

*Comment**A Behavior-Based Model For Determining Software Copyright Infringement**Dennis M. Carleton* †**TABLE OF CONTENTS**

- I. Introduction 405
- II. Defining Computer Program Behavior 408
- III. Fitting Computer Programs Within Traditional Copyright Law 409
 - A. Behavior Is Expression 409
 - B. Why Behavior Is Protectable 411
 - C. Limitations On Protection Of Behavior 416
 - D. Computer Programs As Useful Articles 418
- IV. Relevant Case Law 420
- V. The Proposed Behavior-Based Test 425
- VI. An application of the Proposed Behavior-Based Test 430
- VII. Conclusion 432

I. Introduction

In 1980, Congress amended the Copyright Act to explicitly recognize copyright protection for computer programs.¹ However, Congress left the task of determining the proper scope of such protection to the courts, providing only that the courts maintain the traditional distinction between idea and expression.²

Since then, courts have grappled with copyright protection for computer programs with mixed results.³ The primary struggle surrounds the extent of copyright protection for "non-literal" aspects of computer programs.⁴ This article suggests that program behavior⁵ must be protected as a non-literal aspect of the program. Furthermore, any test for copyright infringement, absent readily provable literal copying of the programmer's source code, should look solely to the behavioral aspects of the program to determine whether infringement has occurred, eschewing any analysis that dissects a program's structure, organization and other elements related to the programming code.

There are several reasons for advocating this "black box"⁶ type of analysis. First, computer programs are included under the Copyright Act as literary works. As such, they should receive the same level of protection that is extended to other literary works. Thus, the traditional copyright principles and doctrines which protect the non-literal aspects of other literary works should also protect the non-literal aspects of the programmer's expression. Second, the underlying purpose of the Copyright Act supports the protection of program behavior. The behavior of a program is the only aspect of the programmer's expression that is perceived and valued by users. Unless the programmer receives protection from copying of program behavior, the programmer's incentive for producing creative works will be diminished. Finally, there are many ways to write and structure a program so that it will behave in a particular way. As a

result, any test for non-literal infringement limited to dissecting the programming code, organization and structure will fail to detect copying of the original work, thus undermining the protection offered by copyright.

To date, the courts have been uneven in their treatment of copyright protection for program behavior. In many cases, courts have extended copyright protection to behavioral elements of computer programs, accepting the notion that computer programs ought to receive the same degree of protection from non-literal copying as provided to other literary works.⁷ Other courts, however, have held that program behavior ought to be excluded from the inquiry into non-literal infringement, and have limited their analyses to the non-literal elements of the source code.⁸

The first part of this article develops the reasons for advocating a behavior-based test for determining software copyright infringement, starting with an analysis of the problem in light of traditional copyright doctrine, the Copyright Act of 1976 and congressional intent. Next, the article analyzes how the courts have struggled with the concept of non-literal infringement of computer programs. Most courts appear to be moving toward the Abstraction-Filtration-Comparison (AFC) test which was formulated by the Second Circuit in *Computer Associates International v. Altai*.⁹ While most courts agree on the test, they disagree over its proper application. This article suggests a behavior-based test which modifies the Abstraction-Filtration-Comparison test. The test proposed by this article would eliminate the "abstraction" step of the AFC test, which involves separating the levels of abstraction within the program's code, organization and structure. Instead, the behavior-based test would replace the abstraction step with an "identification" step, which dissects and identifies the elements of the program's behavior. The behavior-based test would eliminate much of the confusion and difficulty involved with applying the AFC test. Finally, the article applies the proposed test to the facts in *Lotus Development Corp. v. Borland International* to illustrate its advantages. Developing a workable test for non-literal infringement of computer software has become especially relevant in light of the Supreme Court's recent grant of certiorari in the *Lotus* case.¹⁰

II. Defining Computer Program Behavior

Before beginning any discussion of a test for copyright infringement based on the behavior of computer programs, it will first be necessary to define "program behavior." For purposes of this article, a program's behavior consists of the appearance of the user interface, the way in which the user interacts with the program, the manner in which information is input to and output from the program, and the ensemble of functions provided by the program. In short, it is everything that happens once the program is executed on the target system.

The behavior of a program may be quantified in terms of discrete elements. These elements include the discrete functions performed by the program, specific features of the program's user interface, or particular program responses to a user's actions. A program's behavior can thus be summarized as the set containing all the discrete behavioral elements. Some simple examples of discrete behavioral elements include the manner in which the program responds to specific inputs, such as the invocation of a command by a user, or a signal from the printer indicating that it has run out of paper. A program's response may be expressed as a change in the appearance of the screen, the posting of error messages, or the emission of a beep or other audible signal. Behavior also includes a program's expression in the absence of any input, such as a screen saver program which posts a fluctuating pattern on the computer's screen when the user does nothing. Many programs have very limited or almost no interaction with users, yet still exhibit behavior. An example of this would be a program that controls traffic lights.

III. Fitting Computer Programs Within Traditional Copyright Law

A. Behavior Is Expression

In order to be copyrightable, the subject matter must be an "original work of authorship fixed in any tangible medium of expression."¹¹ Copyright law protects an author's expression, rather than merely the idea being expressed. The expression of an idea takes on both substance and form. The substance of the expression lies in the distinction between the author's expression and the ideas which are being expressed. This is the essence of the traditional idea/expression dichotomy of copyright law.¹²

In most cases, the form of the author's expression is readily apparent. For example, the expression of the author of a novel is manifested in the form of printed words on the page. The form of the author's expression, however, can have multiple manifestations, raising the question as to the extent of copyright protection for each distinct manifestation.

The form of the computer programmer's expression can be divided into two distinct manifestations: the literal and the behavioral.¹³ The literal manifestation consists of the text of the human-readable program as written by the programmer (i.e., the source code).¹⁴ The behavioral manifestation consists of the dynamic form of the program as it is being operated on a computer.¹⁵ This bifurcation of

expression is neither a new concept nor one unique to computer programs.¹⁶ Consider the various forms of expression of a musical work. The sheet music embodies the literal manifestation of the composer's expression, while the performance of the musical piece makes up the "behavioral" manifestation of that same musical work. Both manifestations are merely alternative forms of the same expression. Thus, each manifestation of the expression should be protected by copyright, because both are a part of the same "original work of authorship." In the case of the musical composition, no one doubts the conclusion that both forms of expression are protected by copyright. The unauthorized public performance of a musical composition infringes the composer's copyright, as does the copying of the sheet music.¹⁷ By analogy, the same reasoning should hold true for computer programs—copying of program behavior infringes the programmer's copyright, as does the copying of the source code.

Computer programs, however, differ from most other literary works in that they derive most of their value from the behavioral form and not from the specific manner in which the programmer implemented the behavior.¹⁸ The behavior of a program may provide a solution to a problem, offer entertainment, or serve as a tool which the user can employ to do valuable work. Consumers would not want to buy a program that does not "behave, i.e., that [does] nothing."¹⁹ Consumers care only about the benefit derived from how the program behaves; the literal manner in which the programmer has implemented the behavior is of little consequence to the consumer.

The literal manifestation of the programmer's expression, on the other hand, only has value to the programmer, who can modify it to maintain and enhance the behavior in subsequent versions of the program.²⁰ Consumers value a program "not because they have any intrinsic interest in what its text says, but because they value what it does and how well it does it."²¹ When a consumer buys a program, he buys only the program's behavior, not the text of the program's source code.²² The consumer cannot directly perceive the programmer's expression from the literal text of the program, because it is invisible to him. Thus, the program behavior represents to the consumer the totality of the programmer's expression.

Given that programs derive so much of their value from their behavioral manifestation, and given that consumers do not perceive the literal manifestation of the program, the emphasis of copyright protection should be aimed at program behavior rather than the written source code.

