An Analysis of the Copyrightability of the "Look and Feel" of a Computer Program: *Lotus v. Paperback Software*[*]

TABLE OF CONTENTS

---

[*] Winner, 1991 Nathan Burhan Memorial Competition (Ohio State University College of Law).

I. INTRODUCTION

On June 28, 1990, Judge Robert Keeton of the United States District Court of Massachusetts handed down a decision that was instantly hailed by computer industry analysts as a harbinger of doom for the future of computer software. The decision of *Lotus Development Corp. v. Paperback Software International*[1] ended three years of intense debate over whether certain elements (specifically, the "look and feel") of Lotus' spreadsheet program, Lotus 1-2-3, were copyrightable, and, if so, whether or not Paperback's spreadsheet, VP Planner, had infringed Lotus' copyright. The case pitted industry giant and software spreadsheet leader Lotus[2] against a comparative industry midget, Paperback Software,[3] and juxtaposed a minority of experts favoring strong copyright protection against the majority of experts fearing strong copyright protection.

Judge Keeton ruled in favor of Lotus, finding as a matter of law that Paperback's VP Planner infringed part of the "look and feel" of Lotus' 1-2-3.[4] Before the dust had settled, Lotus filed suit against two other spreadsheet vendors,[5] settled with Paperback on the issues of damages and appeal,[6]

---

[1] 740 F. Supp. 37 (D. Mass. 1990).

[2] "Lotus 1-2-3 was . . . the spreadsheet leader, the state of the art in business software, and was reaping handsome economic profits." Comment, *The Incompatibility of Copyright and Computer Software: An Economic Evaluation and a Proposal for a Marketplace Solution*, 978 N.C.L. REV. 977, 1004 (1988). In 1986, Lotus' market share of the spreadsheet industry may have been as high as 70-80%, with few vendors offering a serious challenge to Lotus' market dominance. *See* Radding, *Race of Power vs. Position*, COMPUTERWORLD 51 (Dec. 21, 1987). As of 1990, while Lotus is still the industry leader, Borland's Quattro Pro has gained a significant market share. *See infra* note 210.

[3] Based on dollar volume of sales in 1987, Paperback was the fourth leading spreadsheet, with $2.8 million in sales, compared to an estimated $198 million for Lotus (Microsoft was second at $38 million, and Supercalc was third at $12.5 million). In terms of units shipped, Lotus was in first place with 750,000 units shipped (62%), and Paperback third at 60,000 units shipped. Lotus marketed its spreadsheet (Lotus 1-2-3) at $495, while Paperback marketed its spreadsheet (VP Planner) at $99. *See* Radding, *supra* note 2.

[4] *Lotus*, 740 F. Supp. at 70.

[5] Two court days after *Lotus v. Paperback* was decided, Lotus filed suit against Borland International, vendor of Quattro Pro (Lotus Dev. Corp. v. Borland Int'l, Inc., No. 90-11662-K (D. Mass., filed July 2, 1990)), and against The Santa Cruz Operation, vendors of SCO Professional (Lotus Dev. Corp. v. The Santa Cruz Operation, No. 90-11663-K (D. Mass., filed July 2, 1990)). Lotus and SCO have since settled out-of-court, with SCO agreeing to stop manufacturing, distributing, and licensing SCO Professional. *See generally* Picarille, *Lotus Settles with SCO, but continues Battle with Borland*, INFOWORLD 152 (June 24, 1991). As of August, 1991, the Lotus v. Borland case remains in litigation.

continued litigation with previously-joined litigant Mosaic Software,[7] and sat back to watch the impact of "one of the most closely watched cases in software protection annals."[8]

This Paper will attempt to analyze the *Lotus* holding in terms of present-day copyright law, discussing the merits of the holding, the prospective impact on the software industry, and the prospects of precedential value. Part II of this Paper discusses the background necessary to understand the *Lotus* decision, including copyright law basics and an attempt to understand what is actually meant by "look and feel." Part III discusses the judicial and legislative history of computer copyright litigation leading up to *Lotus*. Part IV discusses the *Lotus* holding itself, including a discussion of some potential problem areas in the decision. Part V discusses an informal survey of the response of, and impacts on, the computer industry. Part VI offers a proposed method of viewing the decision.

II. A BRIEF DISCUSSION OF COPYRIGHT LAW, COMPUTERS, AND THE "LOOK AND FEEL" OF A COMPUTER PROGRAM

A. *A Succinct Statement of the Issue*

Granting that copyright protection has been held to cover literal manifestations of a program such as code and the structure of code,[9] should copyright protection be extended to cover nonliteral elements of a computer spreadsheet program, namely the "look and feel" of Lotus 1-2-3?

---

[6] The parties settled for damages of $500,000, the withdrawal of VP Planner from the market, and a promise from Paperback to refrain from appeal. For an interesting discussion of the reason behind the settlement, *see* Ould, *Legal Dispute Kept Paperback from Lotus Appeal*, 8 PC WEEK 138 (Jan. 21, 1991) (Paperback insurers forced settlement for financial reasons when Paperback wanted to appeal).

[7] Lotus Dev. Corp. v. Mosaic Software, Civ. No. 87-74-K (D. Mass., filed March 6, 1987). The *Mosaic* case had originally been joined with the *Paperback* case, but was split. Mosaic, vendor of the Lotus 1-2-3 clone Twin, argued its case on the week of Nov. 20, 1990, and is expecting Judge Keeton's ruling sometime in 1991.

[8] Greguras, *Implications of Lotus 1-2-3 Copyright Infringement Decision*, 9 SOFTWARE PROTECTION 1, 5 (July-Aug. 1990). Even before the decision in *Lotus*, one commentator noted that "[t]he widespread press coverage given to computer 'look and feel' litigation illustrates the tremendous commercial and legal interest in the potential expansion of copyright law." Yen, *A First Amendment Perspective on the Idea/Expression Dichotomy and Copyright in a Work's "Total Concept and Feel,"* 38 EMORY L.J. 393, 419 n.152 (1989).

[9] *See infra* note 33.

If such protection is extended, would a program, such as Paperback's VP Planner, be considered an infringer if it contains significant "cloning" of Lotus 1-2-3's "look and feel," specifically, direct copying of the structure, sequence, and organization of the menu command system?

B. *Several Basics of Copyright Law*

Copyright law begins with the Constitution of the United States. The Constitution grants power to Congress "To promote the Progress of Science . . . by securing for limited Time to Authors . . . the exclusive Right to their . . . Writings."[10]

In exercising such power, Congress has enacted several copyright acts since the inception of the Constitution. The latest of the acts, the Copyright Act of 1976,[11] occupies the field of copyright, abrogating overlapping state law copyright doctrine and previous Copyright Act doctrines for works authored after January 1, 1978. It is this Act with which this Paper is concerned.

Section 102 of the 1976 Act states the basic requirements for copyright protection: "Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression . . . ."[12] Thus, to be protectable, a creation must pass three basic tests: 1) it must be original; 2) it must be a work of authorship; and 3) it must be fixed in a tangible medium of expression.

To be "original," a claimant must pass one of two tests: if a claimant originated a work, the work must be "one man's alone" (i.e., there is no minimal level of artistic merit required);[13] or if the claimant produced a copy of a work, the copy must show a "modicum of creative work,"[14] and prove that the variation between the original and the copy is more than trivial.[15]

To help explain what "works of authorship" are, Congress created a nonexclusive list of works in the 1976 Act which includes: "(1) literary

---

[10] U.S. CONST., art. I, § 8, cl. 8.

[11] *Codified at* 17 U.S.C. § 101-914 (1989).

[12] 17 U.S.C. § 102(a) (1989).

[13] Bleistein v. Donaldson Lithographing Co., 188 U.S. 239, 250 (1903).

[14] Amsterdam v. Triangle Publications, Inc., 189 F.2d 104, 106 (3d Cir. 1951).

[15] Alfred Bell & Co. v. Catalda Fine Arts, Inc., 191 F.2d 99 (2d Cir. 1951).

works; . . . [and] (6) motion pictures and other audiovisual works . . . ."[16] While a computer program is not explicitly included in this list, computers "fall squarely"[17] within the statutory definition of "literary works."[18]

The third test, fixation in a tangible medium, is a throwback to the constitutional requirement of a "writing." To satisfy this requirement, a work must be "now known or later developed, from which [it] can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device."[19]

In addition to the three-pronged test, several other limitations to copyrightability are present: specifically, copyright protection does not extend to any "idea, procedure, process, system, method of operation, concept, principle, or discovery."[20] These limitations are a statutory codification of several judge-made laws that are pertinent to all prospective copyright protection: specifically, the idea/expression dichotomy, the limitation against functional (utilitarian) items, and the literal/nonliteral distinction. Each of these concepts will be discussed briefly.

The most basic of these concepts is the idea/expression dichotomy.[21] Briefly stated, copyright protection extends only to the expression of ideas, but not the ideas themselves. Drawing the line between ideas and expression is not easy, however, and is a central issue in the *Lotus* case. The concept of merger is tangential to the idea/expression dichotomy; specifically, if there are only a small number of ways to express an idea,[22] or if the expression is inevitably bound up in the idea,[23] then the idea and the expression are said to merge, and the expression is not copyrightable. The theory behind the idea/expression dichotomy and merger helps to aid in understanding; namely, if an idea is copyrightable, a monopoly may be created that would occupy an entire area of thought, preventing others from being able to express such

---

[16] "Works of authorship include the following categories: (1) literary works; (2) musical works, including any accompanying words; (3) dramatic works, including any accompanying music; (4) pantomimes and choreographic works; (5) pictorial, graphic, and sculptural works; (6) motion pictures and other audiovisual works; and (7) sound recordings." 17 U.S.C. § 102(a) (1989).

[17] Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 49 (D. Mass. 1990).

[18] "'Literary works' are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied." 17 U.S.C. § 101 (1989).

[19] 17 U.S.C. § 102(a) (1989).

[20] 17 U.S.C. § 102(b) (1989).

[21] *See* Baker v. Selden, 101 U.S. 99 (1880).

[22] *See* Morrissey v. Procter & Gamble Co., 379 F.2d 675 (1st Cir. 1967).

[23] *See* Continental Casualty Co. v. Beardsley, 253 F.2d 702 (2d Cir. 1958).

ideas. To avoid this type of protection, copyright does not extend to an idea or to an expression whose idea has merged with that expression.

A second limiting concept is the denial of copyright protection to functional, utilitarian, or useful articles.[24] The protection of any such article is governed by patent law. Thus, if an article has any use, specifically as a process, system, or operation, copyright does not protect such functional elements of the article. Copyright protection does, however, cover any nonfunctional aspects of such articles, as long as such aspects pass the three-pronged statutory test and are expressions, not ideas. Just because an aspect of an article is functional does not deny copyright protection to nonfunctional elements. Tangentially, if a work has both functional and nonfunctional purposes, only the nonfunctional aspects of the work are protectable.[25] The issue is said to be one of "separability," namely, whether or not the functional and nonfunctional aspects can be separated from each other. If such aspects are not separable, copyright protection can not be afforded.

A third concept of copyright law is the distinction between literal and nonliteral elements. A literal element is one that is a work of literature,[26] an element written in some form. Such literal elements are protectable by copyright. Additionally, however, it is well-settled law that some nonliteral elements of expression are also subject to copyright protection.[27]

Ever present in all determinations of protection is the policy reason behind copyright protection. "The immediate effect of our copyright law is to secure a fair return for an 'author's' creative labor. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good."[28] Congress thus grants limited monopolies to authors in order to serve the public welfare.[29] As one commentator suggests, noting that the

---

[24] Useful articles are defined as "article[s] having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information. An article that is normally a part of a useful article is considered a 'useful article.'" 17 U.S.C. § 101 (1989).

[25] "The design of a useful article . . . shall be considered a . . . [copyrightable] work . . . if, and only to the extent that, such design . . . can be identified separately from, and [is] capable of existing independently of, the utilitarian aspects of the article." 17 U.S.C. § 101 (1989).

[26] Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 51 (D. Mass. 1990).

[27] Examples of this would be the expression of the setting, characters, or plot in a play. *See, e.g.*, Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930).

[28] Sony Corp. v. Universal City Studios, Inc., 464 U.S. 417, 432 (1984).

[29] *Lotus*, 740 F. Supp. at 52.

author's interests are almost subservient to society's, the arrangement is actually "quid pro quo."[30]

If copyright protection is established, the next element in a copyright infringement claim is proof of the infringement. This is carried out by determining whether the alleged infringing work is "substantially similar" to the original work.

C. *"Look and Feel" Defined*

The phrase "look and feel of a program" is a term of art for an elusive concept. Many people have used the phrase in many different ways, a complication that may, more than any other factor, have lead to the great confusion in the software industry today.[31] This Paper will initially attempt to provide a working definition of "look and feel" in an effort to explain the basic concept. The several definitions of courts and users will be explored, followed by a discussion of the place and importance of "look and feel" in the software industry.

