' As reflected in Table 11, the mean number of years that software developers would like is 22.2, while attorneys would settle for almost 14 years. Again, the standard deviations of the responses indicate greater cohesion in the legal profession than among software developers. In our sample, software developers desired protection periods that ranged from zero to 100 years.

### 4.5. Respondents' Written Opinions

Ninety attorneys (46%) and nineteen (36%) software developers provided written comments. To our disappointment, no novel idea transpired. The majority of the comments either criticized the current legal situation or said it provided proper protection. There was significantly more criticism in the software developers groups.

#### 4.5.1. Attorneys

Some attorneys pointed at the source of problems, e.g.: "The growth in personal computers is attributable to the lack of regulation and standardization of software. Innocent programmers ought not have to worry about patent infringement, and should be free to draw from known programming techniques, as long as they do not copy source code."

Others suggested classification of software and different protection to each class. One wrote: "[There] needs to be a distinction between application software that functions as a tool and software that is used much like an interactive book. The former, along with operating systems lends itself to patent-like protection, while the latter would be better protected under copyright laws. The author should have the choice, providing certain criteria are met."

Protection periods were a 'hot' issue. Several respondents suggested shorter periods, e.g.: "The protection term should be shortened from the usual patent/copyright paradigm. The developer is protected during the lead-time, but the software should not be held away from the public long enough to unduly stifle development and advancement in the industry. Terms of ten years are appropriate."

While many simply opined that the current laws are adequate, others wrote that changes should be made either due to the short 'life span' of software, or because of the great investment involved in patenting. For example, one attorney wrote: "Currently, smaller clients refuse to seek patent protection due to the fact that most software programs become obsolete quickly. Therefore, a quicker 'patent' process, maybe with a shorter protected term, would be helpful. Clearly defined software patenting rules would be helpful." Another wrote: "A new statutory scheme for protection of computer software is needed. Existing laws did not envision protection of matter such as computer software and are ill-suited for the purpose."

To solve this problem, many respondents supported a sui generis law. Some alluded to another special law. For example, one individual commented: "Just as the Semiconductor Chip Protection Act of 1984 protects chip design, a separate law should be passed to protect computer software inventions and codes."

And some accused the public of making much fuss over the issue due to misunderstanding. For example, one wrote: "Most of the perceived problems with the current patent and copyright laws have their roots in the ignorant statements that are regularly published in the press. If more people knew what the law is, there wouldn't be so much controversy."

With a look into a new mode for software sale through networks, one respondent wrote: "The recent position by the US Patent and Trademark Office is in the right direction. What wanted most is a realistic substitute for the 'shrink wrap' license to ensure that trade secrets are maintained on source and object code, etc. unless published in a patent or otherwise. Possibly a 'click' acceptance on the Internet will be interpreted by the courts as sufficient to show a confidential relationship vis a vis the software sent over the net."

*4.5.2. Developers*

Software developers were especially concerned with the overly long periods of protection and the bureaucratic hurdles in obtaining patents. Almost all of those who made comments suggested either short periods for software patents or none at all.

The president of one small company (annual sales: $2 million) opined: ''Over the next ten years software patent battles will destroy the US software industry. Most of the true innovators are with small companies that lack the resources to play the patent game. Many plan to leave the industry if threatened by patents. Also, the impossibility of doing appropriate patent searches on each of the thousands of algorithms in a given piece of software will result in total stagnation among those parties trying to honor the law.''

One software developer discounted the merit of protection and made a financial suggestion: ''Recoup R & D in the first distribution phase and then let go because technology turns within an 18 month product cycle. Protection and enforcement is not a solution.'' Unfortunately, the respondent did not elaborate on the idea. Questions remain: What is ''the first distribution phase?'' This phase may be different for different organizations and different products.

## 5. Conclusion

Majorities of both software developers and attorneys favor copyright as the legal method for protection of software intellectual property, but more attorneys still prefer patenting.

If the US Patent and Trademark Office continues to grant software patents, attorneys prefer a longer protection period. There is no difference between the groups with regard to the desired length of copyright protection. Majorities in both groups prefer the current period of 75 years, and the average number of years desired in both groups is 46 years.