B. Why Behavior Is Protectable

1. The 1980 Changes to the Copyright Statute

When computer programs were explicitly added to the copyright statute in 1980, Congress expressed its intent that such works should be treated as literary works.²³ Though the Copyright Act's definition of a literary works does not expressly list computer programs, the definition clearly encompasses a typical computer program.²⁴ The definition of literary works includes "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."²⁵ In addition, a 1976 congressional report explicitly discussed treating computer programs as literary works "to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves."²⁶

Some confusion over the protectability of program behavior may stem from section 101 of the Copyright Act, which defines a computer program as "a set of statements or instructions used directly or indirectly in a computer to bring about a certain result."²⁷ A strict reading of section 101 might suggest that a computer program for copyright purposes is limited to the written source code because the definition under the Act does not expressly refer to a program's behavior or dynamic structure. Yet, the absence of a direct reference to program behavior under section 101 should not be determinative. The language of section 101 can be interpreted as referring to program behavior indirectly through the phrase "to bring about a certain result."

Furthermore, the use of the defined term "computer program" is conspicuously absent in the section of the Copyright Act defining protectable subject matter, implying that Congress did not intend for computer programs to be treated any differently than other types of literary works. Had Congress intended to limit copyright protection for computer programs as compared to other copyrightable works, it could easily have done so by creating a separate category for computer programs under section 102.²⁸ However, no such attempt was made. The term "computer program" is in fact only used in the copyright statute in section 117,²⁹ which exempts RAM copies created in the owner's operation of a program and copies made for backup purposes from being deemed infringing of the programmer's copyright.³⁰

2. Quasi-Legislative History

Many commentators consider the National Commission of New Technological Uses of Copyrighted Works, Final Report,³¹ (the

CONTU Report) to be the quasi-legislative history of the 1980 amendments to the Copyright Act.³² Although not binding, many courts have accepted the report as evidence of congressional intent because the recommendations contained in the CONTU Report were adopted by Congress without alteration.³³ At the time that CONTU prepared its report, "it had been well-established for decades that copyright protection (for literary works) extends to non-literal copying as well as to literal copying."³⁴ Thus, had CONTU intended to apply a different standard for determining non-literal infringement for computer programs than for other forms of literary works, the commission could have recommended this in its report. Yet, the CONTU Report recommended that programs be protected as literary works, without any qualification.³⁵ The report expressed the belief that existing copyright principles were adequate for the task of protecting computer programs, suggesting that courts were better equipped than Congress to develop the law through case-by-case decisions, rather than by shortsighted legislative fiat.³⁶

The CONTU commissioners recognized the problem of distinguishing between idea and expression when dealing with a computer program:

Drawing the line between the copyrightable form of a program and the uncopyrightable process which it implements is simple [for pure literal copying]. But the many ways in which programs are . . . used . . . and the new applications which advancing technology will supply may make drawing the line of demarcation more and more difficult. To attempt to establish such a line in this report written in 1978 would be futile. Most infringements, at least in the near future, are likely to involve simply copying . . . Should a line need to be drawn to exclude certain manifestations of programs from copyright, that line should be drawn on a case-by-case basis by the institution designed to make fine distinctions—the federal judiciary.³⁷

The commission's refusal to set a standard has given the courts wide latitude for deciding where the line should be drawn. The fact that commissioners recognized the difficulty in drawing such a line suggests their awareness of the distinction between the literal and behavioral manifestations of computer programs. Had the commissioners wished to exclude the behavior of computer programs from the scope of copyright protection, they would have explicitly done so in their report. Their silence on this point implies the intent to include behavioral aspects of computer programs within the scope of copyright protection. Statements by CONTU Vice-Chairman Melville Nimmer suggest that the above interpretation of the CONTU Report is correct.³⁸ Nimmer said he understood CONTU as in no way limiting the application of traditional copyright doctrines to computer programs:

CONTU did not recommend, and did not intend, any change in the continuing applicability to programs of general copyright principles—e.g., as to the copyrightability and infringement—in effect following the enactment of the general revision of the Copyright Act of 1976. The general copyright principles applicable to programs have been, and remain, those which are applicable to novels, plays, directories, dictionaries, textbooks, musical works, maps, motion pictures, sound recordings and other categories of works.³⁹

Thus, CONTU apparently intended that computer programs receive the same level of protection from non-literal infringement as other types of literary works.⁴⁰

3. Other Literary Works with Bifurcated Expression

Most traditional literary works receive protection from non-literal infringement. Take, for example, "architectural plans, choreography, musical scores, musical editions, and stage direction."⁴¹ All of these works possess bifurcated manifestations of the author's expression, and all receive some degree of copyright protection from non-literal infringement.

Many literary works, besides computer programs, serve primarily utilitarian purposes. Such works include "dictionaries, code books, encyclopedias, advertising, and 'how to' instruction manuals, that, like many computer programs, have a primarily utilitarian, rather than aesthetic, entertainment, or educational purpose."⁴² Copyright nevertheless protects both the literal manifestations and the non-literal aspects of these works.⁴³ Literary works with utilitarian aspects have been "accorded protection since our first Copyright Act in 1790, which embraced maps and charts."⁴⁴ Thus, a computer program, if treated as a literary work as Congress has mandated, should also be accorded protection of its non-literal aspects, the program's behavior, where such aspects represent the programmer's expression.

4. Purpose of Copyright

Copyright is a tradeoff which grants an author certain exclusive rights to his work in exchange for the disclosure of the work for the public good.⁴⁵ The exclusive rights granted to the author provide the incentive for the author to create. Thus the valuable aspects of the

author's work must be protected from copying; otherwise the author's incentive to create will disappear.

A program which behaves in an identical manner or a very similar manner to another program has the potential of becoming a "market substitute"⁴⁶ for the original program. Such programs may potentially degrade the incentive to the programmer because fewer copies of the original program may be sold in favor of substitute programs. Protection of a program's behavior becomes critical because program behavior plays a significant role in marketing a program to potential purchasers.⁴⁷ The user interface of a program, one element of the program's overall behavior, plays a crucial role in helping consumers to "differentiate a program from among similar software applications in a crowded market."⁴⁸ Taking into consideration that the programmer relies on this aspect of expression to distinguish his program from other competing programs, the necessity of protecting the behavioral manifestation of the programmer's expression from copying becomes readily apparent.

Furthermore, limiting copyright protection to program code fails to provide adequate protection for the programmer's creative expression.⁴⁹ Excluding behavior in the determination of non-literal infringement exposes the program's most valuable attributes to copying by any skilled programmer, who could reverse engineer a program by analyzing its behavior and producing an identical or infringing copy without ever viewing the literal source code.⁵⁰ Such reverse engineering often requires only a fraction of the time and labor invested by the programmer of the original work. Ignoring the behavior of a program thus circumvents the very purpose of copyright protection: providing an incentive for the programmer's creative expression for the benefit of the public.

Unlike most other literary works, computer programs are unique in that there is a significant degree of independence between the literal and behavioral manifestations. For most other literary works, the performative manifestation of the work is directly dependent upon the literal manifestation of the work. For example, there is only one way to represent a musical composition in standard musical notation. Likewise, the performance of a play springs directly from the script. For both musical and dramatic works, a change in the literal manifestation of the work has a concomitant effect on the behavioral manifestation. The same cannot be said for computer programs, because there are often numerous ways to write source code to produce a specific behavioral effect.⁵¹ Thus, limiting copyright infringement of computer programs to copying of the source code leaves the programmer with virtually no protection at all. Infringers will be permitted to copy the original programmer's work, so long as they do not copy the original source code.

C. Limitations On Protection Of Behavior

The scope of the protection for computer programs should be identical to the scope for other types of literary works. First, the dichotomy between idea and expression, fundamental to the determination of copyrightability for all types of literary works, should apply to program behavior. Second, the limiting doctrines of merger, *scènes à faire* and originality apply to behavior just as they do to the non-literal aspects of other literary works.