1. *Towards a Working Definition of "Look and Feel"*

In general, the "look and feel" of a computer program can be defined as the elements of a program that a user of the program will deal with upon using the program. This is indeed a very broad definition and it is first important, and in fact easier, to discuss what the phrase "look and feel" does *not* encompass.

The "look and feel" of a program generally refers to software elements of a computer, not hardware.[32] In terms of software, "look and feel" also does not describe any element of a program that a user can or will not ever see directly; that is, "look and feel" does not encompass any elements of the

---

[30] U.S. Congress, Office of Technology Assessment, Communication and Information Technologies Program, *Staff Paper on Intellectual Property Protection for Computer Software, reprinted in* SOFTWARE PROTECTION 12, 14 (March 1990) [hereinafter "*Staff Paper*"].

[31] *See infra* notes 176-84 and accompanying text.

[32] This is, perhaps, an overstatement. Because, for a user to be able to use software, hardware is necessary, the particular type of hardware used can and will change the way a user views and uses a program (examples would be certain programs that will run only with a certain type of processor, or certain programs that will run only with a certain type of video screen [i.e., color versus monochrome]). Yet these changes deal only with the limitations of hardware to run a particular program; here, "look and feel" encompasses certain elements of software of which the basics will remain constant, regardless of what hardware is used.

source code, object code,[33] or any structures or organizations thereof that a software developer will program and the computer will read and run. Likewise, microcode, disk operating systems,[34] and any other program that runs "behind the screens," (i.e., that the user will not see or deal with while using a program) are not covered by "look and feel." Additionally, any user support, such as textual documents and training, are not included in "look and feel."

What is left after such omissions are any elements of the program that a user can perceive (see, hear, etc.) and use to interact with the computer. This includes, but is not limited to, the way commands are placed on the screen and accessed, the way screens interact with each other, the process a user will go through to invoke a specific function, the organization and interaction

---

[33] All programs are sequences of instructions called, collectively, code. There are several different types of code. Microcode are commands physically placed inside the computer (untouchable by the user and actually encoded into hardware) that control certain basic internal actions of the computer (such as controlling communications between the CPU and the main memory). Source code is a phrase used to describe a set of commands as written by a person. This code, in and of itself, is not legible to a computer. To perform these commands (of the source code), a computer must translate human-written source code into object code, which is a series of instructions that the computer itself can understand.

Source code is thus the method of communication that a programmer (a person who writes commands for a computer to follow) uses to instruct a computer. There are several different types of commands, or languages, that programmers use to communicate with the computer. One language, machine code, is the language that computers speak; machine code is thus the same as object code. Few programmers use this language any more. Other languages offer more conveniences to programmers by making source code more like English, then providing methods to convert the source code to machine-readable object code. Examples of such languages are Assembly language (the crudest), "high-level" languages such as PASCAL, COBOL, and BASIC (languages close to English), and "next-generation" languages such as C or ADA (which attempt to allow programmers to program roughly in ideas instead of instructions).

[34] In general, there are two types of software, operating programs and application programs. Operating programs, such as DOS and OS/2, control the internal functioning of the computer, while application programs are programs such as wordprocessors, spreadsheets, etc., which attempt to command the computer to perform functions towards a particular applied end.

of a program's functions, and, maybe, any macro code[35] facility based upon such program elements.[36]

A good rule of thumb is that, if you are a user, "look and feel" is whatever you see, and whatever steps you have to take to make the program do what you want it to do.

2. *Differing Views and the Growing Importance of "Look and Feel"*

As the United States Congress, Office of Technology Assessment has recognized, "[t]he legal and technical communities do not have consistent definitions for important terms like . . . 'interface.' Without agreement on a common language, discussing protection issues is extremely difficult."[37] Neither the legal commentators, the computer industry commentators, nor the courts have come up with a consistent definition of "look and feel." Concepts such as "user interface," "total concept and feel," and "structure, sequence and organization" have been used interchangeably. The overall concept is the same, but the details differ greatly. As I will contend, it is this difference in understanding that has led to a great deal of the misinterpretation of the *Lotus* decision. The following is a survey of differing definitions.

The legal commentators are perhaps the most divided. They can be separated into two general groups: those who believe that the terms "user interface" and "look and feel" are interchangeable (and thus encompass almost all, if not all, nonliteral elements of a program), and those who differentiate the terms, believing that different legal standards apply to each.

---

[35] Macro code is a type of program that a user creates to perform certain sequences of commands with simpler commands. This code must be seen as separate from any of the codes previously defined. As a user interacts with a program, the user is said to give the program commands as to which functions to perform. Many of these command sequences are boring, mundane, and do not change dependant upon application. Users may wish for some way of storing a sequence of user commands, so that the commands could be executed once, remembered by the computer, and then be executed again by a shorter and simpler sequence of commands.

[36] This element, as cautiously recognized by the *Lotus* court, will be discussed in more detail. *See infra* notes 147-57 and accompanying text.

[37] *Staff Paper*, *supra* note 30, at 13.

In the minority[38] are the commentators who do not differentiate between the terms.[39] One commentator, for example, defined both "user interface" and "look and feel" as "[t]he visual and tactile 'aura' created by the particular layout of displays and input formats."[40]

The majority of commentators differentiate between "user interface" and "look and feel," with the largest subset defining "user interface" as one element of "look and feel."[41] Those commentators that define "look and feel" without defining "user interface" find that "look and feel" includes all computer/user interactions and other elements, such as keyboard interfaces, which seem to impliedly include "user interface" within "look and feel."[42] The commentators who just define "user interface" seem to include many forms of interactions in their definitions, such as screen displays, keyboard

---

[38] *See generally* Curtis, *Engineering Computer "Look and Feel": User Interface Technology and Human Factors Engineering*, 30 JURIMETRICS J. 51 (1989).

[39] *See, e.g.*, Yen, *supra* note 8, at 419 n.152 ("Briefly stated, the 'look and feel' of a computer program consists of the design and presentation of the software's user interface.").

[40] *Id.*

[41] *See*, for example, Lundberg, Michel, and Sumner, who state:

> In general, "look and feel" can be defined as a set of functional capabilities of a programmed computer and the way it "interacts" with a user. It includes "user interface" features and elements such as menus, icons and status fields, function key assignments and commands, the set of functions performed by a program computer, as well as "artistic" features as might be found on a screen display.

Lundberg, Michel, & Summer, *The Copyright/Patent Interface: Why Utilitarian "Look and Feel" Is Uncopyrightable Subject Matter*, 6 COMPUTER LAW. 5, 5. *See also* Note, *Single Copyright Registration for Computer Programs: Outdated Perceptions Byte the Dust*, 54 BROOKLYN L. REV. 965, 976 n.55 (1988) (quoting Russo & Derwin, *Copyright in the "Look and Feel" of Computer Software*, 2 COMPUTER LAW. 1 (Feb. 1985)) ("Such factors as design and presentation, including the screen displays and general user interface through which the user interacts with the computer, make up the 'look and feel' of a program.").

[42] *See, e.g.*, Bhojwani, *Copyright Laws and the Nature of Computer Software*, 9 SOFTWARE PROTECTION 1, 2 (June 1990) ("Taken together, the idea and the look and feel constitute the architecture of the software, much as they would for a building. This includes, for example, the style of interaction with the software using the keyboard and the mouse, the actual layout of the screen images, etc."); Conley, *Look and Feel: In Defense of the Current Case Law*, 5 COMPUTER LAW. 1, 2 (Dec. 1988) ("[T]he windows, pull-down menus, and more generally, the general ease of use of the Apple Macintosh computer comprise its look and feel;" and "[t]he look of a program can be distinguishable from its feel: the look consists of the audiovisual features that the user sees and/or reads, whereas the feel consists of the overall impression of user-friendliness that the program conveys."); *Id.*, (quoting Siegel and Derwin, *Copyright Infringement of the 'Look and Feel' of an Operating System by Its own Applications Programs*, 4 COMPUTER LAW. 1, 2-3 (Jan. 1987)) ("The look and feel includes 'both *audiovisual* components . . . and *functional* components.'").

interactions, and menus.[43] Other commentators explicitly reject the uses of other phrases.[44]

Computer industry analysts seem similarly confused.[45] Some follow the wording of the *Lotus* case,[46] while others confuse the issue with different wordings.[47]

There is no doubt that the confusion among the commentators derives from the confusion of the courts. The earliest "look and feel" cases did not even attempt to define "look and feel" or "user interface," instead discussing the copyrightability of menu screens, sequence of screens, structure and organization of the user interface,[48] or screen displays and status screens.[49]

---

[43] *See, e.g.*, Note, *supra* note 41, at 973 n.40 ("The user interface includes the keyboard . . ., the command sequences, . . . and the screen displays.");

> User interfaces include all of the devices by which the human user can interact with the computer in order to accomplish the tasks the computer is programmed to perform. Screen displays of images and text are often important components of the user interface, but user interfaces may also include hardware extensions (such as keyboards, "mice," joysticks, or switches) and sounds . . . .

Computer Software and Copyright Protection: The "Structure, Sequence and Organization" and "Look and Feel" Questions, SOFTWARE PROTECTION 7, 13 (July 1989); Bing, *Look and Feel the Norwegian Way*, 8 SOFTWARE PROTECTION 1, 1 (Dec. 1989) ("[T]he choices displayed to the user as menus, and the signals of the user by the keyboard . . . is the user interface.").

[44] *See, e.g., LaST Frontier Conference Report on Copyright Protection of Computer Software*, 30 JURIMETRICS J. 15, 27 (1989) ("[T]he use of terms such as 'structure, sequence, and organization,' 'look and feel,' or 'total concept and feel' obscures rather than assists in the application of copyright principles to software interfaces."); Samuelson, *Reflections on the State of American Software Copyright Law and the Perils of Teaching It*, 13 COLUM. J.L. & ARTS 61, 69 (1988) ("One can understand why Lotus and Apple might want to substitute 'look and feel' for 'total concept and feel' as a focus of their user interface litigations, in view of the fact that the copyright statute specifically states that 'concepts' . . . are not protectable by copyright.").

[45] *See infra* notes 176-84 and accompanying text.

[46] *See* Pane, *The only Sure Conclusion of Lotus' Victory is that Lawyers will be Partners in Development*, INFOWORLD 43, 44 (July 23, 1990) (quoting Judge Keeton's definition of the Lotus user interface as being the "menus and keystrokes."). It is unclear as to whether the menus and keystrokes are considered as part of the interface or the entire interface.

[47] *See* Schachter, *Software Protection in the Throes of a Legal Morass*, 33 DATAMATION 49 (June 1, 1987) (interpreting the case of Whelan Assocs. Inc. v. Jaslow Dental Laboratory Inc., 797 F.2d 1222 (3d Cir. 1986), as holding "structure, sequence, and organization . . . has been termed a program's 'look and feel' . . .").

[48] Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127, 1132 (N.D. Cal. 1986). *See also* discussion *infra* note 88.

Later cases attempting to draw inferences from these cases also failed to define their terms, discussing instead the external sequencing and flow of screen displays and the internal method of navigation between screens.[50]

Other courts have attempted to define their terms. One court stated: "The user interface, also called the 'look and feel' of the program, is generally the design of the video screen and the manner in which information is presented to the user."[51] Another court borrowed a definition from the seminal case of *Whelan Assoc., Inc. v. Jaslow Dental Laboratory, Inc.*[52] and defined user interface as the "overall structure and organization of a computer program, including its audiovisual displays, or screen 'look and feel.'"[53] Judge Keeton in *Lotus* does not attempt to define either term himself, instead relying on plaintiff Lotus' definition. Originally claiming that the phrase "look and feel" described Lotus' copyrightable elements, Lotus amended their arguments, claiming subsequently that Lotus 1-2-3's copyrightable elements are its "user interface," with "user interface" defined as "such elements as 'the menus (and their structure and organization), the long prompts, the screens on which they appear, the function key assignments, [and] the macro commands and language.'"[54]

There is obviously a great deal of confusion as to exactly what "look and feel" and "user interface" mean, and, more dangerously, great confusion over whether a court's holding actually was dependant upon these meanings. The above survey of definitions was not an attempt to derive a general definition or determine which definition was "best."[55] The intention was to show the confusion with the issue, and hint at the danger of such confusion. It seems very easy for a case to hold, for example, that a menu structure is part of the user interface, which is part of the "look and feel" of the program. Only the menu structure could be held as copyrightable, and subsequently an infringement. Yet, because of the confusion of terms, a

---

[49] Digital Communications Assoc., Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449, 455 (N.D. Ga. 1987). *See also* discussion *infra* note 94.

[50] Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 994-95 (D. Conn. 1989).

[51] Johnson Controls, Inc. v. Phoenix Control Sys., 886 F.2d 1173, 1175 n.3 (9th Cir. 1989).