By comparing information from 1992 and 1996 we learn that software developers now are more in favor of protection of user interfaces. This may be due to the proliferation of new small software businesses whose leaders feel that protection of their intellectual assets is essential for the success of their organizations.

Comparing opinions of today's software developers and attorneys, we found little differences between the groups regarding different categories of software. We also found high proportions of support for protection of source and object codes. When the issue was the purpose of the software, we found no significant difference for system software. Both groups prefer stronger protection for systems software than for application software. Attorneys, more than developers, favored greater protection for application software than for game software.

In light of suggestions that Microsoft held too much monopolistic power with its system software, we expected software developers to favor regulations that would force owners of systems software to offer their creation to any interested party with equal terms. To our surprise, there was significantly more support for this idea among attorneys than among the developers although only a small minority of the developers were involved in creating systems software.

Majorities in both groups support a special law for protection of software intellectual property. Majorities in both groups also agreed that the scope of such a law should be broader than that of copyrights but narrower than that of patents.

## Acknowledgements

This study was generously funded by the Richard L. Barber Fund for Interdisciplinary Legal Research.

## Appendix A

### Technical terms

**Source Code:** The program written in the original high-level language, e.g., COBOL or C, in which the instructions are easily readable by a programmer who possesses a basic knowledge of that programming language.

**Object code:** The machine-language equivalent of the source code. To obtain an object code version of the program, a special program called compiler is used to convert the code. While source code can easily be modified by anyone who masters the programming language, the object code is virtually gibberish to anyone who does not know machine language. For

this reason, software developers usually offer for sales only the object code, not the source code.

**Pseudocode:** A generic description of what the program does, written in simple English rather than in any specific programming language.

**Module design:** A graphical description of a part of the program that would perform a distinctive function.

**Algorithms:** Mathematical sets of steps that a program follows to achieve a goal.

**User interface commands:** Commands that the user enters into the computer to invoke a specified reaction. The commands trigger other, internal commands in the program.

**Icons:** Small pictures on a computer screen that the user can click via an input device called mouse instead of inputting a literal command.

**User interface layout:** The arrangement of commands, icons, and other means of input on the computer screen.

**User interface sequence:** The order in which the user instructs the computer to perform a task, via commands, icons, or other input means.

**User interface functionality:** The effectiveness of the part of the program with which the user interacts.

**Computer generated images:** Graphical creations produced through the use of a computer.

## References

[1] C.H. Nadan, Comment: A Proposal to Recognize Component Works: How a Teddy Bears on the Competing Ends of Copyright Law, California Law Review 78, 1990, pp. 1633.
[2] E. Oz, Ethics for the Information Age, Wm.C. Brown, Dubuque, IA, 1994.
[3] N.P. Terry, GUI Wars: The Windows Litigation and the Continuing Decline of Look and Feel, Arkansas Law Review 47, 1994, pp. 93.
[4] P. Samuelson, M. Denber, R.J. Glushko, Developments on the Intellectual Property Front, 35 Communications of the ACM, No. 6, June 1992, pp. 33–39.
[5] R. Stallman, S. Garfinkel, Against user interface copyright, Communications of the ACM 33(11), 1990, pp. 15–18.

EFFY OZ, associate professor of management science and information systems at Penn State University, Great Valley, received his doctorate from Boston University. His research and teaching interests lie in the areas of strategic information systems, ethical issues in information technology, the impact of information technology on decision making, and management of information technology.

Dr. Oz was an executive for a large aerospace and electronics corporation for eleven years. His articles have been published in *Decision Science, MIS quarterly, Communications of the ACM, Information & Management, OMEGA, and Journal of Business Ethics*, among other academic and professional journals. He is the author of four books and several book chapters. In 1997, Dr. Oz was a co-recipient of the Notable Contribution to the Information Systems Literature Award by the Information Systems Section of the American Accounting Association. He was a member of the editorial board of *Journal of Systems Management* and is now a member of the editorial review board of *Journal of Global Information Technology Management*. He has given professional seminars to managers of various organizations and frequently speaks to professional groups.