The Supreme Court opinion in the 1879 case of *Baker v. Selden*⁵² developed the modern concept of a dichotomy between idea and expression for determining copyrightability. The Court held that copyright protection for a book describing an accounting system did not extend to the actual accounting system described by the book.⁵³ The alleged infringer, Baker, had used ruled accounting sheets similar to those that had appeared in Selden's book, but had only changed the column headings and arrangement of columns. The Court concluded that the ruled accounting forms were "necessary incidents"⁵⁴ to the use of Selden's accounting system, which was "open and free to the use of the public."⁵⁵ Copyright protection was limited to Selden's particular description of the accounting system. The Court thus distinguished between the unprotectable idea expressed in Selden's book (the accounting system) and the copyrightable expression (Selden's particular description of the accounting system). The principle of *Baker* has since been codified in section 102(b) of the Copyright Act, which states that copyright protection is unavailable for any "idea, procedure, process, system, method of operation, concept, principle, or discovery."⁵⁶

In order to see how *Baker* applies to a computer program, consider the following example. Take a computer program which performs word processing functions. Assume that this word processing program includes a function which allows the printing of addresses on envelopes, a behavioral or non-literal aspect of the program. Under section 102(b), there can be no doubt that copyright protection does not extend to the "idea" of printing addresses on envelopes. Assume further that in order to print the envelopes, the word processing program must undertake the following process: (1) collect the main and return addresses from the user, (2) collect information regarding the size of the envelope from the user, (3) calculate the spacing of the addresses on the envelope, (4) send the spacing information and address information to the printer, and (5) prompt the user to place the envelope into the printer. Any word processing program that offered the function of printing an envelope would have to go through a similar, if not identical, process. Under *Baker*, therefore, this process would be uncopyrightable because it is necessarily incident to the idea of printing an envelope. Furthermore, this sequence of steps would be expressly excluded from the subject matter copyrightable under section 102(b) because it would be classified as a procedure, process, system or method of operation.

While the idea of printing an envelope and the process used to perform the task are uncopyrightable, the manner in which the program implements the function and process may be considered protectable expression. This would include such things as how the user selects the function, how the program collects necessary information from the user, how the program behaves in case of an error, and so on, subject, of course, to the limiting doctrines of merger, *scénès a faire* and originality. There are multiple ways in which a program could go about performing these tasks. If multiple options for implementation exist, any specific implementation chosen by a programmer cannot be deemed necessarily incident to the idea of the function. Therefore, the particular manner by which the programmer implements the function may be considered the programmer's expression, which falls within the subject matter of copyright. Consider the explanation of this concept in *Baker*:

[T]he teachings of science and the rules and methods of useful art have their end in application and use; and this application and use are what the public derive from the publication of a book which teaches them. But as embodied and taught in a literary composition or book, their essence consists only in their statement. This alone is what is secured by the copyright. The use by another of the same methods of statement, whether words or illustrations, in a book published for teaching the art, would undoubtedly be an infringement of the copyright.⁵⁷

In the example of the envelope-addressing function, the process of printing an envelope is the useful art. The particular manner by which a program implements the function is the "statement" of this useful art, and the "use of another of the same methods of statement" (i.e., an expressive implementation that used the same or substantially similar methods) would be an infringement.⁵⁸

D. Computer Programs As Useful Articles

Objections to the protection of the behavioral aspects of computer programs often focus on the useful article doctrine because computer programs provide utility to their users.⁵⁹ The useful article doctrine of copyright law, articulated by the Supreme Court in *Mazer v. Stein*⁶⁰ and codified in the definition of "[p]ictorial, graphic, and sculptural work" in the current Copyright Act, states,

such works shall include works of artistic craftsmanship insofar as their form but not their mechanical or utilitarian aspects are concerned. The design of a useful article . . . shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates . . . features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article.⁶¹

There are two responses to those who object to copyright protection for computer programs on the grounds that the programs are "useful articles." First, any test for copyright infringement of a computer program which applies the limiting doctrines of traditional copyright law will filter out the purely functional aspects of the program. Only particular *implementations* of functions would be protected. Thus, fears that copyright protection for computer programs will give programmers exclusive rights to certain useful arts are unfounded.

Second, the useful article doctrine, as incorporated into the Copyright Act, does not apply to literary works, which include computer programs. The useful article doctrine often has been erroneously generalized to encompass all objects which are the subjects of copyright protection. A strict reading of the copyright statute, however, indicates that the doctrine is intended only to be applicable to pictorial, graphic, or sculptural works.⁶² The definition of these types of works, for copyright purposes, includes only those aspects of the objects which are not utilitarian, but which represent the form, or creative expression, of the creator. The categorization of a computer program as a literary work would thus seem to preclude its definition as a useful article.⁶³

Additionally, while there is no doubt that mere functions are too close to ideas to be copyrightable,⁶⁴ protection is available for sets of functions as compilations. Section 103 of the Copyright Act provides protection for compilations to the extent of the expression contributed by the author,⁶⁵ even if the material that is compiled is not in and of itself copyrightable.⁶⁶ Thus, if a program provides a very unique set of functions, such that the selection of that particular set of functions represents creative expression by the programmer and is not dictated by functional considerations or constraints, then that set of functions is copyrightable. Likewise, protection should be available for compilations of behavior that implement those functions, above and beyond their inherent copyrightability. The court in the case of *Whelan Associates v. Jaslow Dental Laboratory*⁶⁷ supported this notion, noting that sequencing and ordering of materials would also be covered by copyright as compilations, "i.e., that the sequence and order could be parts of the expression, not the idea, of a work."⁶⁸

To date, few courts have used this concept to protect the behavior of programs from copying. The compilation of discrete functions or discrete elements of behavior into a single product that is useful in some way describes the essence of the creative design task of a programmer—the very thing that copyright law is meant to protect.

IV. Relevant Case Law

The courts have been uneven in their treatment of program behavior. Several courts have recognized copyright protection for some behavioral aspects of computer programs, including such elements as screen displays⁶⁹ and user interfaces.⁷⁰ Other courts have extended non-literal protection to the structure, sequence and organization of a program.⁷¹ Other courts, however, have found program behavior to fall outside the subject matter of copyright.⁷² This lack of consensus among the courts has led to confusion over the protectability of program behavior.

The courts that have upheld copyright protection for behavioral elements of programs have taken notice of the fact that behavior represents the programmer's expression. These courts have realized that behavioral elements are "as valuable, if not more valuable, than the program code and structure itself and, therefore, warrant protection."⁷³ For example, in *Lotus Development Corp. v. Paperback Software International*,⁷⁴ the court expressed concern that if the scope of protection for computer programs is too narrow, "copyright law never would, as a practical matter, provide computer programs with protection as substantial as Congress has mandated-protection designed to extend to original elements of expression *however embodied*."⁷⁵

In *Whelan Associates v. Jaslow Dental Laboratory*,⁷⁶ the Third Circuit took an important first step toward recognition of the behavioral aspects of programs. The court noted that other literary works could be infringed through non-literal copying, citing the example that "[o]ne can violate the copyright of a play or book by copying its plot or plot devices."⁷⁷ The court then drew the conclusion by analogy that the copyright of a program could be infringed "even absent copying of the literal elements of the program."⁷⁸ The court made its analysis within the framework of the traditional idea/expression dichotomy of *Baker* and section 102 (b) of the Copyright Act, deciding that the idea expressed by the programmer was the function of the program, with the actual program being the programmer's expression of that idea.⁷⁹ The court reasoned that since there were a variety of ways in which the program could have been implemented, "the [dynamic] structure is not a necessary incident to that idea,"⁸⁰ and was therefore copyrightable expression. The court also relied on public policy to reach this conclusion, noting that "we must remember that the purpose of the copyright law is to create the most efficient and productive balance between protection (incentive) and dissemination of information, to promote learning, culture and development."⁸¹

While the *Whelan* court has been widely criticized for providing over-broad protection for programs, the court nonetheless correctly concluded that the non-literal aspects of a program are protectable under copyright.⁸² *Whelan* provides the basis for an argument that non-literal aspects of a program-in other words, the program behavior-are copyrightable.