[52] 797 F.2d 1222 (3d Cir. 1986).

[53] Telemarketing Resources v. Symantec Corp., 12 U.S.P.Q.2d (BNA) 1991, 1993 (N.D. Cal. 1989).

[54] Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 63 (D. Mass. 1990).

[55] But if one were to be picked as the most accurate, the courts that refused to define their terms, while holding that very detailed elements of a program were infringing (i.e., it is the menu structure that is copyrightable and was infringed) probably cause the least confusion as to what they hold. The precedential value of such cases, however, may not be so simple.

commentator could easily suggest that the court held that the "user interface" was copyrightable (i.e., the whole interface). Not only would this convey an incorrect idea, but another commentator may state that the case stands for the prospect that the "look and feel" of a program is now copyrightable. This is also an incorrect overstatement of the holding, but a definitionally "correct" one depending upon which definition above is used. Such misconceptions could easily permeate an industry, spreading an incorrect and overbroad interpretation of a case's holding.

To allow a broad generalization: it seems that the term "look and feel" is a superset of the term "user interface." While Judge Keeton at times uses the phrases seemingly interchangeably, this Paper will continue to use the most expansive phrase ("look and feel") to discuss the copyrightability of nonliteral elements of programs. Any other, more restrictive uses (such as "user interface") will be explicitly stated.

Regardless of how it is defined, the "look and feel" of a computer program has become a critical element of all software. Over the last decade or two, as hardware costs have decreased and hardware speeds have increased,[56] the role of user-friendly software has grown in importance, to the point presently when a program's "look and feel" could very well be its most important and marketable factor.[57]

---

[56] A long time ago in computer years, namely over ten years ago, the concept of the importance of the "look and feel" of a program was in its infancy. Hardware was much slower and more expensive than it is now. When making a determination of which product to use, a user or a business placed the program's interface far down on the list of important items, for the cost difference in the speed and power of a machine far outweighed the cost difference in training people to use the machine and program. In short, the people cost was much less than the computer cost. Then, several events in the computer industry changed all of the above.

First, hardware continued an increasingly fast spiral towards decreasing price and increasing speed and power. The machines themselves could do more for less, and eventually the cost of training people and using people began to exceed the cost of using computer time. Secondly and concurrently, the software industry saw an explosion of software vendors, the competition of which heightened the quality of software. Thirdly, and perhaps most importantly, the introduction of the IBM PC started a course of standardization in the computer industry that allowed many companies to compete for many business processes. By 1988, "U.S. independent software developers' revenues exceeded $25 billion, up from $20 billion in 1987." *Staff Paper*, *supra* note 30, at 12, 14.

[57] The sum total of the occurrences in note 56 lead to the present-day importance of the "look and feel" of a program. *See generally* Curtis, *supra* note 38, at 52 ("In contrast to its neglect from the 1940s through the 1970s, during the last decade the user interface has become a paramount concern in designing computer systems."); Note, *Copyright Protection for Computer Screen Displays*, 72 MINN. L. REV. 1123, 1138 (1988) (paraphrasing B. SHNEIDERMAN, DESIGNING THE USER INTERFACE: STRATEGIES FOR EFFECTIVE HUMAN-COMPUTER INTERACTION 3, 8-9 (1987)) ("Designers invest much creative energy toward

As such, software vendors began to change the way software was developed.[58] Instead of spending the majority of their time programming, vendors now spend the majority of time and money studying and designing the "look and feel" of a program;[59] this is the element by which purchasers are most likely to make purchasing decisions.[60] With such a significant investment in the "look and feel" of a program came a wish to protect the

---

making the user interface of even the most complex programs simple and unobtrusive."); Bendekgey, *Copyright Protection for Computer Software Visual Displays: Protecting a Program's Look and Feel*, 11 SOFTWARE PROTECTION 1, 1 (Jan. 1988) ("The development of the mass personal and business microcomputer markets has created a corresponding demand for computer software that can be used with relative ease by people having little familiarity with computers.").

With so many applications running on each type of computer, a user or business was no longer forced to stay with difficult-to-use programs that conserved computer resources but wasted people resources. *Id.* Users instead shopped for programs that served a needed purpose and were easy to use. While concerns such as how a program was written or organized faded, concerns as to how the program augmented human use began to rule. *See* Curtis, *supra* note 38, at 52.

[58] Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1052 (1989) ("[T]he process by which user tasks are defined and interfaces designed has developed into a hybrid science drawing on a wide range of disciplines."); *See also* Curtis, *supra* note 38, at 52-53, who states:

> The design and engineering of the user interface is now emphasized during computer systems development for five reasons. First, advances in computer science and electrical engineering have placed better technology at the disposal of those designing and building user interfaces . . . . Second, with better interfaces for nonprogrammers, the user interface has become a crucial issue in marketing computer systems . . . . Third, the user interface typically accounts for at least *40 percent*, and sometimes much more, of the computer instructions written in building a commercial application program . . . . Fourth, the user interface is a growing concern . . . because standards are being built around them . . . [and] [f]inally, the cost of learning a software package is substantial.

(Emphasis added, citations omitted).

[59] *See supra* note 56; Note, *supra* note 57, at 1138 ("Designers invest much creative energy toward making the user interface . . . ."); Ranney, *'Look and Feel' Discussed as Major Copyright Issue*, INFOWORLD 13 (Nov. 11, 1985) ("The design and presentation or, in short, the 'look and feel' of a computer software product often involves much more creativity and often is of much greater commercial value than the program code which implements the product.").

[60] *See* A. CLAPES, SOFTWARE, COPYRIGHT, & COMPETITION: THE "LOOK AND FEEL" OF THE LAW 202-03 (1989).

investment.[61] That wish propels all advocates of "look and feel" copyrightability today.

III. THE HISTORY OF SOFTWARE LITIGATION BEFORE *LOTUS*

A. *CONTU and the Initial Literal/Nonliteral Debate*

In 1980, the congressionally created National Commission on New Technological Uses of Copyrighted Works (CONTU) produced several recommendations and a lengthy legislative history pertaining to the copyrightability of computer software. While perhaps out of date at publication,[62] Congress recognized CONTU's observation of a need for protection of software, and amended sections 101 and 117 of the Copyright Act of 1976. To Section 101 was added a definition of "computer program,"[63] and section 117 was amended to allow certain legal copying of computer programs.[64] While the final report of CONTU did not make any recommendations as to the copyrightability of the "look and feel" of a program (arguably an even fuzzier concept then than now), the legislative history makes clear that the issue was debated and that both sides of the issue

---

[61] *Staff Paper*, *supra* note 30, at 13 ("[S]oftware developers have modified their ideas about what aspects of software need the most protection. For example, as writing and checking code becomes more automated (computer-aided software development), software producers want to protect the logic and idea of a program, rather than just the effort required to code and debug it."); Bendekgey, *supra* note 57, at 2 ("[R]ecent developments in the mass market for computer programs have provided one group of companies with a strong incentive to protect their rights in certain aspects of their technology.") *See also* Russo & Derwin, *Copyright in the "Look and Feel" of Computer Software*, 2 COMPUTER LAW. 1, 1 (Feb. 1985) and Bhojwani, *supra* note 42, at 10.

[62] "When Congress created . . . [CONTU] in 1974, the 'PC revolution' had not yet begun to bring desktop computing power to millions of individuals." *Staff Paper*, *supra* note 30, at 14. As CONTU was debating, the PC industry was just in its youth. "[B]y the time CONTU issued its final report in 1978, the PC revolution was underway, creating a new generation of computer users . . . ." *Id.* There was no way for CONTU to foresee the booming new industry. In fact, "market changes . . . [have caused] almost . . . [a] hundredfold increase in PC use since CONTU." *Id.*

[63] "A 'computer program' is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101 (1989).

[64] 17 U.S.C. § 117 states, in sum, that limited archival copies of computer programs are legal if used only for archival purposes.

were fortified by experts.[65] One thing that CONTU did recognize, however, was the "inherent difficulties in applying copyright . . . to software."[66] As will be seen, this observation is a common thread that holds all later software copyright cases together; specifically, a common agreement that copyright law does not apply well to software, and that some bending of the law is necessary to apply the law to computer software.

One month before the results of CONTU were released, the Northern District of Texas decided *Synercom Technology v. University Computing*,[67] a case that was truly ahead of its time.[68] In *Synercom*, the court determined that the alleged infringement of an input format was not an infringement because the idea of the input and the expression of the input merged, and thus was not copyrightable.[69] *Synercom* became the first case to apply the idea/expression dichotomy to "look and feel."[70]

Synercom aside, the courts had a much more basic issue facing them first: whether or not the literal manifestations of a program (the source code and object code) were copyrightable. The issue came about as a natural side effect of the changing computer industry; one developer, seeing a profitable program, would simply copy the code,[71] or parts of the code, outright.

The first wave[72] of cases held that the literal manifestations of a program, namely the source code and the object code, were copyrightable.[73]

---

[65] The debate can be summarized by the dispute between Melville Nimmer and Commissioner Hersey. Nimmer felt that "the rights in computer programs would be the same as the rights in other 'literary works,'" or, more generally, that copyright protection should extend beyond the literal aspects of a program to its nonliteral aspects. Commissioner Hersey felt the opposite, stating that "programs should *not* in all respects be protected in the same manner as other copyrighted works." Clapes, Lynch, & Steinberg, *Silicon Epics and Binary Bards: Determining the Proper Scope of Copyright Protection for Computer Programs*, 34 UCLA L. REV. 1493, 1586 (1987). As will be seen *infra* notes 72-85, courts have chosen generally to follow Nimmer's views.

[66] *Staff Paper*, *supra* note 30, at 14.

[67] 462 F. Supp. 1003 (N.D. Tex. 1978).

[68] Bender, *Software Copyright: "Look and Feel" Issues*, SOFTWARE PROTECTION 1 (Nov. 1989).

[69] *Synercom*, 462 F. Supp. at 1005.

[70] *Id.* at 1009.

[71] Bendekgey, *supra* note 57, at 2.

[72] *See generally* Clapes, Lynch, & Steinberg, *supra* note 65, and Bender, *supra* note 68, for general discussions dividing the cases to be discussed into waves.

[73] Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1243 (3d Cir. 1983) (seminal case holding source and object code copyrightable).

This, in effect, is the most settled issue of law in the whole software copyright debate.[74]

As programs continued to get more complicated and lengthy, the designing and structuring of a program took more and more creative work and effort.[75] Courts were soon faced with attempts to extend copyright protection beyond the literal elements of program code, specifically into nonliteral elements of code such as the structure, sequence, and organization of the code.

The earliest case other than *Synercom* to deal with this extension was *SAS Institute v. S&H Computer Systems*.[76] *SAS* held that a program, even though written in a different language for another computer, substantially copied the "original structure" of the plaintiff's program, and was thus an infringement.[77]

Following in this vein of thought, one year later the Third Circuit decided the seminal structure and sequence case of *Whelan Associates v. Jaslow Dental Laboratory, Inc.*[78] *Whelan* extended the scope of copyright protection into the nonliteral realm of the "structure, sequence, and organization"[79] of a program. Addressing the idea/expression dichotomy, the Third Circuit applied a "pragmatic" test[80] between idea and expression, balancing the ultimate goal of copyright and other policy considerations, to hold that at least some nonliteral elements of a program were expression—in this case, the structure, sequence, and organization of the code. *Whelan* stated that there was no bright-line test available to determine what was idea and what was expression; the decision was necessarily ad hoc, as determined by Judge Hand years ago.[81]

The response to *Whelan* at first was not uniformly favorable. The Fifth Circuit one year later explicitly rejected *Whelan* in *Plains Cotton Cooperative Association v. Goodpasture Computer Service, Inc.*,[82] holding that "organizational copying" was not an infringement because the idea and expression of the program merged.[83] While *Plains* seems irreconcilable with

---

[74] Samuelson, *supra* note 44, at 61.

[75] *See supra* notes 58-61 and accompanying text.

[76] 605 F. Supp. 816 (M.D. Tenn. 1985).

[77] *Id.* at 830.

[78] 797 F.2d 1222 (3d Cir. 1986).

[79] *Id.* at 1248.

[80] *Id.* at 1235.

[81] *Id.*

[82] 807 F.2d 1256 (5th Cir. 1987).

[83] *Id.* at 1260, 1262.

Whelan, the fact that the program in *Plains* was greatly dictated by market forces may mitigate the lack of extension of copyright expression to nonliteral elements.[84]

The initial response to these cases was confusion. One commentator suggested that the law depended upon in which circuit one lived.[85] As time passed and other courts addressed further issues, a clear pattern began to develop that the *Whelan* court was to be followed, at least in terms of holding that copyright protection extends to some nonliteral elements of a program.

