Management of Computing

Bob Zmud Editor

WHY THE LOOK AND FEEL OF SOFTWARE USER INTERFACES SHOULD NOT BE PROTECTED BY COPYRIGHT LAW

An attorney looks at how copyright law is applied to the protection of software user interfaces and makes a strong case for reevaluating the way the law views software.

PAMELA SAMUELSON

Until Lotus and Apple sued their competitors for copyright infringement, software user interface designers did not think much about whether intellectual property law might impose some constraints on their design decisions. Even those who knew that copyrights were available to protect software probably understood this to mean simply that a person would have to write his or her own code to develop a program similar to an already copyrighted one.

Given the proliferation of computer hardware clones, the apparent legality of hardware clones, and the extent to which competition and innovation have been flourishing in a market full of hardware clones, it was not surprising that software clones would appear on the market as well. The computing community tends to think that clones—hardware or software—perform a valuable service in the market: providing additional product choices to consumers, price competition, and/or improved product features.

User interfaces, the most visible aspects of software products and the key to their usability, have been among the features of software products that software cloners have chosen to emulate. It made sense to incorporate into one's own products aspects of user interfaces that seemed to work well in already commercially successful software. The success of these products indicated that users had learned and come to be comfortable with a particular style of interface for a particular type of software, and generally expected or wanted the other software producers to use the same or similar user interface in their competitive software. In addition, it seemed like everyone in the field made it a practice to borrow from everyone else's user interfaces. The many conferences held each year to report on advances in human factors engineering for computing systems, attended by industry people as well as researchers, have been forums for sharing knowledge about what works and doesn't work in user interface designs. In the first six years of ACM's annual Conference on Human Factors in Computing Systems, there were no sessions raising legal protection issues. The norm at these conferences has been to share user interface design improvements, not to claim property rights in them.

The Lotus and Apple lawsuits have made user interface designers aware that they can no longer safely ignore the intellectual property implications of user interface design. Yet until these cases are decided, it is hard for those in the industry to know how they are supposed to behave. If the look and feel of the Lotus and Apple interfaces are held to be protected by copyright, such protection will have a profound effect on the industry.

While not all of the questions raised by the Lotus and Apple lawsuits can be answered in one article, this article aims to address the most pressing ones.

THE LOTUS AND APPLE LAWSUITS

In 1987 Lotus Development Corp. filed two copyright infringement lawsuits in a federal court in Massachusetts. One was against Mosaic Software, developer of the Twin software product that competes with Lotus'

1-2-3 spreadsheet program. The other was against Paperback Software, developer of VP-Planner, also competitive with 1-2-3. Both lawsuits focus exclusively on similarities of the user interfaces of the programs.

Lotus makes two kinds of complaints against Mosaic and Paperback: one, that the defendants' programs have the same look and feel as the Lotus 1-2-3 program, and secondly, that specific aspects of the Mosaic and VP-Planner user interfaces are substantially similar to the Lotus 1-2-3 interface and were unlawfully copied from the Lotus interface. (This article will discuss only the look and feel claim, but it is worth pointing out that Lotus thinks it will be able to win the lawsuit if it proves either the look and feel charge or the specific aspects charge.)

Among the specific aspects of the 1-2-3 interface that Lotus claims its copyright protects are: the instruction, command, and menu language of Lotus 1-2-3, the macrocommands and syntax of Lotus 1-2-3, the format and structure of the Lotus textual displays, and the sequence of screen displays used by Lotus 1-2-3. Lotus also claims that by deliberately setting out to be clones of the popular Lotus program, Mosaic and Paperback were intending to infringe the Lotus copyright. It is expected that this case will go to trial during the Fall of 1989.

In the spring of 1988, Apple Computer Corp. brought a copyright infringement lawsuit against Microsoft and Hewlett-Packard, claiming that these defendants had adopted substantially similar graphic and visual elements in their Windows and New Wave software products to those used by Apple in several of its copyrighted programs. Claiming to have invested millions of dollars in creating a highly distinctive, aesthetically pleasing user interface style, Apple claims that other firms are not free to appropriate this style of interface without Apple's permission.

A ruling in Apple's favor would seem to have a broader reach than a ruling in Lotus' favor, for Apple seems to be claiming copyright protection for a style of interface, not just for the interface of a particular program. It is significant that Apple's complaint, unlike Lotus', does not identify specific features of the Apple interfaces that Microsoft and Hewlett-Packard have misappropriated from Apple. Thus, though Apple does not specifically mention look and feel in its complaint, the Apple case may be even more of a look and feel case than the Lotus case, which explicitly tries to be one.

DOES COPYRIGHT LAW PROTECT THE LOOK AND FEEL OF SOFTWARE?

Lotus and Apple have charged their competitors with violating the copyright law because the competitors developed software with the same look and feel as the Lotus and Apple software. Given these charges, it is fair to ask: Does copyright law protect the look and feel of any copyrighted work? More particularly, does copyright protect the look and feel of software? And what is the legal definition of look and feel anyway?

The short and simple answer to these questions is that look and feel has virtually no standing in copyright law as to any category of protected work. Although there are a couple of cases involving copyrighted fabric designs which contain some discussion of similarities in the look and feel of the fabrics, look and feel has no precedent as a copyright standard.