Though the courts continue to disagree over the proper scope of copyright protection for computer programs, most courts appear to agree upon the Abstraction-Filtration-Comparison (AFC) test, formulated in *Computer Associates International v. Altai, Inc.*,⁸³ as the proper test for non-literal infringement of computer programs. The Second Circuit's AFC test has been adopted in some form by the Fifth Circuit,⁸⁴ the Ninth Circuit,⁸⁵ the Tenth Circuit,⁸⁶ the Federal Circuit,⁸⁷ and by district courts in the Eleventh Circuit.⁸⁸

The district court in *Altai* expressly took notice of the bifurcated nature of the programmer's expression in computer programs:

Each view-textual and behavioral-has its own structure, sequence, and organization. In the standard jargon of programmers, there is static structure, which refers to the program-as-text view, and dynamic structure, which refers to the program-as-behavior view. The static structure and dynamic structure of a program can be quite different; indeed from dealing with the behavior of a program, i.e., operating it, one can tell virtually nothing about its text. Thus . . . "it makes no technical sense to talk simply about the 'structure' of a program, because the term is ambiguous and the distinction [between dynamic structure and static structure] matters."⁸⁹

The *Altai* court criticized *Whelan* as being "inadequate and inaccurate" for ignoring the distinction between the static and dynamic structure of a program, and for assuming that a program is the expression of a single idea, instead of the expression of multiple ideas.⁹⁰

The *Altai* court, however, wrongly expressed doubts as to the copyrightability of a program's "dynamic structure"-in other words, the behavioral manifestation of the program. The *Altai* court's misgivings over the protectability of program behavior stemmed from its misinterpretation of section 102(b) of the Copyright Act.⁹¹ The district court stated that "since the behavior aspect of a computer program falls within the statutory terms 'process,' 'system,' and 'method of operation,' it may be excluded by statute from copyright protection."⁹² Yet, the court stopped short of holding as a matter of law that program behavior was a system, process or method of operation, thereby leaving the door open for copyright protection of program behavior. The *Altai* court held that the plaintiff's rights were fully protected by examining the literal program code, thus permitting the court to avoid determining whether program behavior

should be excluded from the subject matter of copyright.⁹³ Thus the district court left the issue of the protectability of program behavior for another court, or for Congress, to decide.

In affirming the district court in *Altai*, the Second Circuit panel devised the three-step AFC test⁹⁴ for determining infringement of computer programs based on the abstraction test devised by Judge Learned Hand in the case of *Nichols v. Universal Pictures*,⁹⁵ which involved a non-literal infringement claim by an author of a play against the producer of another play. The AFC test applies concepts of traditional copyright law such as the idea/expression dichotomy and the limiting doctrines of merger, *scènes à faire* and originality to computer programs. As such, it is seen by many as a narrowing of the protection for non-literal aspects of computer programs after *Whelan*.⁹⁶ Nonetheless, *Altai* affirms that traditional copyright law principles are applicable to the non-literal aspects of computer programs. *Altai's* AFC test also provides an appropriate framework upon which a test for infringement can be built. The behavior-based test proposed in this article is a modification of the AFC test.⁹⁷

Since *Altai*, several other courts have upheld protection for non-literal behavioral aspects of computer programs.⁹⁸ In the case of *Autoskill v. National Educational Support Systems*,⁹⁹ the Tenth Circuit applied the *Altai* test and determined that a program implementing a reading system was infringed by another program which simply changed the names and sequences of tests in the operation of the infringing program. In *Mitek Holdings v. Arce Engineering Co.*,¹⁰⁰ the district court, in applying the *Altai* test, determined that some non-literal, behavioral elements of the allegedly infringed program were protectable.¹⁰¹ The court concluded, however, that there was not enough similarity to make a finding of infringement.¹⁰² Though the court found no infringement, the significance of *Mitek* lies in the court's recognition that some behavioral elements were found to be protectable.

In *Apple Computer v. Microsoft*,¹⁰³ the Ninth Circuit identified similarities between discrete behavioral elements in both the copyrighted and allegedly infringing programs.¹⁰⁴ The Ninth Circuit, however, held there was no infringement because most of the similarities were permitted by a license agreement between the plaintiff and the defendant, or were obvious expressions of basic ideas, and thus precluded from copyright protection by the merger doctrine.¹⁰⁵ As such, the court felt that infringement could only be found if the works as a whole were virtually identical.¹⁰⁶ This case provides yet another example of an instance where a court has embraced the notion that behavioral elements of a program are protectable, though infringement was not found.

Other courts, however, have followed the dicta in *Altai*, concluding that protection from non-literal infringement does not extend to program behavior. In *Gates Rubber Co. v. Bando American, Inc.*,¹⁰⁷ the district court explicitly held that the behavior of a computer program was protectable:

The court will respectfully disagree with the *Altai* decision and hold that *a program's behavior can be protected by copyright law* [T]he commonality of [the] error denotes "behavior" as to how one part of the program works with another. This is *part of the creative expression* of the program itself.¹⁰⁸

The Tenth Circuit reversed, however, holding that the district court had failed to properly eliminate the behavioral elements of the program from the determination of infringement.¹⁰⁹

The First Circuit decision in the case of *Lotus Development Corp. v. Borland International*¹¹⁰ further confused the issue of protectability of behavior. The First Circuit, in reversing the district court's finding that the menu command structure of the plaintiff's program was protectable, held that menu command structure was a "method of operation" and therefore uncopyrightable under section 102(b).¹¹¹ The court stated that the *Altai* test was applicable to non-literal copying, but that the appropriation of the menu command structure represented literal copying, and thus the *Altai* test did not apply.¹¹² What the court failed to take into account, however, was that the literally copied portion of the program was in reality a non-literal behavioral aspect of the program.

Given the uneven treatment by the courts, the protectability of behavioral aspects of programs remains uncertain. Much of the difficulty lies in agreeing upon the common terminology to describe literal and non-literal aspects of a program. The disagreement, however, appears ripe for a decision by the Supreme Court.¹¹³

V. The Proposed Behavior-Based Test

The test advocated by this article would look solely at the behavioral manifestation of the programmer's expression and would ignore the literal manifestation when determining whether infringement has occurred.¹¹⁴ Once the behavioral elements have been isolated, the proposed behavior-based test would apply the idea/expression dichotomy and traditional copyright limiting doctrines such as merger, *scènes à faire* and originality to determine which behavioral elements deserve copyright protection and which elements ought

to be excluded. The behavioral elements of the original work would then be compared with the behavioral elements of the infringing program to determine whether there is substantial similarity between them. A test of this nature makes more sense in light of the nature and realities of computer programs and the public policy surrounding traditional copyright law.

A good starting point for defining such a test is the AFC test articulated by the Second Circuit in *Computer Associates International, v. Altai, Inc.*¹¹⁵ This test has been adopted, in one form or another, by several courts in cases involving infringement of non-literal aspects of computer programs.¹¹⁶ The *Altai* test consists of a three-step procedure which is used to determine if an allegedly infringing program is substantially similar to the allegedly infringed program.

In ascertaining substantial similarity under this approach, a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernel, or possible kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program. The result of this comparison will determine whether the protectable elements of the programs at issue are substantially similar so as to warrant a finding of infringement.¹¹⁷

1. Abstraction

The abstraction step, described above as "break[ing] down the allegedly infringed program into its constituent structural parts," requires the court to reconstruct the programmer's implementation of the program in reverse order.