At this point, courts had only determined issues that related directly to the protection of the program code, whether literal manifestations (line-by-line code) or nonliteral elements (structure, sequence, and organization). The courts were relatively unanimous in protecting literal manifestations, pointing to CONTU for support. As the requested protection moved further and further away from the conclusions of CONTU into the realm of nonliteral elements of a program, the courts became more unsure and divided. When the commentators and courts had barely gotten used to the *Whelan* decision, the computer industry changed again, and software developers began to ask for further extended protection. The stage was set to attempt to extend copyright protection to the "look and feel" of a program.

B. *"Look and Feel" Litigation Up To* Lotus

Several post-*Whelan* cases have directly addressed the issue of the copyrightability of "look and feel." While the decisions are split, there seems to be a general trend towards extending nonliteral software copyright protection to some "look and feel" elements.

The first post-*Whelan* case to address the "look and feel" issue was *Broderbund Software v. Unison World*.[86] In *Broderbund*, the court borrowed from *Whelan* in holding that the "overall structure, sequencing and arrangement of screens" was copyrightable.[87] While this decision is significant in that it extended the nonliteral protection of software beyond mere code and into the realm of audiovisual displays, the misuse of the code-describing phrase from *Whelan* (structure, sequence, and organization) places

---

[84]The court in *Plains* held specifically that "market factors play a significant role in determining the sequence and organization of cotton marketing software, and we decline to hold that those patterns cannot constitute 'ideas' in a computer context." *Id.* at 1262.

[85] Schachter, *supra* note 47, at 55 ("At present, the scope of software copyright protection . . . depend[s], at least in part, upon geography.").

[86] 648 F. Supp. 1127 (N.D. Cal. 1986).

[87] *Id.* at 1128.

the theory of the decision in some doubt.[88] In terms of the idea/expression dichotomy, however, the court was quite explicit, holding that since there were many ways of expressing the purportedly infringing screen display, such display was obviously the expression of an idea and not the idea itself.[89] The court found that the display "was dictated primarily by artistic and aesthetic considerations, and not by utilitarian or mechanical ones."[90]

The initial reaction to *Broderbund* was as negative as the initial reaction had been to *Whelan*. One commentator suggested that the extension was "dangerous" in terms of first amendment consequences.[91] Other commentators, noting the confused use of language from *Whelan*, suggest that the *Broderbund* case is actually an equity case, in which the court was so outraged by the extent of copying that the court was going to find an infringement one way or the other.[92]

One year later, the District Court for the Northern District of Georgia completely disagreed with *Broderbund* in *Digital Communications Associates, Inc. v. Softklone Distributing Corp.*[93] The *Softklone* decision had two main parts: first, the court held that the copyright on a computer program does not extend to the nonliteral screen displays;[94] second, however, the court held that a second and independent copyright held by the plaintiff, one that covered the screen display as an audiovisual work, was valid and had been infringed.[95]

While the court did find an infringement, the *Softklone* case is significant for not finding an infringement based upon the copyrightability of a nonliteral aspect of code. *Softklone* and *Broderbund* stand on opposite ends of the "look and feel" spectrum. One year later, however, the precedential value of *Softklone* was to be put at issue by an action of the Copyright Office.

---

[88] In fact, the *Broderbund* court cites *Whelan* as standing for the proposition that protection extends "to the overall structure of a program, including its audiovisual displays." *Id.* at 1133. While *Whelan* discussed the screen displays in terms of distinguishing idea from expression, the *Whelan* court did not pass upon the copyrightability of an audiovisual display.

[89] *Id.* at 1132.

[90] *Id.* at 1134.

[91] Yen, *supra* note 8, at 416.

[92] Conley, *supra* note 42, at 4 (*Broderbund* is "a powerful equity case in which the court appropriated copyright language already in currency with little technical regard for the computer issues.").

[93] 659 F. Supp. 449 (N.D. Ga. 1987).

[94] *Id.* at 455. The court reasoned that the displays were not copyrightable because completely different programs could create the same display. *Id.* at 456.

[95] In discussing the idea/expression dichotomy, the court found that "[c]ertain aspects of the status screen . . . are unrelated to how the computer program operates and are [thus] 'expression.'" *Id.* at 459.

On June 6, 1988, the Copyright Office announced that it would only accept one copyright registration for a program, the registration of which would cover "all copyrightable expression . . . embodied in a computer program, . . . including computer screen displays."[96] The Copyright Office reasoned that "there is no authorship in ideas, or the format, layout or arrangement of text on the screen."[97] The Office, in the interest of explaining the present scope of copyright protection,[98] recommended that in order to avoid infringement, developers of software should avoid designing functions "in the same manner and with the same or a substantially similar sequence of menus, prompts, messages, and other visual display elements."[99]

Recognizing that the Office's decision taken in step with *Softklone* leaves no copyright protection for screen displays, the Copyright Office held a public hearing to discuss the matter and concluded that the interests of clear registration practice outweigh all other issues supporting *Softklone*. In other words, the Copyright Office decided to accept only one registration in spite of the *Softklone* holding. This, in effect, leaves the *Softklone* holding in doubt; the question being, would the court have denied nonliteral copyright protection via program code if the court had known that there would have been no copyright protection via a separate screen display copyright? No one will ever know for sure.

With the actions of the Copyright Office in mind, two other courts recently decided cases in favor of extending copyright protection to nonliteral elements of a program embodied in the program's "look and feel." The first of the cases, *Manufacturers Technologies, Inc. v. Cams, Inc.*,[100] straddled the line between the expansive *Broderbund* policy and the restrictive *Softklone* policy[101] in holding that copyright protection only extends to certain elements of a screen display. In doing so, *Manufacturers* continued in the general trend of protecting some nonliteral program elements.

---

[96] Russo, *Further Developments in Copyright Protection of the "Look and Feel" of Computer Software: Recent Copyright Office Regulations*, 7 SOFTWARE PROTECTION 1, 3 (June 1988). Previously, the Copyright Office had accepted two registrations; one for code, and one for screen displays. It is this dual registration that *Softklone* was decided upon.

[97] *Id.* at 5.

[98] The Copyright Office admits that "the registration practices of the Copyright Office cannot precisely determine the scope of protection in any work." *Id.* at 11.

[99] *Id.* at 2.

[100] 706 F. Supp. 984 (D. Conn. 1989).

[101] *See generally* Dorny & Friedland, *Copyrighting "Look and Feel":* Manufacturers Technologies v. Cams, 3 HARV. J.L. & TECH. 195 (1990).

Manufacturers specifically held that the flow of the screen displays,[102] the external sequencing of screens,[103] and the selection and arrangement of certain screens were copyrightable.[104] Copyrightability was denied to the format and placement of common components of screen pages,[105] the internal method of navigation,[106] alphabetical columns of information,[107] and certain specific screen displays.[108] The court did not define a bright-line test as to what made any given element copyrightable or not, instead proceeding by a seemingly ad hoc process. While the nature of this decision does not lead to gleaning a general rule, it does dispel the opportunity of over- or under-reading the case. *Manufacturers* stands for the proposition that certain nonliteral screen display elements of a program are copyrightable, but not all such elements are protectable.

In the most recent case prior to *Lotus*, *Telemarketing Resources v. Symantec Corp.*,[109] the District Court of the Northern District of California accepted the concept that copyright protection could extend to the "look and feel" of a program,[110] but found no copyrightable elements in the case.[111]

---

[102] *Manufacturers*, 706 F. Supp. at 994.

[103] *Id.*

[104] *Id.* at 996. Specifically, the screen found copyrightable included a menu of nine terms—the "selection and arrangement of the terms." *Id.* at 997.

[105] *Id.* at 995.

[106]

> By internal method of navigation, the Court is referring to: 1) the use of the space bar to move the cursor down a list of selection; 2) the use of the backspace key to move up a list of selection; 3) the use of the return key to choose or implement a selection or function; and 4) the use of number selection to change or edit an entry.

Id.

[107] *Id.* at 996.

[108] The court denied copyright protection to several screens for various reasons, including: "The use of a columnar format and the use of both upper and lower case letters are not sufficient on their own to warrant copyright protection because they lack originality." *Id.* at 998. Functional/utilitarian considerations also denied some protection. *Id.*

[109] 12 U.S.P.Q.2d 1991 (N.D. Cal. 1989).

[110] In a questionable reading of *Whelan*, the court held that "[c]opyright protection applies to the user interface, or overall structure and organization of a computer program, including its audiovisual displays, or screen 'look and feel.'" *Id.* at 1993.

[111] In a discussion that does not distinguish between whether the copyrightability is at issue or whether protection is granted and substantial similarity is at issue, the court held that certain menu options are fundamental to the purpose of the program, and thus not copyrightable, and that, while certain menus themselves were similar, the elements of the menus themselves were different, and thus not substantially similar. *Id.* at 1995-96.

It seems that, with a few exceptions, the recent trend of cases has been favorable to extending copyright protection to at least some nonliteral elements of a program's "look and feel."[112] Yet the cases weave a confusing pattern, relying on various wordings and viewing different elements as copyrightable. In 1987, two "look and feel" copyright cases were filed, both of which the legal commentators and the computer industry looked towards to eventually clear up the confusion. The first of the cases, a suit involving Apple, Microsoft, and Hewlett Packard, has been strung out and has yet to be decided. As interest and curiosity mounted, the second of the much-watched cases, the *Lotus* decision, was released.

IV. AN ANALYSIS OF THE *LOTUS* DECISION

Briefly stated, Judge Keeton in *Lotus* held that certain elements of Lotus 1-2-3's "user interface," specifically the structure of the menu command system, were subject to copyright protection under a copyright registered to the Lotus 1-2-3 program, and that, as a matter of law, Paperback Software infringed upon Lotus' copyright by producing a spreadsheet with substantially similar menu command systems.

Judge Keeton's decision can almost be regarded as a mini-treatise in its length and breadth. While at times slipping, it is a generally sound and well-written work. The crux of the decision revolves around the idea/expression dichotomy, but elements of functionality, substantial similarity, and several other issues are discussed and decided. Each of these areas will be discussed independently.

A. *The Application of the Idea/Expression Dichotomy to* Lotus

The main issue in *Lotus* is exactly which elements of Lotus 1-2-3 are copyrightable expressions of ideas, and which are ideas or noncopyrightable expressions of ideas. Judge Keeton delivers a very tight, narrow conclusion of what is protectable, and emphasizes the ad hoc nature of such decisions.[113]

---

[112] But note: No courts higher than district courts have ruled on the issue.

[113] First, a brief note on the phrase "look and feel." While *Lotus* has been generalized as a "look and feel" copyright case, Judge Keeton himself shies away from the term. Judge Keeton viewed the concept as wide-open and "[not] significantly helpful . . . ." *Lotus*, 740 F. Supp. 37, 62 (D. Mass. 1990). Keeton attempted an analogy between "look and feel" and "total concept and feel," concluding that any such analogy would lead to confusion. *Id.* at 63. Instead, Judge Keeton focused in on the "user interface" as a sub-set of "look and feel." *Id.* One of the many benefits of Judge Keeton's decision is that he focused even further into specific elements of the "user interface" in terms of copyrightability. *Lotus* does not stand for

In determining what is idea and what is expression, Judge Keeton lists four necessary elements for extension of copyrightability:

> If . . . the expression of an idea has elements that go beyond all functional elements of the idea itself, and beyond the obvious, and if there are numerous other ways of expressing the noncopyrightable idea, then those elements of expression, if original and substantial, are copyrightable.[114]

The requisite four elements are as follows: First, the expression must be original;[115] second, the expression must have elements separate from functionality;[116] third, an expression must not be obvious;[117] and fourth, there must not be a limited number of ways of expressing the idea (merger).[118]

These are four difficult elements to determine, indeed. Judge Keeton notes that the determination of copyrightability is not, in this instance, black and white, but necessarily "a matter of degree."[119] He also notes that complete disentanglement of an expression from its idea is not required in order to extend protection.[120]

To aid in determining these elements, Judge Keeton adopts a three-pronged "legal test":

> FIRST, . . . [in separating idea from expression], the court may conceive, *along the scale from the most generalized conception to the most particularized*, and choose some formulation—some conception or definition of the "idea"—for the purpose of distinguishing between the idea and its expression.
>
> . . . .
>
> SECOND, the decisionmaker must focus upon whether an alleged expression of the idea is limited to elements essential to expression of *that* idea (or is one of only a few ways of expressing the idea) or instead

---

any sweeping generalities upon the copyrightability of the "look and feel" of a program. *See supra* notes 51-57 and accompanying text. *Lotus* stands for the copyrightability of certain elements of a program's "user interface."

[114] *Lotus*, 740 F. Supp. at 59.

[115] That is, the expression must have originated from the author. *Id.* at 58. *See supra* notes 10-16 and accompanying text.

[116] *Lotus*, 740 F. Supp. at 58. *See infra* notes 141-46 and accompanying text.