Look and feel is a phrase invented by two lawyers named Jack Russo and Doug Derwin who wrote an article in 1985 for a computer law magazine about aspects of software user interfaces that the authors thought might be protectable by copyright. It was the first legal article to focus attention on software user interface protection, and had influence on practitioners in the software copyright field. In some sense, the Lotus and Apple cases are the fruit of the intellectual labor of these authors.

Because Russo and Derwin made up the look and feel theory of software user interface protection, there is no legal definition (as where a statute or a judicial opinion had said what the phrase meant) of look and feel—not as to software user interfaces, not as to anything else. Some have speculated that *look* refers to the appearance of the screen displays (i.e., the visual layout of words and pictorial features of the screen), and that *feel* refers to how the program interacts with the user when performing its functions (i.e., the sequence of functions that occur when the user selects an option made available on the screen). Because there is no legal definition of look and feel, and because it is inherently vague as an accusation, a copyright infringement lawsuit based on look and feel may be hard to fight. One can never be sure what one's opponent is claiming one has stolen from him or her.

Because there is no legal definition of look and feel, and because it is inherently vague as an accusation, a copyright infringement lawsuit based on look and feel may be hard to fight.

Though look and feel has no standing in the copyright law, a kindred phrase—namely, *total concept and feel*—does have some standing in the law. As a descriptive term about what kinds of similarities between works can be infringing similarities, total concept and feel has been adopted by one appellate court (the one that includes California). But even that court seems to have limited the scope of the doctrine to cases involving artistic or fanciful works. Some think that this court has abandoned the standard, in view of the strong criticism it has received from other courts and commentators.

One reason that total concept and feel as a test for copyright infringement has been criticized is that the

copyright statute specifically states that concepts—total or otherwise—are not to be protected by copyright law. (The same statutory provision also prohibits copyright protection for systems, procedures, processes, methods of operation, discoveries, principles, and the like; all of these things are what copyright law considers to be the ideas, that is, the unprotectable elements, of a copyrighted work.)

That concepts are forbidden subject matter for copyright protection may explain why software copyright lawyers have substituted *look* for *total concept* in the new round of software user interface infringement cases. Look is not among the things that copyright statute says is unprotected, yet look shares with total concept the fuzzy unspecificity that is, from a plaintiff's standpoint, the major advantage of the standard. Look and feel, therefore, has the same advantages as total concept and feel without its vulnerability to a statutory attack.

Total concept and feel got whatever standing it has in the law from cases involving artistic or fanciful works—the kinds of works as to which it is sometimes difficult to pin down with precision what exactly has been misappropriated from a protected work. The major case endorsing this doctrine is a case in which a McDonald's advertisement featured characters resembling the H. R. Puf'n'stuf characters created by the plaintiff. If one confined one's study of the McDonald's commercial to listing the similarities and differences between the characters (copyright lawyers call this process analytic dissection), the judges recognized that one could lose sight of the overall impression of the two works, and could thereby overlook real and significant similarities between the works. Even if the list of differences was longer than the list of similarities, McDonald's should not escape liability, the court thought, if the total concept and feel of the two works was the same.

The McDonald's case was not the first time that judges in copyright infringement cases have grappled with the dilemma of how much weight to give to analytic dissection and how much weight to give to overall impressions. To resolve this dilemma, judges have invented a two-step procedure for use in copyright infringement cases. One step concentrates on analyzing similarities and differences (often with the aid of expert testimony). The other step concentrates on overall impressions. In general, expert testimony is inadmissible as to overall impressions. The judge or the jury is supposed to make this judgment based on what copyright law says is the firmer, if more naive basis of their experience. Under this two-step test, analytic dissection is a preliminary step to finding infringement; an overall impression judgment is the final step.

Total concept and feel is just a way of expressing this second overall impression step. Thought it is somewhat awkward, this two-step process of analyzing copyright infringements reflects a small profundity of human experience—that if you chop things up very finely and look at each thing separately, you can lose sight of the overall character of similarities and differences. Copyright law does not want thieves to escape by making immaterial variations. Still, it is worth emphasizing that the impressionistic judgment called for by copyright law is supposed to be made on the heels of the analytic dissection, which means that it is an error to start and finish with total concept and feel, or its variant, look and feel, even in California.

Even the appellate court that endorsed the total concept and feel approach in the McDonald's case has limited its use. When someone recently argued to this same court that the total concept and feel of his copyrighted car radiator catalog had been copied by a competitor, the court said that a total concept and feel approach was inappropriate in this kind of case. Other judges have agreed that total concept and feel should be used only when dealing with artistic or fanciful works.

The radiator catalog case is consistent with a long tradition in the copyright caselaw of distinguishing among different types of copyrighted works, and giving very broad protection to artistic and fanciful works, significantly narrower protection to factual works, and giving very thin protection indeed to functional writings, such as engineering drawings, insurance forms, and accounting ledger sheets. (Truly functional works, such as machines, are ineligible for copyright protection because they are potentially patentable.)