Initially, in a manner that resembles reverse engineering on a theoretical plane, a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it. This process begins with the code and ends with an articulation of the program's ultimate function. Along the way, it is necessary essentially to retrace and map each of the designer's steps-in the opposite order in which they were taken during the program's creation.¹¹⁸

This step of the test requires that the court acquire knowledge equivalent to that of a computer program developer, a requirement which is beyond the capability of most courts. As a result, expert analysis and testimony is usually required to assist the court in performing the abstraction step of the test.

The proposed behavior-based test would eliminate the abstraction step of the *Altai* test. The first step of the test would instead involve identifying the discrete elements which represent the program's behavior. Call this the "identification" step. The court should identify the behavioral elements of a computer program simply by operating the program on a computer. This will permit observation of how the program behaves. For example: "the program performs functions X, Y and Z" or "when the user does this, the program responds by doing that." The level of complexity of the observations is a function of the complexity of the program. For a more complicated program, instead of observing "when the user does this, the program does that," it may be necessary to observe that "when the user does this, and conditions A, B and C are met, then the program does that."

As an example, it may be helpful to consider the small icon resembling a trash-can which appears on the lower right hand corner of the Apple Macintosh user interface. When a user uses the mouse to drag a file icon into the trash-can icon, the file icon is prepared for deletion from the computer's disk. This description of the trash-can icon and its function identifies a certain behavioral element of the program which generates the user interface of the Apple Macintosh. Thus, the first step of the behavior-based test has been completed.

2. Filtration

The second step of the *Altai* test, the filtration step, requires the court to apply traditional limiting doctrines of copyright law to the abstractions formulated in the first step of the test. It is described as "examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain." In essence, the court is to apply the doctrines of merger, *scénès a faire* and originality to the various levels of abstraction.

This process entails examining the structural components at each level of abstraction to determine whether their particular inclusion at that level was an "idea" or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea [merger doctrine]; required by factors external to the program itself [*scénès a faire* doctrine]; or taken from the public domain [originality requirement] and hence is nonprotectable expression.¹¹⁹

The purpose of this test is to "filter out" subject matter that is unprotectable. After the application of this step of the test, the fact finder

is left with the protectable expression of the allegedly infringed program.

The *Altai* court first applied the merger doctrine to the abstracted structure of the program. The principle underlying the merger doctrine was well-stated by the First Circuit in the *Concrete Machinery Co. v. Classic Lawn Ornaments, Inc.*¹²⁰ The court stated, "[w]hen there is essentially only one way to express an idea, the idea and its expression are inseparable and copyright is no bar to copying that expression."¹²¹ The *Altai* court also applied this doctrine to computer programs, stating, "[i]n the computer context, this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement."¹²²

The proposed behavior-based test would apply the merger doctrine as stated in *Concrete Machinery* to the behavioral elements of the program which were observed in the identification step of the test. Continuing with the above example of the Apple Macintosh trash-can icon, one can imagine an almost infinite number of ways in which the function which deletes a file from the computer's disk could be implemented. The trash-can icon is therefore not rendered unprotectable by the merger doctrine.

The *Altai* court then applied the "scènes a faire" doctrine. This doctrine states that when external factors constrict the possible ways to express an idea, such expression is not protectable by copyright. Professor Nimmer, in his treatise on copyright law, has identified five instances of external factors which may circumscribe the manner in which a program is designed.¹²³ The *Altai* court adopted these factors.¹²⁴ They are: (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturer's design standards; (4) demands of the industry being serviced; and (5) widely accepted practices within the computer industry.¹²⁵

The proposed behavior-based test adopts these factors as well, but applies them to the observations of the identification step of the test, instead of to the level of structural abstractions identified in the abstractions step of the *Altai* test. In the example of the trash-can icon, it is unlikely that any of the five factors listed above would require that a user interface allow users to delete files from the computer's disk in this fashion.

Finally, the *Altai* court applied the originality standard to the levels of abstraction. This limitation is based on the statutory requirement that copyright protect only "original works of authorship."¹²⁶ In the context of a computer program, this excludes from protection material which is taken from the public domain. The *Altai* court stated, "Such material is free for the taking and cannot be appropriated by a single author even though it is included in a copyrighted work."¹²⁷

The proposed behavior-based test also adopts this standard, as applied to the discrete behavioral elements of the program observed in the identification step of the test. Returning yet again to the trash-can icon example, assuming that Apple is the originator of this particular feature, its copyrightability would be unaffected by the filtration step of the test.

3. Comparison

The final step of the *Altai* test is the comparison step. The court described this step of the test as, "whether the defendant copied any aspect of this protected expression, as well as an assessment of the copied portion's relative importance with respect to the plaintiff's overall program."¹²⁸

The proposed behavior-based test also includes this step of the *Altai* test. The discrete behavioral elements of the alleged infringing program are to be compared with the discrete behavioral elements of the allegedly infringed program which are copyrightable, i.e., those that remain after the filtration step. The comparison should be performed both in terms of quantity and quality of copying. In other words, if the infringing program copied substantial portions of the original program, then a finding of infringement is warranted. Likewise, if only a small portion has been copied but the portion copied is qualitatively substantial, that is, the particular behavior copied makes the original program particularly valuable and desirable to consumers, then a finding of infringement may also be warranted.

VI. An application of the Proposed Behavior-Based Test

To illustrate how the proposed behavior-based "Identification-Filtration-Comparison" test operates, it may be useful to look at its application to a real case. The case of *Lotus Development Corp. v. Borland International*¹²⁹ provides a good opportunity to examine behavioral elements of a computer program.

The dispute in the case arose when Borland copied portions of the Lotus 1-2-3 spreadsheet program in its own spreadsheet product.

Specifically, Borland included in its product a mode which emulated the menu command structure of 1-2-3. The menu command structure is the part of the user interface of the program which allows users to input commands to the program.¹³⁰ Additionally, the 1-2-3 menu command structure also facilitated the running of 1-2-3 user-written macros.¹³¹ The Borland spreadsheet product permitted its users to operate the program in a mode which copied the 1-2-3 menu command structure. This served two purposes. First, it allowed users who were familiar with the 1-2-3 menu command structure to easily switch to the Borland product. Secondly, it allowed users who had written macros for 1-2-3 to run those macros from within the Borland product. The Borland product also featured a "Key Reader" facility which allowed the running of the 1-2-3 macros even when the 1-2-3 menu command structure was not being used.

The district court in the case found that the Borland product infringed the 1-2-3 product. To make this determination, the court used a test which it claimed was compatible with the *Altai* Abstraction-Filtration-Comparison test.¹³² The First Circuit Court of Appeals subsequently overturned this decision, stating that the 1-2-3 menu command structure was a "method of operation" and was thereby uncopyrightable under the copyright statute.¹³³ The Supreme Court has since granted certiorari on the case.¹³⁴

In applying the "Identification-Filtration-Comparison" test outlined above, the first step is to observe the behavior of the program and identify its behavioral elements. There is no doubt that the operation of the menu command hierarchy is a behavioral element, or several behavioral elements of the 1-2-3 program. When a user selects any item in any sub-menu, the 1-2-3 program responds with a specific response, either the posting of a new sub-menu, or the invocation of a program function. In a complete analysis of the case, the court would be required to identify all of the behavioral elements of the program, including the layouts of the screens, the manner in which the functions of the program are carried out, how error messages are posted, etc. For purposes of this example, however, we will stop at the observation that the menu command structure represents a set of discrete behavioral elements of the 1-2-3 program which, if not filtered out by the next step of the test, are copyrightable as the expression of the programmer.

The filtration step of the test determines if any of the behavioral elements observed in the identification step of the test should be rendered unprotectable because of the application of one of the limiting doctrines of copyright law. It does not appear that there is any merger of idea and expression with regard to the menu command structure. The manner in which the Lotus designers implemented the menu command structure of 1-2-3 is certainly not the only way that it could be done. The existence of the alternate method used by the Borland product proves this. Likewise, we will assume that the design of the 1-2-3 menu command structure was original to the Lotus designers.