[117] An obvious expression would be inseparable from its idea, and, if protectable, would occupy the field of the idea. *Lotus*, 740 F. Supp. at 58-59.

[118] *Id.* at 59. *See also, supra* notes 10-30 and accompanying text.

[119] *Lotus*, 740 F. Supp. at 60.

[120] *Id.*

> includes identifiable elements of expression not essential to every expression of that idea.
>
> THIRD, having identified elements of expression not essential to every expression of the idea, the decisionmaker must focus on whether those elements are a substantial part of the allegedly copyrightable "work."[121]

The first prong of the test, or the "sliding scale" test, is modeled after a test devised by Judge Learned Hand;[122] idea is considered at one end of the scale, and expression at the other. As one moves away from idea, a concept becomes less and less of an idea and more and more of an expression, or vice versa when moving away from expression.

While this is obviously not a bright-line test easily useable to determine if an element is idea or expression, it seems to be an accurate test in terms of software elements. Consider: At the highest point of abstraction, everything in software today can be considered an idea,[123] while at the lowest point of abstraction (indeed, no abstraction at all), many elements of software are expressions.[124] If every part of "look and feel" or a "user interface" were considered an idea, then there would be no protection for any elements of nonliteral, noncode parts of programs. This would seem to go against logic[125] and the general trend of the case law.

Very few issues in copyright are black or white, and few works are completely idea or completely expression. The sliding scale test allows a judge to measure the "gray" of an element, to look at all surrounding factors and determine if statute, judge-made law, or policy dictate a finding one way or another. Such decisions, as Judge Hand stated over thirty years ago, are "inevitably . . . ad hoc."[126] As soon as one accepts the fact that present-day copyright law forces such decisions to be ad hoc, one is closer to

---

[121] *Id.* at 60-61.

[122] *See* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

[123] The idea of a spreadsheet, or a wordprocessor, or a disk operating system are examples.

[124] What color to make the background, how big should the characters be, what order should the commands go in? These are concepts that are generally not ruled by function, but are instead open to the creative discretion of the programmer.

[125] If every element were considered an idea (i.e., the simple ordering of commands within a menu, assuming no functional constraints) then what word or concept would be used to describe the initial action of deciding to group the commands together in a menu in the first place? It seems much more logical, and much closer to the standard usage of the words, to consider the grouping of commands into a menu to be the idea, and the ordering of the commands to be a specific expression of the idea.

[126] *Lotus*, 740 F. Supp. at 60, *citing* Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) (L. Hand, J.).

understanding the narrowness of the *Lotus* decision. Any attempt to glean a wide-sweeping rule would violate this ad hoc nature, and would make the purported rule of doubtful value.

The disadvantage of the sliding scale is the lack of a bright-line test. As Judge Keeton notes, there are disadvantages to applying a bright-line test to a gray area such as this.[127] Any bright-line rule devised in this case may lead to injustice if applied, for instance, to a case involving icons or a database. The opposite side of the coin, of course, is that a bright-line test would give the software industry a guideline to follow to avoid litigation. Since a bright-line test seems a near practical impossibility, this choice does not have to be made.

The second prong of the test deals generally with the concept of merger. If there are only a few ways of expressing an idea, then the idea and expression merge, and copyright expression is not extended.

The third prong of the test denies copyrightability if the element in question is insignificant when compared to the work as a whole. In determining whether or not an element is of "significant" stature, a court must weigh qualitative and quantitative matters.[128]

After stating the test, Keeton applies the test to Lotus 1-2-3 and VP Planner, finding several elements that are not copyrightable and several that are. Judge Keeton first holds that an electronic spreadsheet itself is an idea that is not copyrightable.[129] From this base, "[t]he issue [becomes] whether Lotus 1-2-3 . . . go[es] beyond those details essential to any expression of the idea, and includes substantial elements of expression, distinctive and original, which are thus copyrightable."[130]

Several items are explicitly held not copyrightable: first, the general "L" shape of the Lotus spreadsheet;[131] second, the use of the backslash ("\") key to invoke the menu command;[132] third, all arithmetic elements;[133]

---

[127] "[I]n many circumstances hard-and-fast rules, despite their initial attractiveness and false promise of certainty, have consequences that offend one's sense of justice." *Lotus*, 740 F. Supp. at 73.

[128] *Id.* at 61.

[129] "Defendants are quite correct . . . in asserting that the idea of developing an electronic spreadsheet is not copyrightable . . . ." *Id.* at 65. Judge Keeton cites other spreadsheets, such as VisCalc, Multiplan, and Excel as other expressions of the spreadsheet idea. *Id.*

[130] *Id.*

[131] A lack of alternative shapes makes the "L" fail the second prong of the test. *Id.* at 66.

[132] The limited number of keys means that the expression here merges into the idea. *Id.*

[133] These are viewed as essential to every mathematical expression. *Id.*

fourth, the individual elements (commands) of the menus;[134] and fifth, the two-line moving cursor menu.[135]

One element, however, was explicitly held to be copyrightable: namely, "the menu structure, taken as a whole—including the choice of command terms, the structure and order of those terms, their presentation on the screen, and the long prompts . . . ."[136] The menu is said to pass the first prong of the test because the general concept of the menu is the idea, while the "distinctive details of expression . . . ," namely "the precise 'structure, sequence, and organization' . . . of the menu command system"[137] is considered expression. The second prong of the test is passed because the menu command system is "an aspect of 1-2-3 that is not present in every expression of an electronic spreadsheet."[138] The answer to the third prong of the test is "incontrovertibl[y]" yes because "[t]he user interface of 1-2-3 is its most unique element, and is the aspect that has made 1-2-3 so popular."[139]

Judge Keeton follows this very specific discussion and holding with a very confusing statement that, if correct, denies all of the preceding logic or, if wrong, will only serve to confuse the interpretation of the case by encouraging commentators to broaden the holding. Judge Keeton concludes: "Taking account of all three elements of the legal test, I determine that copyrightability of the user interface of 1-2-3 is established."[140] Since this sweeping statement would negate the specific holdings of noncopyrightability, this Paper will interpret this statement as a careless overstatement, undisruptive of the actual holding.

To summarize, *Lotus* holds that only one nonliteral, noncoded element of the spreadsheet Lotus 1-2-3 is copyrightable, namely, the menu command system. This is a very narrow holding. While *Lotus* holds that copyrightability *may* extend to nonliteral elements of a "user interface," the case itself extends the copyrightability to only one element of the "user interface."

---

[134] Some are either obvious or merge with idea. *Id.* at 67.

[135] "The idea for a two-line moving cursor menu is also functional and obvious, and, indeed, is used in a wide variety of computer programs including spreadsheet programs." *Id.* at 65.

[136] *Id.* at 68.

[137] *Id.* at 67 (citations omitted).

[138] *Id.* at 68.

[139] *Id.*

[140] *Id.*

B. *Functionality: Perhaps a Dangerous Precedent*

1. *The Question of Functionality and Lotus 1-2-3*

Also at issue in *Lotus* was whether or not functional aspects of Lotus 1-2-3 precluded copyright protection. Paperback claimed that Lotus 1-2-3 was itself an inherently useful article, and, even if not inherently so, that the immense popularity of Lotus 1-2-3 made the program the industry standard in terms of spreadsheets, making the program a useful article. Judge Keeton ruled against both arguments.

In terms of Lotus 1-2-3 being inherently "useful," Judge Keeton recognized this fact,[141] but also noted that the concept of separability allowed the copyrighting of separable, nonfunctional aspects of a work if independent from functional ideas.[142] Keeton justified this view by noting that the concept of functionality must not completely destroy protection of expression.[143] As previously discussed,[144] this allowed Judge Keeton to separate one specific part of Lotus 1-2-3, the menu structure, label it nonutilitarian, and uphold its copyrightability.

In terms of the usefulness of Lotus 1-2-3 being proved by its immense popularity and standardization, Judge Keeton invoked almost a quasi-equity argument, claiming:

> It does not follow that when an intellectual work achieves the feat of being useful as well as expressive and original, the moment of creative triumph is also a moment of devastating financial loss—because the triumph destroys copyrightability of all expressive elements that would have been protected if only they had not contributed so much to the public interest by helping to make some article useful.[145]

Keeton's equity argument, however, is at odds with other policy arguments.[146]

---

[141] "For example, Lotus 1-2-3 is surely 'useful.'" *Id.* at 57.

[142] *Id.* at 58.

[143] "I conclude that a court, in determining whether a particular element is copyrightable, must not allow one statutory mandate—that functionality or usefulness is not itself a basis for copyrightability—to absorb and destroy another statutory mandate—that elements of expression are copyrightable." *Id.*

[144] *See supra* notes 113-40 and accompanying text.

[145] *Lotus*, 740 F. Supp. at 57.

[146] *See infra* notes 165-75 and accompanying text.

2. *Functionality, Copyrightability, Macro Languages, and the Structure of Menu Commands*

In addition to the issue of the general useful nature of Lotus 1-2-3, Paperback raised another independent claim of functionality—namely, that since the structure, sequence, and organization of Lotus 1-2-3's menu command system was an integral part of Lotus 1-2-3's macro language, that such participation made the command structuring useful, or, alternatively, that the command structuring was part of a programming language, which itself should not be copyrightable.

In a short subsection of the *Lotus* decision descriptively entitled "Strained Analogies and Word Games," Judge Keeton summarily dismissed the above claims as "totally without merit."[147] Judge Keeton's reasoning behind his holding, combined with the confusion surrounding the discussion of this issue, have made the precedential value of *Lotus* in the area of the copyrightability of programming languages,[148] as one commentator has suggested, "particularly troublesome."[149]

Judge Keeton's explanation for dismissing Paperback's claim in this instance flatly states that "the argument depends on arbitrary definitions of words, adopted for undisclosed reasons."[150] While it is unclear whether Judge Keeton's terse dismissal of the argument was based upon complete

---

[147] *Lotus*, 740 F. Supp. at 73.

[148] The issue of the copyrightability of a programming language has not been addressed by the courts. A recent case, Ashton-Tate Corp. v. Fox Software, Inc., which placed the copyrightability of a macro programming language directly at issue, was originally dismissed via summary judgment on grounds of inequitable conduct. The case has recently been rejuvenated by the rescinding of the previous order to dismiss; Ashton-Tate Corp. v. Fox Software Inc., CV 88-6837 TJH (Tx) (C.D. Cal. Apr. 18, 1991) (LEXIS, Genfed library, Dist. file), and a decision on the merits would shed a great deal of light on this issue.

While the copyrightability of a computer programming language is beyond the scope of this Paper, a brief note is merited because of the impacts of the issue on the copyrightability of "look and feel." Copyright protection of "look and feel" may be strengthened or weakened depending upon the copyrightability of a macro language that uses part of the "look and feel" as part of that language. Specifically, if programming languages are found to be copyrightable, then any element of "look and feel" that is intertwined with a language (such as a macro language) would have another leg to possibly support copyright protection. Or, on the other hand, if languages are not copyrightable, then some "look and feel" elements that are intertwined with languages (such as menu ordering) may have to survive an attack on inseparable functionality.

[149] Abramson, *Why* Lotus-Paperback *Uses the Wrong Test and What the New Software Protection Legislation Should Look Like*, 7 COMPUTER LAW. 6, 8 (Aug. 1990).

[150] *Lotus*, 740 F. Supp. at 72.

disfavor of the argument,[151] or a lack of understanding on the issue,[152] two results of the dismissal are clear: first, Judge Keeton did not believe that the use of menu commands in a macro language made the structure, sequence, and organization of the commands functional, and second (a double negative) Judge Keeton did not hold that programming languages were uncopyrightable.[153]

The implications of the first result mentioned above are somewhat troublesome. Implicit in Judge Keeton's extension of copyright protection to the menu structure is that the menu structure itself is either nonfunctional (wholly expressive in nature) or has expressive, protectable elements separable from functional elements. Yet both of these implications seem contradicted by analysis elsewhere in *Lotus*. First, as Judge Keeton himself stated, albeit in dicta, that "[b]ecause macros may contain many menu choices, the exact hierarchy—or structure, sequence, and organization—of the menu system is a fundamental part of the *functionality* of the macros."[154] Second, the functional and/or useful nature of the menu command system was considered an integral part of Lotus 1-2-3's macro language,[155] as

---

[151] Judge Keeton's tone makes this disfavor obvious. *See id.* at 71-72.

[152] While basing his dismissal of Paperback's argument on purported bad usage of language, Judge Keeton himself seems to create great linguistic confusion in attempting to explain his understanding of Paperback's argument; specifically, Judge Keeton seems to confuse the phrase "sets of statements or instructions" as the copyrightable work derived from a noncopyrightable "language." In general, when referring to the syntax of a language, particular words, symbols, or phrases that are part of the syntax of a language are called instructions of that language. It is this syntax, of which the menu command structure is part, that creates the questions and confusion surrounding Paperback's functionality argument. Whether Judge Keeton understood this use of the phrase "instructions of that language" to mean syntax (the correct understanding) or the actual code that is written using a language (the incorrect understanding, but the one that fits Judge Keeton's reasoning), we will never know.