Though functional writings, such as engineering drawings, are copyrightable, the scope of copyright protection for them is so narrow that it generally takes an identical or near-identical competitive product to infringe a copyright in these works. This is because it is the role of the patent law to protect innovative processes, systems, procedures, and the like, and the role of copyright only to protect expressive elements of their depiction. So as not to tread on patent law's domain, copyright law considers the engineering design depicted in the drawing, for example, to be the idea depicted in the drawing. As a consequence, judges tend to use much more analytic dissection and much less overall impressionism in making copyright infringement decisions in cases involving functional writings than in cases involving artistic works.

Because computer software is a functional work, it might seem obvious that this more cautious analytic approach should be used in software user interface cases, but that has not been so. In the few software copyright infringement cases decided thus far, judges have been unaware of the functionality of user interfaces. In the Broderbund case, the judge asserted that the Broderbund interface was artistic, explicitly denying that the interface had any functional character, and ruling that because the Unison interface had the same total concept and feel as the Broderbund interface, Unison had violated Broderbund's copyright. (See "Copyright Decisions on Protection of Software User Interfaces.") Although the judge in the later Softklone case did a more careful dissective analysis of the software user interfaces involved in that case, toward the end of his opinion, this judge added that the total concept and feel of the interfaces was the same.

Because of these precedents, the defendants in the Lotus and Apple cases may find it hard to persuade the judges not to use a total concept and feel (or look and feel) approach, and to move the judges toward a more dissective and analytic comparison of similarities and differences. Apple's strategy of emphasizing the artistic and visual character of its interface is aimed at ensuring that the judges who decide its case will follow the pattern set in Broderbund. It is no coincidence that the Apple lawsuit was filed in the same court in California where the Broderbund case was decided. Lotus may have a tougher time winning its lawsuit not only because its user interface is more functional, and less graphic and pictorial, than the Apple interface, but the Lotus lawsuit is also taking place in Massachusetts where a total concept and feel approach has not been endorsed in any case, let alone in a software user interface case.

REASONS JUDGES COULD USE TO REJECT A LOOK AND FEEL TEST

If the judges in the Lotus or Apple cases were looking for good reasons to reject a look and feel test for copyright infringement, it would be easy to find them. As mentioned earlier, total concept and feel, and its cousin look and feel, have little or no standing in the copyright law, and none at all as to functional writings. If the judges were prepared to recognize the functionality of many aspects of the interfaces at issue in these cases, that alone would preclude use of a look and feel test.

But apart from the technicalities of copyright doctrine, it would be wise for the judges in these cases to reject a look and feel approach to determining copyright infringement for user interfaces because such a test makes it virtually impossible to edit out many things about the interfaces that copyright is not supposed to protect.

Originality Issues

For one thing, a look and feel test for copyright infringement does not allow one to edit out those aspects of an interface that are not original to the copyright owner. Section 103 of the copyright law says that a copyright gives an author rights to protect only that which is original to him or her. This is significant in the software user interface cases because much of what Apple now claims as distinctive features of its MacIntosh user interface style were derived from user interface designs and concepts developed at Xerox's Palo Alto Research Center. Similarly, much of what Lotus now claims to own about the Lotus 1-2-3 user interface was adopted by Lotus from the successful Visicalc program, the first prototype of the spreadsheet program. If unoriginal material is not protectable by copyright, then it should not be inadvertently swept back in by application of a look and feel test.

Layouts and Other Uncopyrightable Features

Apart from what might be unoriginal material, there may be a number of other unprotectable features of the

Copyright Decisions on Protection of Software User Interfaces

There have been four judicial decisions thus far that have a direct bearing on the protectability of software user interfaces through copyright law. The cases are not in agreement with one another which is one of the reasons why it is hard to predict how the Apple and Lotus cases will be decided. Though the cases sometimes reflect little understanding of software and interfaces, the tale of how the judges have struggled to apply copyright law to software user interfaces is an interesting one.

SYNERCOM CASE

The first in the sequence of software user interface cases is *Synercom Technology, Inc.* v. *University Computing Co.* It was decided by a trial judge in a Texas federal court in 1978. Synercom, the plaintiff, had developed a structural analysis program for use by engineers. Synercom's user manuals described the order in which data representing the variables about the structure to be analyzed had to be entered for the program to run successfully. Engineering Dynamics, one of the defendants, developed a competitive structural analysis program which used the same data input format as the Synercom program, though Engineering Dynamics included the input formats in its preprocessor program, rather than in the user manual. At trial Synercom's lawyers were able to show that although there were hundreds of structural analysis programs available on the market, only Synercom's and Engineering Dynamics' input formats were the same. Moreover, it was clear that Engineering Dynamics had adopted these formats in order to compete more successfully with Synercom's program.

Engineering Dynamics convinced the trial judge that Synercom's input formats should no more be protected by copyright than would the H pattern for car stick shifts. Although the H pattern might have been chosen randomly from a variety of alternatives, and could be expressed in a variety of forms (such as a prose description, a diagram, or a photograph), each of which could be copyrighted, the copyright in any of these instantiations would not, under traditional copyright law, extend to the H pattern itself. The first manufacturer of cars to copyright a diagram of the H pattern who also used the pattern in its cars would not be able to get the exclusive right to make cars with this design for a stick shift. Rather, the H pattern would be the idea depicted in the diagram which copyright law would regard as unprotectable. It would take a patent to give the manufacturer the exclusive right to make cars with an H pattern for the stick shift.