The application of the *scènes à faire* doctrine, however, raises issues as to the copyrightability of the menu command structure. One of the factors identified by Nimmer and adopted by the *Altai* court was "compatibility requirements of other programs with which a program is designed to operate in conjunction." In this case, the Borland program wishes to allow its users to run the macros they have created using the 1-2-3 program. There is no way to allow the running of the 1-2-3 macros without the menu command structure being recreated in one form or another. The copying of the menu command structure represents a design constraint of making the Borland product compatible with the 1-2-3 program. Thus, the discrete behavioral elements making up the 1-2-3 menu command structure must be filtered out from the set of protectable behavioral elements of the 1-2-3 program.

The application of the comparison step of the test is moot at this point because all of the behavioral elements which have been copied by the alleged infringing Borland program have been filtered out of the set of copyrightable behavioral elements of the 1-2-3 program. Thus, there can be no infringement.

VII. Conclusion

Some commentators have argued that protection for computer programs would be better provided under patent law¹³⁵ or a sui generis intellectual property regime.¹³⁶ This article, however, suggests that copyright law is flexible enough to accommodate new creative works like computer programs by applying traditional copyright doctrines and weighing modern public policy objectives. Furthermore, a copyright regime that denies a programmer protection from copying of the behavioral aspects of his programs is tantamount to providing him no protection whatsoever. This article has presented several arguments why such protection is necessary and suggested a method by which courts may properly carry out the congressional mandate to extend copyright protection to computer programs.

† J.D. Candidate, 1996, University of Pittsburgh School of Law; 1983, Master of Software Engineering, Carnegie-Mellon University; 1990, BSEE, Carnegie-Mellon University. The author has ten years of professional software engineering experience. Special thanks to Professor Pamela Samuelson of the University of Pittsburgh School of Law for her comments and help.

1. Pub. L. No. 96-517 § 10(a) (1980) U.S.C.A.A.N. 94 Stat. 3028. The 1980 amendments changed the Copyright Act by adding a definition of "computer program" to section 101, and by adding section 117, which grants certain rights to users of computer programs.

2. *Id.* The 1980 amendments to the Copyright Act adopted almost verbatim the recommendations of the NATIONAL COMMISSION OF NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT (1979) [hereinafter CONTU REPORT]. The report called for the courts to determine the protectible elements of software. CONTU REPORT, 18.

3. See Irwin R. Gross, *A New Framework For Software Protection: Distinguishing Between Interactive And Non-Interactive Aspects Of Computer Programs*, 20 RUTGERS COMPUTER & TECH. L. J. 107, 132 n.113 (1994).

4. Non-literal aspects of computer programs are "those aspects that are not reduced to written code." *Computer Assoc. v. Altai Inc.*, 982 F.2d 693, 696 (2d Cir. 1992).

5. For a discussion of the definition of the behavior of a computer program, see *infra* part II.

6. The term "black box," as used in computer science, describes the method of software testing whereby the functional interface of the software is tested without knowledge of or regard for the internal implementation of the functions of the program. Compare ROGER S. PRESSMAN, *SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH* 470, 484 (2nd ed. 1987) and RICHARD E. FAIRLEY, *SOFTWARE ENGINEERING CONCEPTS* 284 (1985) with Duncan M. Davidson, *Common Law, Uncommon Software*, 47 U. PITT. L. REV. 1037, 1080 (1986) (using the term "black box" in a slightly different context to support the notion that external attributes of a program "should be considered completely reverse engineerable").

7. See, e.g., *Whelan Assoc. Inc. v. Jaslow Dental Lab.*, 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1987) (holding non-literal aspects of program copyrightable); *Broderbund Software v. Unison World, Inc.*, 648 F. Supp. 1127 (N.D. Cal. 1986) (holding sequence of screens, computer screen displays protectable); *Manufacturers Technologies v. Cams, Inc.*, 706 F. Supp. 984, 993 (D. Conn. 1989) (concluding that program's copyright protected the literal and non-literal elements of the program's screen displays, user interface and structure to the extent that each contains copyrightable subject matter); *CMAX/Cleveland v. UCR*, 804 F. Supp. 337 (M. D. Ga. 1992) (holding screen displays and reports were protectable expression); *Apple Computer v. Microsoft Corp.*, 35 F.3d 1435 (9th Cir. 1994) (holding "look and feel" of user interface not protectable expression apart from individual elements of interface); *Mitek Holdings v. Arce Eng'g Co.*, 864 F. Supp. 1568 (S.D. Fla. 1994) (extending protection to text of commands and the way the program looked, sounded, and interacted with the user); *Autoskill, Inc. v. Nat'l Educ. Support Sys.*, 994 F.2d 1476 (10th Cir. 1993), *cert. denied*, 114 S.Ct. 307 (1994) (finding infringement in reading testing program where infringer had merely changed the names and sequences of the tests and made minor format changes); *Napoli v. Sears, Roebuck and Co.*, 874 F. Supp. 206 (N.D. Ill. 1995) (holding that copyright extends to computer screen displays).

8. See, e.g., *Brown Bag Software v. Symantec*, 960 F.2d 1465 (9th Cir.), *cert. denied*, 113 S. Ct. 198 (1992); *Gates Rubber Co. v. Bando Chemical Indus.*, 9 F.3d 823 (10th Cir. 1993); *Lotus Dev. v. Borland Int'l*, 49 F.3d 807 (1st Cir. 1995), *cert. granted*, 116 S.Ct. 39 (1995).

9. 775 F. Supp. 544 (E.D.N.Y. 1991), *aff'd in part, vacated in part, remanded*, 982 F.2d 693 (2d Cir. 1992).

10. 116 S.Ct. 39 (1995).

11. 17 U.S.C. § 102(a) (1988).

12. See *Baker v. Selden*, 101 U.S. 99 (1880); 17 U.S.C. § 102(b) (1988) (codifying the holding of *Baker*, and the idea/expression dichotomy).

13. At least one commentator has expressed the difference as that between interactive and non-interactive elements of the program, defining interactive elements as those which are "directly perceived by humans in the course of using the program." Gross, *supra* note 3, at 112-15.

14. Source code is defined as symbolic coding in its original form before being processed by a computer. The computer automatically translates source code into a code it can understand by a process called "compiling." DONALD SPENCER, WEBSTER'S NEW WORLD DICTIONARY OF COMPUTER TERMS 539 (5th ed. 1994) [hereinafter WEBSTER'S DICTIONARY OF COMPUTER TERMS].
15. *See supra* part II for a definition of behavior of a computer program.
16. *See* Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 DUKE L. J. 663 n.320-22 and accompanying text.
17. 17 U.S.C. § 106(1) (1988) (providing the author the exclusive right "to reproduce the copyrighted work in copies or phonorecords") and 17 U.S.C. § 106(4) (1988) (providing the author the exclusive right "to perform the copyrighted work publicly").
18. *See* Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2317 (1994) [hereinafter *Manifesto*].
19. *Id.* at 2315.
20. *See* Gross, *supra* note 3, at 114-15 ("[T]ypical computer users care very little about a program's non-interactive elements so long as they function properly.").
21. *Manifesto*, *supra* note 18, at 2318.
22. *Id.*
23. *See* CONTU REPORT, *supra* note 2, 18-23. The Congressional Commission on New Technological Uses of Copyrighted Works recommended that the copyright statute be amended to provide protection for computer programs as both literary and audiovisual works. Congress adopted CONTU's recommendations essentially verbatim.
24. A computer program is defined as a formal expression of the sequence of actions required for a data processing task; the programmer's specification of the task(s) to the computer in a formal notation that can be processed by the computer. It consists of a series of statements and instructions that cause a computer to do a specific job. WEBSTER'S DICTIONARY OF COMPUTER TERMS, *supra* note 14, at 117.
25. 35 U.S.C. § 101 (1988) (definition of "[l]iterary work").
26. H.R. REP. NO. 1476, 94th Cong., 2d Sess. 54 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5659, 5667 (protecting computer programs as literary works to the extent they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves).
27. 17 U.S.C. § 101 (1988).
28. 17 U.S.C. § 102 (1988).
29. 17 U.S.C. § 117 (1988).
30. The statute provides:

Notwithstanding the provisions of § 106, it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:

(1) that such a new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other manner, or

(2) that such new copy or adaptation is for archival purposes only and that all archival copies are destroyed in the event

that continued possession of the computer program should cease to be rightful.