[153] *See Lotus*, 740 F. Supp. at 72. Keeton did not hold directly whether or not a programming language was copyrightable, instead noting that Paperback had failed to show any case or statutory law against language copyrightability, and concluding that the macro-functionality argument had no bearing on his decision. *Id.*

[154] *Id.* at 65 (emphasis added).

[155] A macro command is actually a collection of standard commands given to Lotus 1-2-3 by a user. Since a user may actuate certain Lotus functions by using cursor keys to highlight certain functions listed in a menu, the actual order of the functions within the menu is important in describing which function is to be actuated. For example: if a user wishes to perform the third function (from the left) listed on a menu, the user will press the right arrow key twice (highlighting the third function), then press the enter key to actuate the function (or move to sub-menu). In order to create a macro program to actuate the same function, a user stores three commands: two right arrow keystrokes, and the enter keystroke.

Judge Keeton himself explained.[156] Additionally, Paperback claimed that the issue of compatibility with Lotus 1-2-3's macro language was the driving factor in choosing to copy the Lotus 1-2-3 command structure.[157]

Third, and finally, the separability of function from expression is questioned by the nature of the development of the interaction between Lotus 1-2-3's macro language and menu command system. Consider: Which came first, an *aesthetically* driven, nonfunctionally designed menu hierarchy followed by the accompanying macro language, or a *functional* macro language followed by the necessarily functional menu hierarchy designed to interact with the macro language? Judge Keeton held that the expressive nature of the menu structure was separable from the functional nature. To hold this, one must assume that the menu hierarchy was created first, by nonfunctional considerations ("File" *feels* better to the left of "Exit"), and that the macro function in essence was created "next," using the nonfunctional ordering of commands to create the macro language. Yet, at this point the expression of the menu order seems indivisibly inseparable from the language because one still depends on the other. If either changes, the other does not function. At this point, has the menu structure crossed the inseparable functional line? On the other hand, if the concept of the macro language was developed before the hierarchy of the menu commands, then there would seem to be a stronger argument for viewing the menu structure as a functional element of the macro language. The ordering of the functions would then not be based on expressive elements, but on functional constraints of conforming to or building a programming language.

Regardless of the reasons for dismissal, the dismissal of Paperback's argument denied an important question in determining which "look and feel" elements can be afforded copyright protection. At an extreme, if Judge Keeton had seen the menu command structure as functionally entwined with

---

[156] *See Lotus*, 740 F. Supp. at 64-65.

[157]

> [M]aking the changes required for macro compatibility meant that we had to revise existing elements of the [VP Planner] spreadsheet interface, including the hierarchical menu structure; [and] ensure that keystroke sequences would bring about the same operational result in both programs . . . .
>
> . . . .
>
> Several types of changes were required in the VP-Planner [sic] program to achieve keystroke macro compatibility. First, the menu structure had to be altered so that all menu commands would have the same first letter and be in the same location in the menu hierarchy as in Lotus 1-2-3.

Id. at 69 (quoting Stephenson Affidavit ¶¶ 144, 146, Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990)).

the macro language, he may have ruled the case completely differently. At another, more dangerous extreme, *Lotus* may be read as encouraging the copyrightability of a programming language; a conclusion that was *not* directly stated in Judge Keeton's holding. At this point in the *Lotus* decision, Judge Keeton had established and defended the extension of copyright protection to one element of Lotus 1-2-3; next would come the test of infringement—whether or not VP Planner was substantially similar to Lotus 1-2-3.

C. *Substantial Similarity: The Test for Infringement*

Once copyright protection is extended, an issue remains as to "whether the alleged infringing work, measured by the 'substantial similarity' test, did contain elements that infringed upon the copyrightable elements of the copyrighted 'work.'"[158] The issue thus became whether or not VP Planner's menu command system was substantially similar to Lotus 1-2-3's menu command system.

Judge Keeton first held that, since the copying was so "overwhelming and pervasive; . . . the defendants . . . have *admitted* that they copied,"[159] that as a matter of law any question of independent creation was precluded. Judge Keeton then discussed the slight differences in detail between the programs,[160] concluding that even with such differences, the programs were "substantially, indeed, strikingly, similar."[161]

In terms of the substantial similarity test, Judge Keeton listened to expert testimony and attempted to place himself in a lay observer's position, concluding that "[f]rom the perspective of both an expert and an ordinary

---

[158] *Id.* at 61.

[159] *Id.* at 68.

[160] Judge Keeton lists the following:

> Most VP-Planner [sic] menu lines begin with a help ("?") command, and some additional commands are included at the end of some menu lines (*i.e.*, "DBase, Multidimensional" on the "/File Erase" menu line; and "Page #, No Page #, Row/Col. #, Stop Row/Col. #, Background" on the "/Print Printer Options Other" menu line). Other differences between the two programs appear in the start-up screens, the placement on the screen of the menu lines, the exact wording of the long prompts, the organization of the help screens, the increased width of the VP-Planner [sic] screen, and the ability of the VP-Planner [sic] to hide certain columns.

Id. at 70.

[161] *Id.*

viewer, the similarities overwhelm the differences."[162] Judge Keeton also concluded that "there is no genuine dispute of material fact,"[163] and found liability as a matter of law.

It is unclear exactly what amount of differences would have been necessary to avoid a holding of substantial similarity. If the differences listed by Keeton were indeed trivial, then one wonders what amount of differences would be considered substantial. Judge Keeton made several observations to the effect that the commands in the menus were obvious copies because the commands were even in the same order for both menus. What if one were to merely change the order of the commands within the menu? It is quite possible that *Lotus* is narrow enough to suggest that the mere re-ordering of commands within a menu is a noninfringing entirely different idea rather than an infringing expression. How far, and to what extent, one "look and feel" may be similar to another and not infringe is not clearly stated in *Lotus*. One extreme, however, is clear: an *exact* copying of the order of commands within a menu will be deemed substantially similar.[164]

D. *The Role of Policy in* Lotus

Perhaps the greatest amount of discussion and debate leading up to *Lotus* surrounded the policy issues inherent to software copyright protection.[165] The majority of commentators, including a vast amount from the software industry, feared that strong protection of "look and feel" would lead to

---

[162] *Id.*

[163] *Id.*

[164] Defendants in *Lotus*, however, had reason to copy the order of the commands, or so they claimed. Paperback claimed that the order of the commands was a significant part of Lotus 1-2-3's macro language, and to ensure compatibility with Lotus macros, that the "only way to accomplish this result . . . was to ensure that the arrangement and names of commands and menus in VP-Planner [sic] conformed to that of Lotus 1-2-3." *Id.* at 69 (quoting Stevenson Affidavit ¶ 117, *Lotus*, 740 F. Supp. at 69). Paperback further claimed that compatibility was an economic requirement to survive in the spreadsheet market. Judge Keeton held against all of these claims, stating first, that the success of incompatible Excel proved that success does not depend upon compatibility; second, that compatibility could be achieved by other, noninfringing ways, such as macro conversion programs; and finally, in terms of policy, "the desire to achieve 'compatibility' or 'standardization' cannot override the rights of authors to a limited monopoly in the expression embodied in their intellectual 'work.'" *Id.*

[165] The details of the policy arguments themselves are outside of the scope of this Paper. This Paper is an attempt to discuss the legal issues, the reasons surrounding them, the exact holding, and the impact of such holding. This is not, however, intended to diminish the importance of the policy argument. As Judge Keeton himself noted, such arguments are central and crucial in their place—namely, the legislature or higher courts. This Paper, however, will continue to confine itself to the *Lotus* case itself.

monopolies, increased costs, and decreased standardization, which would all lead to decreasing the vitality of the software industry.[166] On the other side, a minority of commentators claimed that protection with teeth was required to protect small developers, innovation in general, and investment in research.[167]

In his decision, however, Judge Keeton rejected the antiprotection policy arguments that he claimed were inconsistent with congressional legislation in the 1976 Copyright Act and CONTU.[168] While, previously in the decision, Keeton noted that Congress had not directly spoken on the issue of "look and feel" protection, Keeton also noted that CONTU had been aware of such policy arguments, and that such arguers were in the minority.[169] Keeton stated that, at the district level, courts have limited functions, one of which is to follow legislative policy when answering questions not directly governed by statute. As such, any policy arguments against protection, and thus against Keeton's reading of CONTU, were fundamentally flawed.[170]

In terms of expert testimony on the policy arguments, Keeton allowed (over objection) such arguments, but, for the reasons stated above, Keeton declined to use the testimony in his decision.[171] Additionally, Keeton noted that he would have no way of determining which group of experts were correct, and that he was not in a position to make such a determination even if he could.[172]

---

[166] *See Computer Software and Copyright Protection: The "Structure, Sequence, and Organization" and "Look and Feel" Questions*, SOFTWARE PROTECTION 7, 14-15 (July 1989); *Staff Paper*, *supra* note 30, at 13; Menell, *supra* note 58, at 1068; Spector, *Software, Interface, and Implementation*, 30 JURIMETRICS J. 79, 87-88 (1989); Note, *supra* note 41, at 977; Bendekgey, *supra* note 57, at 1; Comment, *supra* note 2, at 978, 997-98, 1004, 1006; Farrell, *Standardization and Intellectual Property*, 30 JURIMETRICS J. 35-37, 49 (Fall 1989); Clapes, Lynch, & Steinberg, *supra* note 65, at 1508; Curtis, *supra* note 38, at 59, 75-77; *LaST Frontier Conference Report on Copyright Protection of Computer Software*, *supra* note 44, at 26-28; Samuelson & Glushko, *Comparing the Views of Lawyers and User Interface Designers on the Software Copyright "Look and Feel" Lawsuits*, 30 JURIMETRICS J. 121, 137 (1989); Hemnes, *Three Common Fallacies in the User Interface Copyright Debate*, 7 COMPUTER LAW. 14, 15, 17 (Feb. 1990).

[167] *See generally* Note, *supra* note 41, at 976-77, 985; Farrell, *supra* note 166, at 48; Spector, *supra* note 166, at 88; Curtis, *supra* note 38, at 76; Menell, *supra* note 58, at 1068, 1094, 1100; Schachter, *supra* note 47, at 59; Clapes, Lynch, & Steinberg, *supra* note 65, at 1509; Hemnes, *supra* note 166, at 16.

[168] *Lotus*, 740 F. Supp. at 71.

[169] *Id.* at 77.

[170] *Id.* at 71.

[171] *Id.* at 73-75.

[172] *Id.*

In sum, Keeton seems to be saying that changes based on such policy arguments should be legislative, not judicial. Or, to an extreme, any such judicial answers to policy questions would have to be made by higher courts in the land.[173] Judging from Keeton's tone, it is unclear whether he would have ruled in favor of the antiprotectionist policy arguments even if he had held on the policy issue. Keeton cites pro-protectionist arguments quite favorably,[174] and seems to hint that his decision is much like CONTU's; since protection has been extended to areas about which the commentators predicted doom, and since doom does not seem to have arrived, Keeton disregarded all new predictions of gloom.[175]

Judging from the reaction to *Lotus*, it does not appear that the "look and feel" protectionist policy argument will go away any time soon. In fact, it appears that this issue will be debated for some time, in both courts and legislatures. In the remainder of this Paper, the debate will generally be discussed in light of the *Lotus* decision. First, a brief survey of the software industry will give a hint as to the empirical validity of the policy arguments, and then the future of "look and feel" protection will be discussed, along with a modest proposal.

V. THE IMPACT OF *LOTUS*: INDUSTRY OPINION AND RESPONSE

The initial computer industry response to the *Lotus* decision was negative.[176] Critics instantly declared the decision as the harbinger of destruction to the software industry,[177] signalling a chilling effect on the

---

[173] Since the case has been settled, however, we will never know.

[174] *See Lotus*, 740 F. Supp. at 75-77.

[175] *Id.*

[176] As one commentator suggested: "Right now, all we're doing is going backward, unfortunately." Pane, *supra* note 46, at 46. This, however, is not a new position. Even before the *Lotus* decision, copyright protection of software was not very popular with the software industry. *See* Samuelson & Glushko, *supra* note 166, at 126 (citing survey response of software legal and industry conference: "Seventy-seven percent of the respondents with an opinion felt that 'look and feel' should not be given protection by either copyright or patent law. Only 18 percent thought that copyright law should protect 'look and feel' or user interfaces."); Seymour, *The Case Against 'Look and Feel' Lawsuits*, 4 PC WEEK 34 (Mar. 17, 1987) ("Let's not mince words here: Look-and-Feel lawsuits stink.").

