btlionline BERKELEY TECHNOLOGY LAW JOURNAL
University of California, Berkeley, Boalt Hall School of Law

Article

Convergence in the Law of Software Copyright?

Mark A. Lemley †

Table of Contents

I. Introduction

Courts and commentators have spilled a great deal of ink-and paid an enormous amount in legal fees-over the last 15 years in an effort to determine the appropriate scope of copyright protection for computer software.[1] To a large extent, this debate has focused on how software copying should be tested. On one side, "broad constructionists" have emphasized the need to compare the copyrighted and accused works *as a whole*, in order to give protection to the "total concept and feel" of the works. On the other side, "narrow constructionists" have urged the methodical dissection of copyrighted works into their component parts in order to determine what exactly qualifies for copyright protection.

For all intents and purposes, this aspect of the debate is over. In the last three years, virtually all the courts to consider this issue have lined up with the "narrow constructionists," engaging in "analytic dissection" of computer programs in order to determine whether any copyrightable expression has actually been copied.[2] Most commonly, this analytic dissection has taken the form of the "abstraction-filtration-comparison" test set forth in *Computer Associates v. Altai*.[3] While there are still a few courts in which the "total concept and feel" approach remains the law, the approach is moribund: since *Altai* was decided, no court has endorsed the broader "total concept and feel" approach.[4]

This does not mean that we can all go home, however. Rather than ending, the debate over software copyright law appears to be shifting its focus. Having finally resolved the debate that has been plaguing software copyright law since its inception, courts are discovering to their chagrin, that deciding *what* test to apply actually tells you very little about *how* to apply that test. Despite the convergence of courts on *Altai*'s filtration approach, courts remain fundamentally conflicted in deciding how broadly to protect software copyright.[5] Further, there remains a good deal of misunderstanding about what exactly it means to "abstract" and "filter" a computer program.

Part II of this Article traces the history of legal protection for computer software, beginning with trade secret protection. It discusses the relatively quick development of a rule that literal copying of computer programs is prohibited, and the more intractable problem of non-literal infringement. Part III chronicles the convergence of courts and commentators on the filtration test. It then focuses on court decisions since *Altai* which have attempted to apply this test, and concludes that there is still substantial disagreement among the courts as to how that test should work in practice. Finally, Part IV suggests a unified approach to evaluating non-literal infringement in software copyright cases. This approach focuses on exactly what is alleged to have been copied. It also acknowledges the increasing role of patent law in protecting computer software, and the role of other copyright concerns such as compatibility and fair use. The result of this unified approach is to provide relatively narrow copyright protection for computer programs in most cases of non-literal infringement. In this part, I examine some of the implications of this unified approach by looking at its effect on recent cases such as *Lotus v. Borland* [6] and *Apple v. Microsoft*.[7]

II. The History of Software Protection

As is common in intellectual property law, technological development in the computer industry has outrun the pace of legal change. In the 1960s and early 1970s, federal intellectual property law did not protect computer software. As a result, software vendors attempted to protect their programs in two ways-through trade secrets laws and by contract. But the vendors faced a problem: if they were selling multiple copies of a program to whomever wanted them, how could the program be a secret? The answer was that, while the program might be widely distributed, in fact all that was sold was a disk containing object code. Object code is virtually impossible for humans to read without machine assistance.[8] Because of this, some courts held quite early in the history of computer software protection that widely distributed programs could in fact retain their trade secret status.[9]

This was the purpose of the proprietary rights contract.[10] Software vendors needed proof that they were not in fact disclosing their trade secrets by selling copies to whoever wanted them. To provide such proof, they created the legal fiction that they were really licensing rather than selling their software. Because the "license" contained provisions that required customers to keep the software confidential, many courts held that the trade secrets contained therein could be protected.

As we shall see, however, the shortcomings of trade secret protection led software vendors to seek alternate avenues to protect their works.[11]

A. Establishing Copyright Protection for Software

Copyright law was originally established to prevent the unauthorized printing of books.[12] To a large extent, the scheme of copyright protection still shows its literary roots. Copyright law was designed in order to protect physical works of creative authorship which could be *used* without being *copied*. The law's assumption that the objects of copyright protection are physical has colored the application of copyright law to computer software cases. Courts repeatedly look-often in vain-to the familiar physical world of copyright to find analogies to apply in the world of computer software.