Id.

31. See CONTU REPORT, *supra* note 2, 18-23.

32. See Arthur J. Levine, *Comment on Bonito Boats Follow-Up: The Likely Rejection of Nonliteral Software Copyright Protection*, 6 COMPUTER L. 29, 31 (1989); *Micro-Sparc, Inc. v. Amtype Corp.*, 592 F. Supp. 33, 35 n.7 (D. Mass. 1984) ("The CONTU Report . . . comprises the entire Legislative history of § 117.").

33. *Whelan Assoc. Inc. v. Jaslow Dental Lab*, 797 F.2d 1222, 1241 (3d Cir. 1986).

34. Allen R. Grogan, *Bonito Boats and Whelan: A Simple Contrast Between Patent And Copyright Protection*, 6 COMPUTER L. 33, 34 (1989); see also *Nichols v. Universal Pictures Corp.*, 45 F.2d 119, 121 (2d Cir. 1930) ("It is essential to any protection of literary property . . . that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations.").

35. See CONTU REPORT, *supra* note 2, at 11, 18-23.

36. *Id.* at 21.

37. *Id.* at 23.

38. See Melville Nimmer, *Declaration Regarding the National Commission on New Technological Uses of Copyrighted Works (CONTU) Final Report* (Nov. 15, 1984) reprinted in Anthony L. Clapes et al., *Silicon Epics And Binary Bards: Determining The Proper Scope Of Copyright Protection For Computer Programs*, 34 UCLA L. REV. 1493, app. (1987) [hereinafter *Nimmer Declaration*].

39. *Id.* at ¶12.

40. However, commissioners Miller and Levine disagree with Nimmer on this point. See Steven R. Englund, Note, *Idea, Process, Or Protected Expression?: Determining The Scope Of Copyright Protection Of The Structure Of Computer Programs*, 88 MICH. L. REV. 866, 888-90 (discussion views of commissioners Miller and Levine).

41. Jane C. Ginsburg, Comment, *Four Reasons and a Paradox: The Manifest Superiority of Copyright Over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559, 2567 (1994).

42. Arthur R. Miller, *Copyright Protection For Computer Programs, Databases, And Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977, 986 (1993).

43. *Id.* at 986-87.

44. *Id.* at 986. The author also compares computer programs to architectural works and states that the "recently enacted Architectural Works Copyright Protection Act strikingly echoes the present state of computer copyright law." *Id.* at 988 n.45.

45. See 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT §§ 3.01-04.

46. *Manifesto*, *supra* note 18, at 2319; see also Gross, *supra* note 3, at 115 n.28 (even programs that are not identical can compete as substitutes).

47. See Bill Curtis, *Engineering Computer "Look and Feel": User Interface Technology and Human Factors Engineering*, 30 JURIMETRICS J. 51, 53 (1989) (discussing the importance of the use interface in marketing software); see also Garard J. Lewis, Jr., Comment, *Lotus Development Corp. v. Paperback Software International: Broad Copyright Protection For User Interfaces Ignores The Software Industry's Trend Toward Standardization*, 52 U. PITT. L. REV. 689, 694-95 n.17 (1991) (citing Curtis).

48. Curtis, *supra* note 47, at 52-54.

49. See Margaret L. Pittman, Comment, *What The Judge Sees Is What You Get: The Implications Of Lotus v. Paperback For Software Copyright*, 37 WAYNE L. REV. 1527, 1585 (1991) (noting that limiting copyright protection for computer programs will not effectively protect the programmer's expression).

50. *Manifesto*, *supra* note 18, at 2317-18; see also FAIRLEY, *supra* note 6, at 9 (noting that an estimated 40% of the total development time of a software project is spent on analysis and design, while only 20% is typically spent on implementation (what is termed in this paper "translating the expressive design into literal source code"), debugging and unit testing); see also Whelan Assoc. v. Jaslow Dental Lab., 797 F.2d 1222, 1231 (3d Cir. 1986) ("[T]he coding process is a comparatively small part of programming.").

51. See *Manifesto*, *supra* note 18, at 2315-16. See also Gross, *supra* note 3, at 159 n.229 and accompanying text.

52. 101 U.S. 99 (1879).

53. *Id.* at 104, 107.

54. *Id.* at 103.

55. *Id.* at 101.

56. 17 U.S.C. § 102(b) (1988).

57. *Baker*, 101 U.S. at 104.

58. See *Nimmer Declaration*, *supra* note 38, at ¶13.

59. See *Manifesto*, *supra* note 18; Leo J. Raskind, *The Uncertain Case For Special Legislation Protecting Computer Software*, 47 U. PITT. L. REV. 1131, 1143-44 (1986) (arguing that programs are uncopyrightable because of their utilitarian nature).

60. 347 U.S. 201 (1954) (holding that statuette of a dancing figure used as a lamp base was copyrightable as a work of art despite being used in a useful object such as a lamp). "Artistic articles are protected in 'form but not their mechanical or utilitarian aspects.'" *Id.* at 218 (citing 1909 Copyright Act, 17 U.S.C. § 202.8).

61. 17 U.S.C. § 101 (1988).

62. See Jack E. Brown, *"Analytical Dissection" Of Computer Software-Complicating The Simple And Confounding The Complex*, 25 ARIZ. ST. L. J. 801, 833 (1993) (discussing how the term "useful article" only appears in §§ 101 and 113 of Copyright Act, which pertain to the scope of protection for pictorial, graphic or sculptural works).

63. See *E.F. Johnson Co. v. Uniden Corp. of Am.*, 623 F. Supp. 1485, 1498 (D. Minn. 1985) (rejecting the characterization of plaintiff's program as "a useful work" and affirming Congress' decision to treat computer programs as "literary works:" "[T]he limitations placed on the copyrightability of useful articles by section 101 of the Act are simply not applicable here.").

64. See *supra* part III.C.

65. "The copyright in a compilation . . . extends only to the material contributed by the author of such work, as distinguished from the preexisting material employed in the work. . . ." 17 U.S.C. § 103(b) (1988).

66. *Miller*, *supra* note 42, at 1003. See also *supra* note 45, at §§ 3.01-04; *Harper House v. Thomas Nelson, Inc.*, 889 F.2d 197, 204-205 (9th Cir. 1989) (uncopyrightable elements of a notebook protectable as a compilation).

67. 797 F.2d 1222 (3d Cir. 1986).