[177] *See* Abramson, *supra* note 149, at 9 ("*Lotus - Paperback*, if followed, will produce a line of cases creating unwarranted monopolies."); Greguras, *supra* note 8, at 4, 5 ("This decision will make it more difficult for developers to create new software products which are competitive with successful programs.") ("Software developers . . . can use it aggressively against others as Lotus is doing.") This, again, is not a new fear, occurring years before the *Lotus* holding. *See* Churbuck and Freedman, *Suits Against 1-2-3 Imitators may have Wide*

industry[178] and a wave of new law suits.[179] This, however, so far has proven to be untrue;[180] the software industry, along with Lotus' direct

---

User Impact, 4 PC WEEK 1, 2 (Jan. 20, 1987) ("Developers will be forced to create products so different that people will be afraid to buy them because they are hard to learn." (quoting Wayne Maples, Information-Center Consultant at the Federal Reserve Bank in Dallas)).

[178] *See* Pane, *supra* note 46, at 51 ("Amid this debate, it seems clear that the ruling will, if not chill, at least cool the industry."); Lewis, *The Executive Computer; When Computing Power is Generated by the Lawyers*, N.Y. TIMES, July 22, 1990, § 3, at 4, col. 1. ("'The uneasiness of the industry shot up after Lotus sued Borland, . . . Lotus winning the Paperback suit has had an enormous destabilizing effect on the industry. The whole thing is starting to unravel and nobody knows what is going to happen.'" (quoting Mitchell Kapor, founder of Lotus and now head Technology, Inc., of Cambridge, Mass.)); Darrow, *Lawsuits Muzzle Industry Spokesmen*, INFOWORLD 46 (July 23, 1990) ("The ongoing flurry of copyright infringement cases has already taken its toll on software developers . . . [e]ven normally outspoken executives are loath to comment on any cases without first consulting with counsel.").

[179] *See Copyright Suits Could Slow Innovation*, INFOWORLD 1, 2 (July 9, 1990) ("We're going to see companion cases—progeny of the Lotus decision." (quoting John Yates, Partner at Morris, Manning & Martin)).

[180] While it is impossible to measure the chilling effect on an industry, and equally impossible to point to software packages which did not get past the design phase due to litigation fear, the U.S. software industry seems just as competitive and innovative as before the *Lotus* decision, with a noticeable lack of monopolies in the spreadsheet sector. *See* notes 176-84 and accompanying text; Zachmann, Lotus-Paperback *Precedent Need Not Harm the Industry*, 7 PC WEEK 10 (July 9, 1990) ("The present decision, however, gives no cause for alarm . . . . The precedent it sets, as far as it goes, is a reasonable one that need not do the industry harm.").

Those who foresee destruction have another problem with the validity of their predictions; namely, why hasn't doom already set in? The "industry doom" argument was proffered after the *Whelan* case and when the Lotus suits were filed, yet doom has not been realized. *See* Furger and Parker, *Software on Trial; Are Look and Feel Lawsuits Putting a Stranglehold on Innovation or are They Just a Sign That the Industry has Grown Up?*, INFOWORLD 31, 36 (Jan. 9, 1989).

> Each time a suit is filed, users, vendors, and analysts are quick to predict the doom and gloom that will be cast over the industry. . . . In reality, these dire results haven't come to pass. . . . [W]hen Lotus sued Paperback and Mosaic, critics cried that all development of compatible products—including the add-ons—would come to a screeching halt. But, in fact, the spreadsheet market shows few scars from the suits.

Parker, *Suits Have Had Surprisingly Little Effect on Software Industry*, INFOWORLD 41 (Oct. 9, 1989) ("But for all the ink and breath spent on the look and feel suits, very little has changed."); Parker, *Copyright Law's Haziness Obstacle for Developers*, INFOWORLD 27 (July 13, 1987) ("But six months after the suits were filed, the impact seems less dramatic, with most developers proceeding with business as usual.").

competitors, appears to be doing fine,[181] and the myriad of litigation has not yet appeared.[182]

---

[181] In general, software sales are up, *see* Coale & Picarille, *Windows Fervor Spurs Development in 1990; Mac Software Hurt by Delay of System 7.0*, INFOWORLD 39 (Jan. 21, 1991) ("Software sales grew 50 percent in the first six months of 1990 . . . ."). Specifically, software sales of competing spreadsheet Quattro Pro, by Borland International, a subject of a subsequent Lotus lawsuit, continue to be strong. *See* Quinlan, *Object-Oriented Products Spur Borland's Revenue Surge*, INFOWORLD 54 (Oct. 22, 1990) ("Sales also more than doubled . . . . Even the company's ongoing legal problems with Lotus . . . haven't materially affected the company's performance, as users are still willing to buy the Quattro Pro spreadsheet . . . .") and Picarille, *supra* note 5 (garnishing 24 percent of the market). The resolution of the *Lotus-Borland* suit may have an even greater impact upon the industry than the *Lotus-Paperback* suit, for Quattro Pro is generally recognized as a technical improvement over Lotus, allowing users to simulate the Lotus interface as an interface option, while offering a unique interface of its own. *See generally* Lewis, *supra* note 178.

[182] Initial fears of expansive litigation were fueled by Lotus itself, filing two suits the week after the *Lotus* decision. (Filing against Borland International, Inc. for Quattro Pro and Santa Cruz Operation Inc. for SCO Professional.) *See generally* Greguras, *supra* note 8. After these suit filings, however, there has not been a marked increase in software litigation over the pre-*Lotus* levels.

An informal computer-database survey of several computer and legal services has revealed two software copyright suits having been filed in the first year after *Lotus*, Weyerhaeuser Co. v. Osmose Wood Preserving Inc. (*see* Garner, *DOS Developer Sues Over App's Mac Look, Feel*, 4 MACWEEK 97 (Sept. 11, 1990)) and Icot Corp. v. American Airline, Inc. (*see* Scheier, *Icot Sues American, Claims Airline Copied Reservation Software*, 8 PC WEEK 4 (May 13, 1991)). This may be compared to the six month period previous to the *Lotus* decision, in which three suits were filed: Software Publishing Corp. v. Computer Support Corp. (filed Apr. 13, 1990); *see* Pane, *CSC Countersues SPC Over Clip Art; CSC Claims Copyright Infringements by Harvard Graphics Programs*, INFOWORLD 8 (June 18, 1990), Pane, *SPC, CSC Settle Harvard Graphics Copyright Dispute*, INFOWORLD 38 (Aug. 6, 1990); Intel v. AMD (filed week of Apr. 30, 1990); *see* Pane, *Intel Sues AMD for Allegedly Infringing Microcode Copyrights*, INFOWORLD 6 (Apr. 30, 1990), Pane, *AMD Wins Right to Sell 80C287 Coprocessor*, INFOWORLD 51 (Aug. 20, 1990); and Peter Norton Computing Inc. v. Fifth Generation Systems Inc. (filed May, 1990); *see* Lyons, *Battle of the Backups Escalates as Fifth Generation Sues Norton*, 7 PC WEEK 146 (May 14, 1990).

This in turn may be compared to the number of patent computer suits reported as filed in 1990: two before *Lotus* and five after. *See In Focus Sues CA*, INFOWORLD 40 (May 14, 1990) (Focus Systems Inc. v. Computer Accessories, May 1990); Coale, *Adobe Files Patent Suit Against EFI*, INFOWORLD 3 (June 4, 1990) (Adobe Systems Inc. v. Electronics for Imaging, June 1990); *TI Sues Five*, INFOWORLD 39 (July 16, 1990) (Texas Instruments v. Analog Devices Inc., July 1990); Coursey, *TI Sues Dell Over Claimed Patent Infringements*, INFOWORLD 51 (Sept. 24, 1990) (Texas Instruments v. Dell Computer Corp., Sept. 1990); Miller, *New Laws Needed for Intellectual Property*, INFOWORLD 29 (Dec. 24/31, 1990) (Hayes Microcomputer Products Inc. v. Ven-Tel Inc., Dec. 1990); Pane, *Cyrix, Intel Head for Legal Tangle*, INFOWORLD 3 (Dec. 24/31, 1990) (Cyrix v. Intel, Dec. 1990). Not all of these suits were software related, but the amount of patent litigation in software is increasing

Perhaps the main reason for these unfounded fears of industry desolation are incorrect and over-broad readings of the *Lotus* decision itself.[183] While many commentators have correctly read the *Lotus* decision narrowly,[184] the general split in authority has lead to great confusion in the industry over exactly what is protected and what is not. It would seem that the greatest danger to the industry does not lie in the *Lotus* holding itself, but in the uncertainty that surrounds the legal interpretation of the holding.

Through the confusion of the status quo, the message of *Lotus* should be made clear: innovative change is still encouraged, while slavish cloning (profiting from the creative expression of others) may be punished. Functional expressions, like the phrase "file" or the backslash key, are not protectable. Aesthetic expressions, like choosing the order of the commands in a menu, are protectable. If a software firm wishes to chance cloning a protected expression instead of creating one of their own, the cloning firm must now factor possible litigation costs into their product equation.

Obviously, only time will tell the final effect of the *Lotus* decision on the software industry. Other suits, still pending, will help to define the outer reaches of protection and infringement. Software litigation is still very young and uncharted; while *Lotus* may always be seen as a large island in the

---

also, perhaps showing a trend against attempted use of copyright protection and toward patent protection. *See generally* Parker, *Microsoft as Industry Bully May be an Unearned Reputation*, INFOWORLD 46 (Dec. 10, 1990).

While this survey is obviously limited in scope and only as accurate as the reporters covering the issue, it is useful to view the overall trend of the year of 1990. There was no major onslaught of litigation in the post-*Lotus* months. It is, of course, impossible to measure if others are still planning litigation, or awaiting resolution of other pending software suits, such as *Lotus-Mosaic* or *Microsoft-Apple*.

[183] The confusion has generally been on the industry side. *See* Pane, *supra* note 46, at 44 ("In his decision, Judge Keeton said that Lotus' user interface—which he defined as the menus and keystrokes—are protected by copyright, and that VP-Planner [sic] infringed that copyright."); Darrow, Lotus *Litigation Sparks Corporate Resentment; Suit Viewed as a Way to Gain Market Share*, INFOWORLD 46, 46 (Sept. 3, 1990) ("[C]orporate customers express worry that Lotus might attempt to extend its copyright to keystroke sequences, and thus to customer-written macros.").

[184] Several commentators seem to have hit the nail on the head, but they appear to be in the minority. *See* Zachmann, *supra* note 180, at 10 ("[The ruling] is neither a triumph for 'look and feel' nor a disaster for software innovation . . . this decision is on sufficiently narrow grounds that it isn't likely to have a very dramatic impact on the industry. . . . The ruling . . . sets a clear precedent only in forbidding the direct copy of an existing product as a whole. It goes no further in that respect than have other legal precedents."); Lotus *Decision Extends Copyright Law; Menu Structure and User Interface Are Protectable, Judge Rules*, INFOWORLD 85, 85 (July 9, 1990) ("it boils down to the fact that the menu structure is copyrightable . . . .").

swirling sea of rules and holdings, many rules and guidelines have yet to been made.

VI. A PROPOSED METHOD FOR EVALUATING THE COPYRIGHTABILITY OF "LOOK AND FEEL"

Generally speaking, there are three ways to proceed from the status quo of software copyright protection: first, the status quo can be maintained, progressing as in the past by judicial decisions; second, legislative change can be enacted; and third, present copyright analysis can be molded to more properly fit computer software cases. This Paper will address all three options, concluding that the third option is most beneficial.

The first option is actually to do nothing; specifically, to apply present-day copyright law to computer software applications, adding to the growing body of cases that have interpreted the law and the legislative intents of CONTU. Several commentators favor this solution, claiming that "[m]any in industry and in the legal profession take the position that existing structures like copyright and/or patent are adequate to deal with software and that *sui generis* approaches risk obsolescence . . . ."[185]

Some fear that courts will continue to "struggle, as they have in the past, to protect authors' creativity while preserving competition in the marketplace."[186] Yet others fear "that even a correct application of copyright law leads to anticompetitive results."[187] All commentators have agreed that basic copyright law was not designed to accommodate computer software protection. The past ten years of cases have shown an increasing divergence in case holdings, with circuit courts of appeals disagreeing with each other at one level and district courts disagreeing on others. Perhaps the greatest difficulty with the status quo is the signal, or lack thereof, that is being sent to the computer industry. The absence of a bright-line test leads to a lack of predictability. Statements of antimonopoly power and holdings of copyrightable user interfaces further confuse the software developer. When such confusion reigns, the call usually sounds for legislation.

Perhaps the easiest method to plan, but the most difficult to implement, would be legislative change directed at computer software protection. The call for legislation has already gone out.[188] Many consider that any

---

[185] *Staff Paper*, *supra* note 30, at 15.

[186] Russo & Derwin, *supra* note 61, at 11.

[187] Comment, *supra* note 2, at 1006.