The fundamental principle of copyright law is that unauthorized copying is not permitted.[13] Copying violative of the statute includes not only the literal taking of the words or expression of another, but also what is called "non-literal infringement"-the taking of the essence of the author's expression without using the author's actual words. Were copyright protection limited to literal infringement, as Judge Hand has noted, a plagiarist could "escape by immaterial variations."[14] Indeed, protection for copyrighted works extends beyond even the language or particular creative expression used, to encompass the "structure, sequence and organization" of a work,[15] and its "total concept and feel."[16] Furthermore, when copying cannot be proven by direct evidence, it may be inferred from proof of the infringer's access to the copyrighted work plus proof of substantial similarity between the two works.[17]

Not all copying rises to the level of copyright infringement, however. There are several important limits on the rights of copyright owners to protect their works. First, and most important, a copyright plaintiff must prove that the defendant has copied *protectable* *expression*.[18] In particular, a copyright owner may not lay claim to the *ideas* within her work, but only to her particular means of expressing those ideas.[19] Further, copyright protection does not extend to any part of the work that consists of unprotectable facts,[20] stock literary or artistic themes,[21] or expression that is not sufficiently creative or original with the copyright owner to qualify for protection.[22] Finally, copying of protectable expression is only actionable if the copying is substantial.[23]

In the computer software context, each of these basic elements of copyright law has had to be painstakingly reestablished. Even after the passage of the 1976 Copyright Act, it was not obvious to the courts that the copyright laws protected computer programs at all.[24] The Congressional Commission on New Technological Uses of Copyrighted Works (CONTU) recommended that the copyright statute be amended to provide protection for computer programs. CONTU's recommendations were adopted essentially verbatim by Congress in 1980.[25]

Even after the 1980 amendments, establishing copyright protection for the literal text of computer programs was a protracted process. Courts separately considered and affirmed the copyrightability of source code,[26] object code,[27] microcode or "firmware,"[28] applications programs,[29] and operating systems.[30] A similar debate occurred over the literal copyrightability of program outputs, such as video game displays.[31] In each case, courts found the literal elements of computer programs copyrightable only over vigorous opposition. Nonetheless, by the mid-1980s it had been firmly established that computer programs were "literary works" that could not be literally copied, and that program outputs were "audiovisual works" that could not be literally copied. Courts then turned to the thornier question of non-literal infringement.

B. The Problem of Non-Literal Infringement

The difficulty with non-literal copyright infringement is that when only concepts and not actual language have been copied, courts are put to the test of distinguishing idea from expression. This is not a problem that arises only in the computer context. Indeed, Judge Learned Hand commented early and often on the arbitrariness of any such determination.[32] Two factors make the idea-expression dichotomy particularly difficult to apply in computer cases, however. First, computer programs are written for a utilitarian purpose.[33] Expression in the code or structure and organization of a program is normally only incidental to that purpose. Courts must therefore identify and protect that incidental material, while leaving the functional aspects of the program free for all to duplicate. Second, computer programs are technically complex and inaccessible to the lay person, a category which includes most federal judges. Judges are therefore forced to rely on second-hand knowledge and testimony about the programming process to a far greater extent than in other literary cases.

These problems have driven a number of courts to search for rules to help them evaluate non-literal infringement in software cases. Two such rules grew to prominence early in the history of computer copyright cases. First, some courts solved the "problem" of non-literal infringement by refusing to protect computer programs against non-literal infringement at all. The clearest example of this approach was Judge Higginbotham's decision in *Synercom Technology v. University Computing Co.*[34] In that case, Synercom had designed a series of "input formats" which would accept data from users in conjunction with a program which analyzed engineering problems with the design of buildings. EDI, the defendant, wrote a program in a different language which accepted information in the same formats as Synercom's program. The court held that the sequencing and ordering of data inputs constituted the idea of Synercom's program, rather than its expression, and therefore found no infringement in the copying of those sequences.[35] Judge Higginbotham explained that "[h]ere . . . the form, arrangement, and combination is itself the intellectual conception involved. It would follow that only to the extent the expressions involve stylistic creativity *above and beyond* the bare expression of sequence and arrangement, should they be protected."[36]