In ruling that Synercom's input formats were not protectable by copyright, the judge was trying to be consistent with an 1879 Supreme Court case, *Baker* vs. *Selden*, that is the classic case articulating the difference between what is protectable expression in a copyrighted work and what is unprotectable idea. Selden had sued Baker for copyright infringement because Baker had included similar sample ledger sheets in his book about the accounting system that Selden had invented and written about in a separate book. The Supreme Court ruled that Selden's copyright in the book did not give him a right to stop Baker from writing a book about the same accounting system and including sample ledger sheets in it. It would have taken a patent to give Selden such broad rights against Baker. Under copyright law, the accounting system was the unprotectable idea in Selden's book. Because the similarities in the two books' ledger

sheets were due to their being designed for use in connection with the same accounting system, the Supreme Court ruled that these similarities did not constitute infringement.

Engineering Dynamics also successfully argued that, just as users of automobiles benefited by standardization of stick shifts because it reduced the retraining that would be necessary if all automobiles had different stick shift organizations, users of statistical analysis programs would also benefit from the consistency between the input formats utilized by Synercom and Engineering Dynamics.

WHELAN CASE

The *Whelan Associates, Inc.* vs. *Jaslow Dental Laboratories* case, decided in the summer of 1986 by a federal appellate court, is sometimes described as the case that established that the look and feel of software user interfaces can be protected by copyright. Curiously enough, Whelan is not really a user interface case at all, nor is it a look and feel case. However, the judicial opinion announcing the reasons the judges thought that Whelen's copyright was infringed contains some broad vague language which has given rise to an expansive interpretation of the meaning of the case.

Rand Jaslow, one of the defendants in the case, had hired Elaine Whelan to develop a program that would computerize the way Jaslow's lab had been conducting its business. Both of them contemplated that the program would be marketed to other labs as well. Jaslow agreed to pay the cost of the development effort, and to let Whelan own the rights in the program, subject to a royalty back to Jaslow on sales to other labs. Jaslow gave Whelan, who had no prior experience with dental lab business operations, extensive access to his lab and also helped her design the user interface for the program so that it reflected his business methods.

The Dentalab program, written in the now obscure programming language EDL (Event Driven Language) to run on an IBM mainframe, was delivered to Jaslow in 1979. A few years later, Jaslow decided there would be a market for a program of this sort that would run on a personal computer, so he undertook to develop his own program in BASIC. There is no question but that Jaslow studied how Whelan had organized her program before writing his own program, but he didn't make use of any of her code. When Jaslow began to market his PC program in competition with Whelan's program, Whelan sued him for copyright infringement.

From the appellate court decision in the case (decided in the Third Circuit Court of Appeals, which includes New Jersey, Pennsylvania, and Delaware), it is clear that Elaine Whelan was not asserting copyright infringment based on similarities between the user interfaces of the two programs. Rather, her claim was that Jaslow had copied the underlying structure of the Dentalab program, and that the overall structure of the program was part of what her copyright protected.

Nevertheless, user interface similarities seem to have been an important factor in the case. If one reads the trial judge's opinion carefully, it appears that the judge did not really grasp the difference between the programs and their interfaces, and thought that if the two user interfaces looked alike, that must mean that the programs were similarly structured. The judge was very impressed by the similarities in the screen displays produced by the two programs (which should not have been surprising, since Jaslow himself designed both of them), and considered these similarities very damning in view of the fact that Whelan had produced evidence that there were other dental lab business programs which had very different screen displays.

Although the appellate court decision recognized that two programs can have very different underlying structures and still produce similar screen displays, the appellate court still permitted screen display similarities to be considered as evidence of underlying program similarities. And the appellate judges relied on the trial judge's finding that there were other ways to structure the software besides the one Whelan and Jaslow had used as conclusive evidence of infringement, even though the trial judge had only seen the screen displays of these other programs, not their underlying structure.

It is partly because the judges in the Whelan case were somewhat confused about the relationship between the underlying structure of a program had its users interface that some have thought of *Whelan* as a user interface case, even though the only issue the court directly ruled on was the protectability of the overall structure of the underlying program. But the main reason that the Whelan case has been thought to protect user interfaces by copyright is that the "test" for software copyright infringement which the appellate court used in *Whelan* is broad enough to cover virtually all aspects of software user interfaces.

The test announced and applied in *Whelan* was that the general purpose or function of a copyrighted computer program was to be considered its unprotectable idea. Everything else about the program was to be considered protectable expression unless there was only one or a very small number of ways to achieve that purpose, in which case that one way would also become part of the program's unprotectable idea. That is, a particular way of implementing a general function would generally be considered protectable expression in a program, unless there were no other alternative implementations, in which case that implementation would become part of the idea, too. The judges in the Whelan case thought that the judge in the Synercom case had been wrong in his analysis of the issues.