[188] "If *Lotus-Paperback* is affirmed and/or followed . . . we do need legislation in this area. . . . Such a development would mean that . . . copyright protection for software . . . is heading off in an untoward direction." Abramson, *supra* note 149, at 9.

legislation would be better than the status quo.[189] Many commentators have discussed varying types of legislation,[190] from compulsory licensing (akin to records) to ultra-thin or ultra-wide definitions of software "expression." While legislative plans are easy to devise, history has shown that they are difficult to agree on, difficult to apply in grey areas, and are inflexible to a changing industry.

As to the first problem noted, just what would the agreement on protection be? The industry and commentators seem evenly divided[191] between heavy and light protection, both sides with viable policy arguments. Who would prevail? Secondly, how could any statute differentiate the "grey" areas of law? With over 200 years of precedent, the greatest minds of American legal history still haven't been able to define a bright-line test to determine idea from expression. Would computer legislation be any easier to devise? Finally, the rapidly changing nature of the software industry[192] could possibly make any legislation outdated before it is passed into law (much like CONTU). In sum, legislation may be more trouble than it is worth.

The third option is a modification of the interpretation of present-day copyright law. Granting that such modifications can never be a panacea, this author submits that this option is the lesser of all evils.

Looking at the problem objectively, the key seems to be finding a balance, within copyright law, between protecting an author's creative work, providing an incentive to innovate, and protecting learning costs of users by promoting standardization. Unfortunately, these are contradictory objectives that are mutually exclusive. In order to avoid the violation of any one objective, any proposed solution would most likely have to blend all three objectives in order to achieve a compromise. Any such compromise must also blend these specific "look and feel" objectives into the standard body of copyright law without infringing upon the well-established objectives of copyright.

This author proposes a modification to the test for infringement; namely, an exception (or added factor) to the definition or determination of substantial similarity. I propose that a purported infringing program be considered not

---

[189] "[O]thers consider that modifications to existing structures, or development of *sui generis* protection, are preferable to forcing software to fit models more suitable to other types of works and discoveries." *Staff Paper*, *supra* note 30, at 15.

[190] *See generally* Abramson, *supra* note 149.

[191] *See supra* notes 176-85 and accompanying text.

[192] "Another problem in determining where software fits in the intellectual-property system is that computer software and hardware technologies are changing rapidly, both qualitatively and quantitatively." *Staff Paper*, *supra* note 30, at 13.

substantially similar (and thus not infringing), if, in addition to present tests, *the second program offers a substantially significant benefit that the original does not offer, and the use of the copyrighted expressions of the original program are required.* More simply stated, punish those who merely copy, but do not punish those who copy the "look and feel" but add something significantly improved to the program.

The idea of rewarding innovation while punishing slavish copying is not new. Several commentators have suggested finding infringements only in cases of close copying.[193] Similarly, others are hesitant to find infringement when innovation or effort are present.[194]

In fact, finding infringement only with minimal protection, i.e., protection against literal copying only, is not new at all to copyright, being a well-established principle in some areas of copyright law. For example, in the seminal case of *Continental Causalty Co. v. Beardsley*,[195] the Second Circuit Court of Appeals held that insurance forms and instruments with only small degrees of variation available are copyrightable, but infringed upon only when literal copying has occurred.[196] In *Morrissey v. Procter & Gamble*,[197] the First Circuit Court of Appeals applied minimal protection to another factual setting, namely the instructions to a game.[198] A modification of the substantial similarity doctrine would follow along these lines.

Many commentators have noted that the present system of protection does not really work for computer software.[199] Others have noted that present-day principles will eventually have to adapt to new technology.[200] The question then becomes *where* copyright law should be modified, attempting

---

[193] "Our conclusion is that copyright should be extended to protect against close copying . . . ." Russo & Derwin, *supra* note 61, at 11. "[C]ourts should . . . require fairly extensive literal similarity before considering such claims." Yen, *supra* note 8, at 434.

[194] "[C]ourts should be sensitive to these factors and should be more hesitant to find infringement where the second comer's development efforts involve substantial investments of time, money, and energy." *Computer Software and Copyright Protection: The "Structure, Sequence, and Organization" and "Look and Feel" Questions*, *supra* note 43, at 10.

[195] 253 F.2d 702 (2d Cir. 1958).

[196] *Id.* at 706.

[197] 379 F.2d 675 (1st Cir. 1967).

[198] *Id.* at 678.

[199] "Software does not fit comfortably into the traditional intellectual-property frameworks of copyright (which protects writings) or patent (which protects processes and machines)." *Staff Paper*, *supra* note 30, at 13.

[200] "[T]he conferees are in agreement that courts will have to adapt traditional copyright principles to a new and different technology." *LaST Frontier Conference Report on Copyright Protection of Computer Software*, *supra* note 44, at 17.

to gain the most benefit with the smallest abrogation of existing law. The test for infringement seems a perfect place to modify. Unlike the idea/expression dichotomy and functionality, the determination of substantial similarity is not defined in, nor restrained by, statute, and the area of law itself is presently unsettled.[201] The test for substantial similarity itself is generally considered a grey area.[202]

The proposed modification can be implemented as either an exception to the substantially similar test[203] or as an added factor in the determination of substantial similarity.[204] For discussion, the proposed modification can be broken down into two general parts: substantially significant benefit and required use.

The first part, the requirement that, to avoid infringement, a secondary program must offer a substantially significant benefit that the original program does not offer, is an attempt to encourage innovation by building on another's shoulders while disallowing noninnovative cloning. Recognizing the cost/benefit analysis that a user generally goes through in deciding what software to purchase,[205] this requirement depends on a correct determination of what a "substantially significant benefit" is.

---

[201] "[I]t is not settled in software copyright law what procedure should be used for determining when infringement has occurred in software cases and, more importantly, what the test for infringement should be." Samuelson, *supra* note 44, at 68.

[202] "Somewhere between the one extreme of no similarity and the other of complete and literal similarity lies the line marking off the boundaries of 'substantial similarity.'" Russo & Derwin, *supra* note 61, at 5 (quoting 3 NIMMER ON COPYRIGHT, § 13.03[A]).

[203] As an exception to the substantially similar test, the proposed modification would be actuated after copyright protection has been established and substantial similarity has been found using standard rules. Only at this time would the decision-maker determine if the proposed modification exempted the purported-infringer from infringement.

[204] Unlike an exception, here the proposed modification would be actuated after copyright has been established but *during* the determination of substantial similarity; the proposed modification could be used as a factor of any weight (greater or lesser) in persuading the decision-maker's determination of substantial similarity.

[205] In general, a software purchaser is either making an initial purchase of software or looking to upgrade a present system. If the software is an initial purchase in an application area, the purchaser must recognize that there will be significant start-up and training costs for whichever package is purchased. At this point all software packages start out equal.

With computers becoming more integrated into modern-day businesses (especially spreadsheets like Lotus 1-2-3), a purchaser is more likely looking to upgrade software. At this point the purchaser must be aware of training and start-up costs already spent on present-day software that would have to be re-invested if purchasing a new package which would not allow transferability of training. At this point, software switching costs become a disincentive towards switching software packages. A competing software package would have to have benefits that would outweigh all switchover costs in order to justify switching packages.

While not a bright-line test, the determination of a substantially significant benefit must still have guidelines. A substantially significant benefit must be an element of the software that, everything else being equal,[206] would be considered a significant factor in a user's decision as to which software package to purchase. In other words, if both packages were sitting next to each other on a shelf, a user-purchaser would consider the innovative difference of the secondary program as significant enough to merit the purchase of the secondary program over the original.

This economic interpretation of substantially significant benefit has many advantages. First, it protects a creative work from mere cloning, without over-protecting the work by giving it a monopoly over competitors. The creator would be able to benefit from the work (reaping the benefits of creation) while being given an incentive to further his work through innovation. There would be no disincentive to author innovation (as may be the case of stagnation due to monopoly denying competition); rather, the original author would be required to compete with other developers to improve upon the initial work.[207] Second, innovation is encouraged without the fear of loss due to a monopoly (i.e., loss due to a lack of competition).[208] Since a competing developer could capitalize on an author's work *only* by improving it, both competitors and the author would be encouraged to innovate and improve the original work—a much better proposition for society than mere cloning by competitors or monopolistic redundancy from an over-protected author. Third, and finally, initial user learning costs would be protected by standardization of the original "look and feel." While such standardization will help users by decreasing training time, the incentive to innovate should also keep a program's "look and feel" from stagnating. Developers would be encouraged to add on to a user's basic training without necessitating any retraining.

Several other boundaries of substantial significant benefit must be discussed. First, since the substantially similar test is a test for infringement,

---

[206] That is, if there are no software switchover costs or other costs that would make one package enter a cost/benefit comparison with a pre-determined cost.

[207] This is not a new concept in the software industry. Many manufacturers, in an attempt to keep customers from switching to competing packages, continually upgrade their original software packages. Such upgrades are termed "new versions" and generally sold to old customers at a significant discount. This allows the initial author to keep a competitive edge, since others may be trying to copy an older version while the author is innovating a new version.

[208] The fear of litigation due to strong copyright protection over a certain expression of an application that has become a standard may be significant to "chill" other developers and innovators; to use the standard would invite litigation, and to develop a new standard would be too costly in time and effort and too risky in terms of enticing software package switching.

not for copyrightability, a substantially similar benefit may be either functional (not copyrightable itself) or nonfunctional (aesthetic and subject to copyright). Second, a pure economic benefit (i.e., a lower price) on its own would not be considered a substantially significant benefit.[209] Third, and finally, secondary programs that are not direct competitors (such as programs produced for noncompatible machines) would be treated as if they were available as direct competitors for the purposes of a substantially significant benefit.[210]

The second half of the proposed modification is an attempt to pre-empt abuse of the modification by necessitating that the copying of expressions is "required." The interpretation of the term "required" is intended to be loose; only a showing of direct application competition,[211] or significant user training required for the original, is needed for the term to be satisfied. The purpose of this second half is to disallow a developer from placing a token competing application in a program merely to justify the cloning of a "look and feel." Such circumvention would deny society the benefits of innovation from competition in the application arena.

While the proposed modification to the substantial similarity test may not be a panacea, and may have built-in complications (such as the lack of a bright-line test), it still may be the least evil of the three available courses of

---

[209] The intent of this provision is to prevent a second developer from cloning the "look and feel" of a program and being able to market the program for a lower cost (because the second developer did not have to swallow any development costs). A mere cost savings is not, by itself, a substantially significant benefit in light of the proposed modification.

[210] For an example of the use of the proposed modification, consider the modification in the light of *Lotus v. Paperback.* Since copyright protection was established and substantial similarity found as a matter of law, the proposed modification could only act as an exception to infringement. The test would be whether VP Planner offered a substantially significant benefit over Lotus 1-2-3 which, minus pure software price considerations, would justify a purchaser choosing VP Planner over Lotus 1-2-3. Since, as Judge Keeton notes, the differences between the two spreadsheets are trivial, VP Planner would not evoke an exception, and would be considered an infringer.

The results may be different, however, when Lotus is compared to a program such as Borland's Quattro Pro. Under that prospective case, the decision-maker would have to determine if Quattro Pro contained a substantially significant benefit over Lotus 1-2-3 to justify purchase of Quattro Pro over Lotus 1-2-3. If empirical figures supporting Quattro Pro's popularity are any indicator of a substantially significant benefit, Quattro Pro would be much more likely to evoke an exception than VP Planner. *See generally* Picarille, *supra* note 5 ("Borland is Lotus' most formidable opponent to date . . . . [Quattro Pro] accounted for 24 percent of the 2.1 million spreadsheet programs sold last year, while 1-2-3 held 58 percent of the market . . . .").

[211] That is, the application programs in contention must be of the same general type, such as spreadsheets, wordprocessors, or, more generally, business programs.

action. Even if the proposed modification itself gains no support, other manipulations of present-day copyright law may still be more advantageous than making no changes or drafting legislation. Regardless which course of action is chosen for software copyright, perhaps the most important part of any available course of action is to keep in mind both software and copyright policy objectives and goals.

VII. CONCLUSION

In *Lotus*, Judge Robert Keeton held that one specific element of the user interface of the spread sheet Lotus 1-2-3, the menu command structure, was copyrightable and infringed by defendant Paperback Software. While being hailed as a decision about the copyrightability of the "look and feel" of a program, the *Lotus* holding is actually a very narrow decision that incrementally follows the present prosoftware protection trend of the courts. The decision itself has been greatly misunderstood, the misunderstanding of which may have led to its immense unpopularity. Furthermore, the destruction of the industry predicted by *Lotus* followers has arguably not occurred.

The magnitude of the interest in the decision may eventually lead to legislative change, but this Paper suggests instead that a slight modification of the substantial similarity test, rewarding innovators and punishing slavish copiers, may be more beneficial.

Brian Johnson