Synercom's conclusion was endorsed, albeit indirectly, by the Fifth Circuit's decision in *Plains Cotton Cooperative Ass'n Inc. v. Goodpasture Computer Service*.[37] That case held that the structure and organization of computer input formats could not be protected because they were determined in large part by functional considerations. The court did not expressly hold that input formats could never be protected, but it did cite the *Synercom* decision with approval.[38]

The *Synercom* approach to the non-literal infringement problem has the virtue of simplicity. Unfortunately, it also has the effect of denying copyright protection even to obviously expressive elements of a computer program, such as original compilations and designs, if an infringer copies the "look and feel" of those elements, rather than their actual implementation in code. Judge Hand's warning about plagiarism by means of "immaterial variations"[39] seems particularly apt here. Under the Fifth Circuit rule, copiers were free to take the "heart" of a computer program, including its expressive elements, a freedom they did not have with any other type of work.[40]

To avoid this problem, while still devising a rule that was easy to apply, some courts went to the opposite extreme, protecting nearly all elements of a computer program against non-literal infringement. The paradigm case of this "broad protectionism" is the Third

Circuit's decision in *Whelan Associates Inc. v. Jaslow Dental Laboratory Inc.*.[41] There, the court was called upon to evaluate a claim of infringement of a custom computer program for record keeping in a dental laboratory. Whelan was a computer company that had developed the program to the specifications of its customer, Jaslow. When Jaslow developed and began marketing its own program in competition with Whelan, Whelan sued. The record indicated that Jaslow had not copied Whelan's code, but that the two programs were similar in their overall structure, sequence and organization.

The *Whelan* court began its analysis by concluding that the non-literal expressive elements of a computer program were entitled to copyright protection, citing the universal rule with regard to other types of works.[42] The court acknowledged that this result put it at odds with *Synercom*, but found that there was "no statutory basis for treating computer programs differently from other literary works in this regard."[43]

Having decided that the non-literal elements of computer programs were entitled to protection, the court set about the task of formulating a test to distinguish protectable from unprotectable elements.[44] The Court reviewed *Baker v. Selden* in detail, and drew its test from what it took as the central lesson of that case:

> [T]he purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of that idea . . . Where there are various means of achieving the desired purpose, then the particular means chosen is not necessary to the purpose; hence, there is expression, not idea.[45]

The court felt that the test it articulated would "provide the proper incentive for programmers by protecting their most valuable efforts," citing evidence that most of the cost of developing a computer program is in designing the structure and logic of the program.[46]

The court's application of its test to the case before it was straightforward. The court described the "idea" of the computer program as "the efficient management of a dental laboratory (which presumably has significantly different requirements from those of other businesses)."[47] The choice of this basic concept as "the" idea of the Dentalab program made application of the idea-expression dichotomy quite simple: "Because that idea could be accomplished in a number of different ways with a number of different structures, the structure of the Dentalab program is part of the program's expression, not its idea."[48] The court acknowledged that its approach was at odds with *Synercom*, but felt that there was no merger of idea and expression in the case before it, since the "idea" of efficiently managing a dental laboratory was separable from the many possible ways of accomplishing this goal.[49]

During the same period, a number of district courts took an approach similar to *Whelan*, at least to the extent of rejecting *Synercom* and protecting software against non-literal copyright infringement.[50] On that issue, *Whelan* remains uncontroversial to this day. Indeed, the Fifth Circuit has since reversed *Synercom* and affirmed the copyrightability of non-literal elements of computer programs.[51]

Whelan's broader holdings, however, sparked a firestorm of controversy, with most commentators critical of the decision.[52] Commentators have pointed to two weaknesses in *Whelan*'s approach. First, *Whelan*'s statement of the idea-expression dichotomy as excluding protection for the idea of the program and everything necessary to that idea is accurate but arguably incomplete. The court left the definite impression that this central idea was the *only* thing that could not be protected by copyright. As subsequent cases have noted, under traditional copyright rules protection does not extend to facts, "scenes a faire," elements dictated by efficiency or functional considerations, patentable processes, material drawn from other works or from the public domain, and expression which has "merged" with the underlying idea.[53] While certain language in *Whelan* indicates that the court recognized that facts and scenes a faire at least were unprotectable,[54] the court apparently considered them unprotectable only to the extent that they had merged with the overarching idea of the work.[55]