Applying this test to the case at hand, the appeals court said that the general purpose or function of Whelan's program, which was to organize dental laboratory business functions, was its "idea" which Jaslow was free to use as he saw fit. The structure of the program was expression because it wasn't part of the general purpose or function of the program and because there were other ways for Jaslow to have structured such a program besides the way that Whelan had structured hers, so it was infringement for Jaslow to have used a similar structure to Whelan's.

BRODERBUND CASE

The next case in the series of software user interface cases is *Broderbund Software, Inc.* v. *Unison World, Inc.*, which was decided in a federal trial court in California a few months after the Third Circuit decision in *Whelan*. *Broderbund* was similar to the Whelan case in a number of respects. As in *Whelan*, there were no similarities in the underlying code of the two programs. As in the Whelan case there were similarities in the program screen displays. Also, as in *Whelan*, there had at one time been some collaboration in the software development process between the plaintiff and defendant which ultimately broke down. As in *Whelan*, the plaintiff in the Broderbund case won the copyright infringement suit.

What was different about *Broderbund* and *Whelan* was that in *Broderbund*, it was similarities in the screen displays, rather than in the structure of the underlying program, that was the basis of the infringement claim. Broderbund and Unison had both developed programs that were useful for designing greeting cards, banners, and the like. Both programs featured the same basic format for their menu screens and sets

of choices per menu, although there were some differences in the graphics and wording of the menu screens.

Relying on the Whelan case, the judge in the Broderbund case said that the menu screens displayed by the programs were protectable by copyright. The court rejected arguments that there was anything functional about the screen displays generated by these programs, and frequently emphasized the artistic, aesthetic, stylistic nature of the menu screen designs, and the alternative-expressive possibilities available to the defendant. At one point in the opinion, for example, the judge took Unison to task for using precisely they same words ("choose a font") in one of its menus as Broderbund, giving examples of some of the alternative possible expressions ("select a font," "indicate a typeface preference," or "which style do you prefer?"). As in *Whelan*, it was because the defendant had chosen to do something the same as the plaintiff when there were other ways to do it that the judge thought there was infringement.

The Broderbund case is also notable in the software copyright caselaw for use of a kind of look and feel approach to judging copyright infringement based on user interface similarities. Relying on some prior decisions of the Ninth Circuit Court of Appeals (which includes California) that had involved copyrights for artistic or fanciful works, the judge in *Broderbund* stated that the total concept and feel of the Broderbund and Unison screen displays was the same, and therefore, Unison had infringed Broderbund's copyright. Although total concept and feel is a controversial test for copyright infringement under any circumstances (because the copyright statute says that concepts cannot be protected by copyright), even the Ninth Circuit seems only to apply it where the work is artistic or fanciful, which was why it was so important that the judge in *Broderbund* characterized the user interface in the case as an artistic work, and almost completely denied its functionality.

SOFTKLONE CASE

The fourth in the sequence of software user interface copyright cases is *Digital Communications Associates, Inc.* v. *Softklone Distributing Corp.* This case was decided in 1987 in a federal trial court in Georgia. As in *Whelan* and *Broderbund*, the plantiff won the case, but this time the plaintiff got a more limited verdict than was sought, with important consequences for Lotus.

DCA was the owner of a copyright in a popular data communications program called Crosstalk XVI. Softklone developed a competitive program called Mirror. The sole basis of the infringement charge was the similarities in the main menu (or status) screen which displayed the set of available commands, as well as several categories of values that had to be set by the user for the program to perform its major function which was to facilitate the transmission of data between two otherwise incompatible computers or software systems. (There was, for example, a speed category which permitted the user to set the baud rate which would allow data from one computer to be transferred to another.)

There were three types of similarities between the Crosstalk and Mirror status screens. One was that both programs had the same set of commands and values (e.g., both used "speed" for the baud rate). Secondly, the first two letters of the commands and value names were capitalized and highlighted (e.g., SPeed) on both status screens. Thirdly, the value terms (but not the command names) were grouped in an identical manner (e.g., both programs had a "send control setting" category, which had subsets of CWait, LWait, None).

The judge in the Softklone case decided that it was not infringement for the two programs to have the same set of command and value-setting terms. The judge likened this to the input formats of the Synercom case, and said the terms were not protectable by copyright. But the judge found that the capitalization and highlighting device and the groupings were part of the protectable expression of the Crosstalk status screen because there were other ways to do these things than the way that Crosstalk had done. Rather than capitalizing the first two letters of the commands, for example, Softklone could have capitalized the last two or the middle two letters, or it could have capitalized all but the first letter, or any one of a number of other possibilities. Also, the judge said there were almost an infinite number of ways that the value terms could have been grouped, so the fact that Mirror grouped its values in the same was as Crosstalk had was proof of infringement. The judge perceived these features of the program to exhibit "considerable stylistic creativity and authorship above and beyond the ideas in the status screen."

What is curious about the judge's decision on the capitalization and grouping issues was that it appears the judge did not seem to understand the functionality of both devices. Crosstalk capitalized and highlighted the first two letters of the commands and value terms as a way of signaling to the user that it was only necessary to type the first two letters to invoke the command or set the value. Because there were 78 terms on the status screen, even had it been possible to come up with arcane command names beginning with x or z, there would have been too many command and value terms on the screen to use the most common designation device used in software user interfaces, namely, capitalizing the first letter of the command term. Sure, Mirror could have used the first three letters or the last three as a signaling device, but that would not be as efficient as using the first two letters. The judge seemed to think that the capitalization of the first two letters was simply a stylistic expression of the author.