68. *Id.* at 1239.
69. *See* Broderbund Software v. Unison World, 648 F. Supp. 1127, 1132 (N.D. Cal. 1986).
70. *See* Mitek Holdings v. Arce Eng'g Co., 864 F. Supp. 1568 1576 (S.D. Fla. 1994).
71. *See* Whelan, 797 F.2d 1222, 1240 (3d Cir. 1986).
72. *See, e.g.*, Brown Bag Software v. Symantic Corp., 960 F.2d 1465 (9th Cir. 1992); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 844 (10th Cir. 1993); Computer Assoc. Int'l. v. Altai, 775 F. Supp. 544, 560 (E.D.N.Y. 1991), *aff'd*, 982 D.2d 693 (2nd Cir. 1992).
73. John Houston, Comment, *A Unified Test For The Copyright Protection Of The User Interface To Computer Programs*, 32 DUQ. L. REV. 133, 142 (1993).
74. 740 F. Supp. 37 (D. Mass. 1990).
75. *Id.* at 56 (emphasis added).
76. 797 F.2d 1222 (3d Cir. 1986).
77. *Id.* at 1234.
78. *Id.*
79. *Id.* at 1238-40.
80. *Id.* at 1240.
81. *Id.* at 1235.
82. *See, e.g.*, Englund, *supra* note 40; Michael A. Jacobs, *Copyright and Compatibility*, 30 JURIMETRICS J. 91 (1989); Andrew O. Martyniuk, Comment, *Abstraction-Filtration-Comparison Analysis and the Narrowing Scope of Copyright Protection for Computer Programs*, 63 U. CIN. L. REV. 1333 (1995); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045 (1989); Pamela Samuelson, *Computer Programs, User Interfaces, and Section 102(b) of the Copyright Act of 1976: A Critique of Lotus v. Paperback*, 6 HIGH TECH. L. J. 209 (1991); Peter G. Spivak, Comment, *Does Form Follow Function? The Idea/Expression Dichotomy in Copyright Protection of Computer Programs*, 35 UCLA L. REV. 723 (1988); Nicholas P. Terry, *GUI Wars: The Windows Litigation and the Continuing Decline of "Look And Feel,"* 47 ARK. L. REV. 93, 142, n.222-24 (1994).
83. 775 F. Supp. 544 (E.D.N.Y. 1991), *aff'd in part, vacated in part, remanded*, 982 F.2d 693 (2d Cir. 1992).
84. Engineering Dynamics v. Structural Software, Inc., 26 F.3d 1335, 1342-43 (5th Cir. 1994); Kepner-Tregoe, Inc. v. Leadership Software, Inc., 12 F.3d 527, 536-37 (5th Cir. 1994).
85. Apple Computer v. Microsoft Corp., 35 F.3d 1435, 1442-43 (9th Cir. 1994).
86. Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 834 (10th Cir. 1993); Autoskill v. Nat. Educ. Support Sys., 994 F.2d 1476, 1487-98 (10th Cir. 1993).
87. Atari Games Corp. v. Nintendo of Am., 975 F.2d 832, 839 (Fed. Cir. 1992).
88. Mitek Holdings v. Arce Eng'g Co., 864 F. Supp. 1568, 1577-78 (M.D. Fla. 1994); CMAX/Cleveland, Inc. v. UCR, Inc., 804 F. Supp. 337, 352-54 (M.D. Ga. 1992).

89. *Altai*, 775 F. Supp. at 559-60.
90. *Id.* at 559.
91. "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b) (1988).
92. *Altai*, 775 F. Supp. 544 at 560.
93. *Id.*
94. *See infra* part VI.
95. 45 F. 2d 119, 121 (2d Cir. 1930).
96. *See supra* note 68 and accompanying text.
97. *See infra* part VI.
98. *See supra* note 7.
99. 994 F.2d 1476 (10th Cir. 1993).
100. 864 F. Supp. 1568 (S.D. Fla. 1994).
101. *Id.* at 1577-78.
102. *Id.* at 1580.
103. 35 F.3d 1435 (9th Cir. 1994).
104. *Id.* at 1438.
105. *Id.* at 1446.
106. *Id.* at 1447.
107. 798 F. Supp. 1499 (D. Colo. 1992), *vacated and remanded in part, aff'd in part*, 9 F.3d 823 (10th Cir. 1993).
108. *Id.* at 1518-19 (emphasis added).
109. 9 F.3d 823 at 835.
110. 49 F.3d 807 (1st Cir. 1995).
111. *Id.* at 815.
112. *Id.* at 814.
113. The Supreme Court has granted a writ of certiorari in the *Lotus* case. *Lotus Dev. Corp. v. Borland Int'l*, 116 S.Ct. 39 (1995).

114. If a program is literally copied (i.e. the source code of the program is duplicated) then the behavior of the copy would be identical to the behavior of the original, and would thus infringe under this test.

115. 982 F.2d 693, 706-711 (2d Cir. 1992).

116. *See, e.g.*, Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 816 (1st Cir. 1995), *cert. granted*, 116 S.Ct. 39 (1995) ("*Altai* test may provide a useful framework for assessing the alleged nonliteral copying of computer code."); Lotus Dev. Corp. v. Borland, Int'l., 799 F. Supp. 203 (D. Mass. 1992) (concluding that the test developed by the district court was "compatible substantively, though different in methodology" than the *Altai* test); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 828 (10th Cir. 1993) ("In substantial part, we adopt the 'Abstraction-Filtration-Comparison' test."); Productivity Software Int'l. v. Healthcare Tech., 1995 WL 437526 (S.D.N.Y. 1995) (using *Altai* test to determine substantial similarity), Triad Sys. Corp. v. Southeastern Express Co., 31 U.S.P.Q.2d (BNA) 1239, 1248 (N.D. Cal. 1994) (using first two steps of the *Altai* test for distinguishing between protectable expression and unprotectable ideas, processes, etc., where substantial similarity was not disputed); Atari Games Corp. v. Nintendo of Am., Inc., 30 U.S.P.Q.2d 1401 (BNA) (N.D. Cal. 1993) ("In our view, in light of the essentially utilitarian nature of computer programs, the Second Circuit's approach is an appropriate one.").

117. *Altai*, 982 F.2d at 706.

118. *Id.* at 707.

119. *Id.*

120. 843 F.2d 600 (1st Cir. 1988).

121. *Id.* at 606.

122. *Altai*, 982 F.2d at 708.

123. 3 NIMMER & NIMMER, *supra* note 45, at 13-65.

124. *Altai*, 982 F.2d at 709.

125. *Id.*

126. 17 U.S.C. § 102(a) (1988).

127. *Altai*, 982 F.2d at 710.

128. *Id.* at 710.

129. 788 F. Supp. 78 (D. Mass. 1992), 799 F. Supp. 203 (D. Mass. 1992), 831 F. Supp. 202 (D. Mass. 1992), 831 F. Supp. 223 (D. Mass. 1992), *rev'd.*, 49 F.3d 807 (1st Cir. 1995), *cert. granted*, 116 S.Ct. 39 (1995).

130. A menu is a list of command options available to the user of a program. *See* WEBSTER'S DICTIONARY OF COMPUTER TERMS, *supra* note 14, at 365. The menu command structure of 1-2-3 consists of a series of commands, spaced horizontally across the top of the screen, from which the user can select, using the computer's mouse or keyboard. These commands are the top of a tree of commands available to the user. When one of these commands is selected, a sub-menu of commands, or another branch in the tree of commands, appears. Each item in the sub-menu that the user could select is either another branch in the tree of commands, leading to another sub-menu, or a leaf of the tree, representing a command which tells the program to perform a specific function.

131. A macro is a single command, made up by a user, which can replace a series of the 1-2-3 commands. A user can define a series of 1-2-3 commands, and assign a name to the series. This "macro" can then be invoked by using the assigned name, instead of having to enter each 1-2-3 command in the series individually. *See* WEBSTER'S DICTIONARY OF COMPUTER TERMS, *supra* note 14, at 351.

132. *Lotus*, 799 F. Supp. at 223-23, 831 F. Supp. at 245.

133. *Lotus*, 49 F.3d at 815.

134. *Lotus*, 116 S.Ct. 39 (1995).

135. Mark A. Lemley, *Convergence in the Law of Software Copyright*, 10 HIGH TECH. L.J. 1 at 26 (1995) ("Altering copyright law rather than employing patent protection has arguably overprotected computer software . . .").

136. *See generally Manifesto*, *supra* note 18.