The second problem with *Whelan* is its definition of the unprotectable idea. As Judge Hand noted long ago, ideas can be defined at different levels of abstraction.[56] Choosing the level at which an idea is defined is critical to determining how much expression will be protected; the more general the idea, the more "expressive" material will fit under its umbrella. As an admittedly extreme example, the *Whelan* court might have defined the idea of the Dentalab program very generally (say, "to make money"). Under the test the court articulates, if making money is the idea, anything not necessary to that idea is expression protectable by the copyright laws. Since a dental laboratory is not the only way to make money, the concept of a dental laboratory itself would be protectable "expression" that could not be copied under this test. Most people would agree that this is an absurd result, but it does demonstrate the power that defining the idea has in determining the scope of copyright protection.

Unfortunately, the *Whelan* court's definition of the idea of the Dentalab program is not much narrower than our hypothetical. By defining the idea as "efficiently managing a dental laboratory," the court opened the way for any particular means of (efficient) management to be protected. For example, since it is presumably possible to manage a dental laboratory efficiently without using a computer program, Whelan's *entire program* is arguably merely an "expression" of the overarching idea of efficient management. Certainly, the court translates its broad idea into broad protection for the structure of the Dentalab program, even though the court conceded that similar structures could also be found in competing, presumably non-infringing programs.[57]

In short, the problem with *Whelan* is the same as the problem with *Synercom*, from the opposite side. In both cases, the courts sacrificed accuracy in separating protectable from unprotectable material in order to achieve a workable rule that is easy to apply. Neither *Synercom*, which protects virtually none of a program except the literal code, nor *Whelan*, which protects virtually the entire program, actually address the difficult problem of how to separate idea from expression in the context of computer software. Instead, both lines of authority do their best to pretend the problem does not exist.

III. Convergence: Modern Trends in Software Copyright

A. *Altai* and the Filtration Bandwagon

It was not until 1992 that an appellate court tackled this tough problem head on. In *Computer Associates Int'l, Inc. v. Altai, Inc.,*[58] Computer Associates ("CA") sued Altai for allegedly copying portions of its operating system scheduling program in order to make Altai's program compatible with CA's. An Altai employee had originally copied portions of CA's code directly; when Altai found out about the copying, it rewrote its program so that it did not include any CA code. Nonetheless, CA claimed that the rewritten Altai program infringed CA's copyright in the non-literal elements of its program.[59]

The Second Circuit began its discussion by agreeing with *Whelan* and its progeny that copyright protection extends to the non-literal elements of computer programs.[60] However, the court said, "that conclusion does not end our analysis. We must determine the *scope* of copyright protection that extends to a computer program's non-literal structure."[61] In this second determination, the *Altai* court diverged substantially from the method of analysis used in *Whelan*. The *Altai* court labeled *Whelan*'s application of the idea-expression dichotomy "suspect because it is so closely tied to what can now be seen-with the passage of time-as the opinion's somewhat outdated appreciation of computer science."[62]

In its place, *Altai* adopted a three-step procedure that has come to be known as the abstraction-filtration-comparison approach.[63] Rather than treating the non-literal elements of a computer program as a unified whole, the *Altai* approach uses "analytic dissection"[64] to break the program into its constituent parts, and then evaluates the copyrightability of each of those parts. The first step in this process is to break the "structure" of a computer program into different levels of abstraction, based on Judge Hand's test in *Nichols v. Universal Pictures Corp*.[65] At its lowest level, the structure of a computer program is based around a set of individual mathematical instructions. At progressively higher levels, these instructions are organized into subroutines, subroutines into routines, routines into software modules, modules into higher-level modules, and these high-level modules into the overall function of the program.[66]

Once this separation has been completed, the *Altai* test proceeds to "filter" the unprotectable elements out of the program structure at each level of abstraction.[67] Like *Whelan*, the *Altai* court concluded that the underlying idea of a program, and any structural elements necessary to that idea, are unprotectable. In addition, *Altai* identified several factors to be considered in the filtering process. First, the court removed from consideration elements of CA's program structure which were dictated by efficiency, on the grounds that the idea and expression have merged where "efficiency concerns . . . so narrow the practical range of choice as to make only one or two forms of expression workable options."[68] Because computer programs are utilitarian works, and the goal of computer programmers is normally to implement their ideas efficiently, this is a significant limitation on the protection afforded program structure, particularly at higher levels of abstraction.