Similarly, the judge did not seem to grasp the reason for the similar groupings in the status screens—there was a functional relationship among the terms within the group that made it appropriate to group them together. (It made sense, for example, to group speed, data, and port together because they were related values that needed to be set before other functions were invoked.) Of course, it would have been possible to group speed with help; it just wouldn't make any sense to do so.

Although the judge in the Softklone case was quite specific about what aspects of the Crosstalk status screen were protected expression and why, toward the end of the opinion the judge went on to say that the total concept and feel of the two status screens was similar, as a way of reinforcing his conclusion of infringement, thereby endorsing the more vague impressionistic test used by the judge in the Broderbund case.

WHERE THINGS STAND NOW

Neither the *Broderbund* nor the *Softklone* decisions was appealed to a higher court. Because both seem to endorse a kind of look and feel approach to copyright infringement based on screen display similarities, the defendants in the Lotus and Apple cases will be waging an uphill battle to avoid use of such an approach in their cases. They will also be waging an uphill battle to establish the functionality of the interfaces (or certain aspects of them) because the judges so far seem to perceive screen displays in more artistic terms.

Lotus and Apple interfaces that a look and feel test for user interface infringement won't edit out. The U.S. Copyright Office has issued a policy statement on the extent of copyright protection for computer programs which states that layouts and other screen displays that resemble blank forms are not protectable by copyright. Other aspects of interfaces, such as use of a command line at the top of the screen or a menu listing of commands, may be so simple and standard that no copyright protection should be available to them. Because the Lotus and Apple cases involve some layout and simple standard feature similarities, the test of copyright infringement in these cases should ensure that similarities of these sorts are not considered, and a look and feel test won't offer this assurance.

be recognized as narrow in order not to conflict with the scope of patent protection for interfaces which is in the process of becoming clarified.

Human Factors Engineering

User interface design for software is largely a human factors engineering design process. There is a long history of protecting the products of human factors engineering by patent law, and no history of protecting it through copyright. Improved machine display panels, such as an improved automobile dashboard, that allow the machine's users to make more efficient and effective use of the machine are patentable machine improvements. No one would ever have thought to copyright a design for an automobile dashboard because it would so obviously not be copyrightable on account of its utility.

Software user interface patents are becoming more common. IBM, for example, has a patent on a method of highlighting portions of the text displayed on a screen.

Patent Issues

But more important even than editing out unoriginal material and simple layouts is removing from copyright consideration features of the Lotus and Apple user interface that are functional enough to be eligible to be protected by patent law. It is a fundamental principle of federal intellectual property law that that which is patentable subject matter is ineligible for copyright protection (See sidebar.) It would undermine the policies underlying patent law if one could protect through copyright law aspects of user interfaces that failed to be inventive enough to qualify for patent protection.

Several articles have been written recently by patent lawyers who assert that the look and feel of software user interfaces, including perhaps aspects of the Lotus and Apple interfaces, are patentable. Software user interface patents are becoming more common. IBM, for example, has a patent on a method of highlighting portions of the text displayed on a screen and permitting certain functions to be performed on the highlighted material. Apple has a patent on the pulldown menu for use with a mouse.

Unfortunately, judges in the software copyright cases, as well as many copyright commentators, are oblivious to the fact that patents are available for user interfaces, causing many judges and commentators to interpret the scope of copyright protection very broadly as if there was no other law to provide protection for software user interfaces if copyright did not reach it. Although some software copyright lawyers are now arguing that dual copyright and patent protection for software is acceptable, this position is not consistent with the long tradition of exclusivity of patent and copyright subject matters. (See "The Odd History of Copyright and Patent Protection for Computer Software.") If we were to be consistent with past copyright and patent tradition, the scope of copyright protection for user interfaces would

Yet human factors engineering for software seems thus far, at least in the copyright context, to be treated differently. There seem to be several reasons for this. For one thing, the judges in the cases so far have not recognized that user interface design is a human factors engineering process, any more than they have recognized that software user interfaces are largely functional in nature. This is not just the fault of the judges. It was Congress that decided to put software (a technology) into a body of law (namely copyright) which not only had no experience protecting technologies, but which had a history of being antithetical to the protection of technologies. To make matters worse, Congress did not understand that software was a technology when it put computer programs into the copyright system. Much of the confusion of computer software copyright law has come from this willful ignorance of the technological dimensions of software and the willful insistence on treating software as if it was just another writing. But in addition, software user interfaces, unlike the user interfaces for other machines, tend to make greater use of words and visual symbols to achieve the human–machine interaction desired, which is what makes copyright seem so much more applicable.