Second, the *Altai* court removed from consideration the programming equivalent of *scenes a faire*-standard programming techniques and elements of the program dictated by external factors. In particular, the court identified five elements of computer programs that were not entitled to protection under the *scenes a faire* doctrine.[69] Finally, *Altai* held that factual material or material taken from the public domain could not be included in the copyrightable expression of the program.[70]

Once the unprotectable elements had been filtered out of the program, what remained was "a core of protectable expression."[71] The court applied the traditional infringement test to this core expression, comparing the equivalent structures in the two works to determine whether they are substantially similar and, if so, whether any copying was substantial.[72] Under this approach, a plaintiff will be successful only if she can demonstrate substantial copying of protectable expression.

Altai rejected CA's argument that its filtration approach would inadequately protect computer programmers and therefore would be a disincentive to computer research and development. Even if that were true, the court said, the Supreme Court's decision in *Feist Publication v. Rural Telephone Service*[73] makes it clear that "substantial effort alone cannot confer copyright status on an otherwise uncopyrightable work."[74] The court noted that "[t]he interest of the copyright law is not in simply conferring a monopoly on industrious persons, but in advancing the public welfare through rewarding artistic creativity, in a manner that permits the free use and development of non-protectable ideas and processes."[75]

While the *Altai* filtration approach was criticized by large computer companies who are commonly copyright plaintiffs-IBM's lawyer called it "a legal Chernobyl";[76] Apple's lawyer said it "complicated the simple and confounded the complex"[77]-it has found rapid acceptance among the courts.[78] In the two and one-half years since *Altai* was decided, the Second Circuit's filtration approach[79] has been endorsed by the Federal Circuit,[80] the Fifth Circuit,[81] the Ninth Circuit,[82] the Tenth Circuit,[83] and district courts in the Eleventh Circuit.[84] *Altai* has also been endorsed by courts in Canada,[85] the United Kingdom,[86] and France.[87] Only the First Circuit has questioned the filtration approach, and they complained that the test was too broad rather than too narrow.[88] Certainly, no court has followed *Whelan* since *Altai* was decided.

To be sure, application of the filtration test has not been uniform in the courts. The test has acquired some accouterments (and perhaps greater sophistication) on its way to general acceptance. The Tenth Circuit decision in *Gates Rubber* is particularly notable in this regard. In that case, the court acknowledged that "[a]pplication of the abstractions test will necessarily vary from case-to-case and program-to-program. Given the complexity and ever-changing nature of computer technology, we decline to set forth any strict methodology for the abstraction of computer programs."[89] Nonetheless, the court identified six levels of "generally declining abstraction": (1) the main purpose of the computer program, (2) the structure or architecture of a program, generally as represented in a flowchart, (3) "modules" which comprise particular program operations or types of stored data, (4) individual algorithms or data structures employed in each of the modules, (5) the source code which instructs the computer to carry out each necessary operation on each data structure, and (6) the object code which is actually read by the computer.[90] The court used these levels of abstraction to facilitate its analysis of the program at issue.

The *Gates Rubber* court also gave further content to the filtration part of the *Altai* analysis. The court filtered out six unprotectable elements: ideas,[91] the processes or methods of the computer program,[92] facts,[93] material in the public domain,[94] expression which has "merged" with an idea or process,[95] and expression which is so standard or common as to be a "necessary incident" to an idea or process.[96] Finally, the court indicated that comparison of the protected elements of a program should be done on a case-by-case basis, with an eye toward determining whether a substantial portion of the protectable expression of the original work has been copied.[97] The *Gates Rubber* court specifically rejected a comparison of the "total concept and feel" of the two programs, on the grounds that that test "was developed in different contexts and it is not very helpful in comparing similarities among protected components of computer codes."[98]

Gates Rubber adds a great deal of detail to the framework established by *Altai.* The Tenth Circuit's gloss on *Altai* has itself received substantial support among the courts adopting the filtration approach.[99] In spite of the differences in the particular filtration test applied by different courts, it is now possible to say with some certainty that the Third Circuit's approach in *Whelan* is dead. Between 1992 and 1994, the federal courts converged on the filtration approach exemplified by *Altai* and *Gates Rubber*.