If the old copyright rule against protecting engineering designs and engineered products was applied to software, little about software besides the object code would be protected. In the user interface context, human factors design features would have to be edited out. A look and feel test obviously doesn't permit that to happen. If the look and feel (and other products of human factors engineering) of software interfaces were protected through patent law rather than copyright, only the more significant advances in the state of the

The Odd History of Copyright and Patent Protection for Computer Software

The present legal controversy about whether the look and feel of software user interfaces are protectable by copyright can best be understood in the larger context of the history of patent and copyright protection for software. The oddness of the history of software protection derives from the fact that software is both a writing and a machine in a legal system that has assumed that something could be either a writing or a machine, but could not be both. Because software is a hybrid in a system of law that assumes that such hybrids can't exist, the legal system has experienced considerable difficulty in integrating software into the intellectual property scheme. That is, software is too much of a writing to fit comfortably into the patent system, and too much of a machine to fit comfortably in the copyright system. The look and feel controversy is just one manifestation of this difficult accommodation process.

Because computer software is a set of instructions for executing operations of a computer—and indeed is simply an alternative implementation to hardwiring a sequence of functions—it would seem natural subject matter for patent protection. Patent is the body of intellectual property law that traditionally has protected machines, processes, and other technological innovations. To get a patent, an inventor must file an application with the Patent and Trademark Office specifying exactly what his or her invention is and must persuade the patent examiner that the innovation is really inventive and not found in the state of the prior art. (By contrast, copyright protection attaches automatically from the time an author fixes a work of authorship in some tangible medium, and has a very low-level originality standard, i.e., owing its origin to its author.) If the patent examiner is satisfied that the patent standards have been met, a patent will issue and will give the patent owner 17 years of exclusive protection of the invention. (Again, by contrast, copyright protection lasts for the life of the author plus 50 years, and does not require a rigorous examination process like patent law does.)

Software is too much of a writing to fit comfortably into the patent system, and too much of a machine to fit comfortably in the copyright system.

Although patent law is the natural body of intellectual property law to look to protect software inventions, the fact is that patent law got off to a very slow start as a way of protecting software innovations.

For many years, particularly in the 1960s and 1970s, the Patent Office and the courts were very hostile to software patents, and it looked for a while like patent law would play only a very limited role in protecting computer software. Several early Supreme Court decisions cast doubt not only on the patentability of algorithms, but of software as well. The Commission that in 1978 urged Congress to add computer software to the copyright system did so in part because of the perceived absence of meaningful patent protection for software.

Things began to turn around for software patents, however, after a broad and favorable software patent decision by the Supreme Court in 1981. At the moment, the Patent Office is very receptive to software patent applications, and many software patents are being issued, including patents for software user interfaces. Several patent lawyers have recently written articles saying that the "look and feel" of software user interfaces is patentable.

It is worth starting the discussion of the history of the copyright side of intellectual property protection for software by pointing out that computer software is the first technology ever to have been admitted to the copyright realm. Traditionally, copyright law has protected only writings, not machines or other technologies, which copyright law calls useful articles. Over time the term *writings* has been construed quite broadly, both in judicial decisions and in statutory provisions, to include not only books, but also photographs, movies, sound recordings, and other works whose sole function was to convey information or display an appearance, but it has never protected a technology before.

Copyright defines as "useful," and thus as excluded from protection, those works that have a function beyond just conveying information and displaying an appearance. It has been patent law that has protected machines, technologies and other useful works. Only if some artistic feature of a useful article could exist separately from the useful part of the thing and stand alone as a work of art could copyright protect any part of the useful article. For example, a sculpture of a Balinese dancer can be copyrighted even if its creator intends to reproduce the sculpture as a lamp base.

Copyright law has traditionally treated the functional aspects of copyrightable works as the ideas in the work which copyright law would not protect. For instance, a drawing of a machine might be copyrighted as a drawing, but the copyright would only protect the drawing as a drawing. The design of the machine depicted in the drawing would be the idea that would be excluded from copyright protection because of its potential patentability.

The reason that computer hardware clones have been legal and software clones may not be is that copyright law does not protect the hardware of a computer from being copied, whereas copyright does protect software. Unless computer hardware components are patented or protected by the Semiconductor Chip Protection Act, they can be freely copied by competitors. Even where some hardware components are patented, it may be possible to buy the component in the open market and incorporate it into a subsequent product or to design around that one component to make it compatible without being exactly alike. Copyright law has a much broader reach and a more hostile attitude toward incorporation of a copyrighted work into other works.

When Congress passed the law admitting computer software to the copyright system in 1980, it did not understand that it was extending protection to a technology. Rather, Congress thought of software only as a literary work, because it was most often created by writing source code which then would be converted to object code. (Copyright

law defines *literary work* to include a wide range of written works, not just things like Ernest Hemingway's *A Farewell to Arms*.)

Because Congress did not conceive of software as a technology, Congress did not anticipate how difficult the functionality of software would make it to accommodate software smoothly into the system of copyright. After all, it is a contradiction to put a technology (such as software) into a body of law (such as copyright) which has as a fundamental tenet the nonprotection of technologies. Accommodation of software into the copyright system has been made even more difficult than the contradiction alone might have predicted because Congress and the judges in most of the software copyright cases thus far have continued to ignore the technological dimensions of software. Among the consequences of this unrecognized contradiction is that there is considerable uncertainty in the law about the proper scope of copyright protection for software user interfaces and about the proper relationship between copyright and patent protection for software, which is why it is not yet clear whether the look and feel of software user interfaces should be protected by copyright or patent law.

art would be eligible for intellectual property protection. More obvious and incremental improvements would be freely available for all to use. Even the most significant advances would also become freely available once the seventeen-year patent expired.

COPYRIGHT ATTITUDES TOWARD INCREMENTAL IMPROVEMENTS AND STANDARDIZATION

In general, copyright law assumes that the public is better off if the law encourages (or maybe even forces) subsequent writers to express themselves differently than their predecessors. The more different, the better. This is one reason why the copyright law gives copyright owners the right not only to prevent others from reproducing their works, but also to control the making of derivative works. It is not a defense to a claim of infringement based on the derivative work right that one has improved on or built upon a preexisting copyrighted work. The law assumes that since there are usually a great many ways to express the same idea, a second author won't be impeded in creating a valuable new work by having to make up something new, rather than building upon the work of predecessors. The principles and doctrines of copyright law are set up to promote diffuse expression.

Patent law may be better equipped than copyright law to understand the need for incremental improvements in user interface designs.

Patent law, in contrast, has recognized that technology tends to grow in a more incremental fashion, which is why modest improvements in technology are not protectable at all, why inventive improvements are protected for a significantly shorter time than the copyright duration, and why patent law gives patent owners no right to control derivative inventions. Indeed, one who invents an improvement on a patented machine can separately patent his or her improvement without the patentee's permission. (The improver may still need the original patent owner's permission to make the underlying invention, but the improver will own the improvement patent and can stop the patent owner from making the improvement.) Patent law may, therefore, be better equipped than copyright law to understand the need for incremental improvements in user interface designs.

Because copyright law has no experience dealing with a technology, it also has no experience understanding that sometimes standardization may be necessary for widespread public use of a technology and/or as a base for further innovation. The QWERTY keyboard is a good example of the need for standardization. The QWERTY keyboard (which was once patented and was once an efficient keyboard arrangement because it prevented key gridlock) is a standard keyboard configuration, not only for typewriters, but also for computers. Even though there are now more efficient keyboard arrangements (because the technology has evolved so that key gridlock is not a problem), QWERTY is still used because of the advantages for users of a standard arrangement of keys. If one followed the recent software copyright user interface cases as applied to keyboard configurations, the fact that there are other possible arrangements of the keys would make the first manufacturer to use this arrangement the owner of it for his life plus fifty years. Thus, when the defendants in the Lotus case attempt to argue that the Lotus interface (especially its command set) has become a standard in the spreadsheet market, there is no direct precedent in copyright law with which to support the argument. And once again, the look and feel test seems to exclude consideration of the standardization issue.

CONCLUSION

Because of the functionality of user interfaces, because the sequence of screen displays generated by software is generally reflective of a sequence of functions being performed by the software, and because machine–human interaction devices have traditionally been patentable, it is more consistent with legal tradition to protect most aspects of software user interfaces through patent law than through copyright law.

Even if patent law took over much of the role in protecting software user interfaces, copyright could still be available to protect purely graphical or explanatory material displayed on a monitor screen, but it should continue to consider potentially patentable features of user interfaces as unprotectable ideas under copyright law. It would undercut the public policy of patent law to allow a modest user interface innovation to get through copyright law a much longer period of protection than would be available if the innovation was inventive enough to be patentable. Patent policy has traditionally left in the public domain and available for free copying those more modest innovations that are not inventive. By passing the law that allowed software to be protected by copyright, Congress did not intend to undermine this longstanding policy of the patent law or to change the historic relationship of the patent and copyright laws.

If the judges in the Lotus and Apple cases were to address the issue of look and feel protection in a way that took into account the availability of patent protection for software user interfaces, the functionality of many of the features of the Lotus and Apple interfaces, and the traditionally narrow role for copyright in protecting more functional kinds of works, the judges would not find "look and feel" to be an appropriate copyright standard and would filter out the functional features of these interfaces that should be protected by patents if at all.

It is possible that upon this closer and more analytic examination of the user interface similarities, the judges might still find sufficient similarities to find copyright infringement, but it would be a major accomplishment if the courts used these two cases to bring the kind of clear and definitive analysis of the scope of copyright protection for interfaces that the software industry needs to have to guide it, at least in general terms, about what aspects of interfaces are protectable and what are not. The present uncertainty in the law hurts the industry and freezes up development efforts until the legal issues are definitively resolved.

Acknowledgment. I want to thank Elaine Rich, Jonathon Gruden, and Robert Glushko for their helpful comments on this article.

CR Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Tools and Techniques—*user interfaces*; K.5.1 [**Computing Milieux**]: Software Protection—*copyrights*

General Terms: Legal Aspects

Additional Key Words and Phrases: Court decisions, intellectual property protection, look and feel, total concept and feel

ABOUT THE AUTHOR:

PAMELA SAMUELSON is a professor of law at the University of Pittsburgh School of Law where she teaches intellectual-property law. In 1985 and 1986, she was the principal investigator of the Software Engineering Institute's Software Licensing Project which recommended substantial changes in the U.S. Defense Department's Software acquisitioin policy. Author's Present Address: Emory Law School, Gambrell Hall, Atlanta, GA 30322.