B. Difficulties in Applying *Altai*

Given the time, money and energy that have been poured into the debate over how to evaluate non-literal infringement claims in software cases, it would be reasonable to assume that judicial resolution of that debate would go a long way towards determining the scope of protection actually afforded computer software. Reasonable, but wrong. The Fifth Circuit observed in *Engineering Dynamics* that its adoption of the filtration approach "should not convey a deceptive air of certitude about the outcome of any particular computer copyright case."[100] Indeed, a quick review of the recent cases applying the filtration approach indicates that filtration is not particularly well correlated with "narrow" copyright protection. Not only have at least two courts given a broad scope of protection to programs while applying the filtration test,[101] but one court has criticized the *Altai* filtration approach for giving too much protection to programs.[102]

To be sure, several courts applying the filtration test have concluded that the particular programs at issue are deserving of little if any protection against non-literal infringement. For example, in *Altai*, the Second Circuit affirmed the district court's determination that the OSCAR program "effectively contained no protectable expression whatsoever."[103] Not surprisingly, therefore, it also agreed that similarities between the few remaining protectable elements were de minimis and did not warrant a finding of copyright infringement.[104] Similarly, the *Gates Rubber* court found that most elements of the program structure and command operations were not protectable, although it did remand to the district court for further evidence on similarities between the screen menus and sorting criteria in the two programs, as well as to consider the effect of programming errors common to the two programs.[105] And in *Apple*, the Ninth Circuit held that virtually none of the elements of Apple's graphical user interface (GUI) were protectable in the circumstances of that case.[106] While it did consider the program as a whole protectable as a compilation, that compilation received only thin copyright protection and was not infringed by the defendants.[107]

On the other hand, some courts that have adopted the filtration approach (at least in name) have nonetheless granted broad protection to the non-literal elements of computer software. For example, the district court in *CMAX* held that the organization of the blank forms designed to collect information from the user of the program was not dictated by function or by industry custom, and therefore was protectable.[108] And the Fifth Circuit's decision in *Kepner-Tregoe* defines expression quite broadly in finding non-literal infringement.[109] Still other courts have taken a similarly broad approach in applying the filtration test, but with certain reservations or limitations. For example, in *Atari*, the Federal Circuit held that Nintendo's instruction code for its lockout chip-in essence, a method of generating the proper data stream to unlock the Nintendo system-was copyrightable because the instructions themselves were arbitrary, and therefore not dictated by function.[110] At the same time, the court seemed to suggest that Atari would not have infringed Nintendo's copyright if it had used one of the "multitude of different ways" of generating the data stream necessary to unlock the Nintendo console.[111]

To complicate matters further, the First Circuit's recent decision in *Lotus v. Borland* takes another tack altogether, eschewing the filtration test entirely. In that case, Lotus had sued Borland for copying the menu command hierarchy of its 1-2-3 spreadsheet program. Borland admitted copying the menu structure, but both parties agreed that Borland had not taken any Lotus code. The First Circuit concluded that Lotus' menu structure was not entitled to *any* copyright protection because it was a "method of operation," which section 102(b) of the Copyright Act says is unprotectable. The *Lotus* court criticized *Altai* as "misleading because, in instructing courts to abstract the various levels, it seems to encourage them to find a base level that includes copyrightable subject matter that, if literally copied, would make the copier liable for copyright infringement."[112] The filtration test "obscures the more fundamental question of whether a menu command hierarchy can be copyrighted at all."[113]

This wide divergence between results achieved by courts applying the filtration test cannot be explained by the nature of the programs at issue. Courts have reached different results with respect to protection for GUIs and menu-driven I/O formats,[114] with respect to copying of compatibility elements,[115] and with respect to the copying of program structures.[116] Nor can the amount of copying alone explain this divergence. Some courts have found no infringement in spite of substantial copying,[117] while other courts have found infringement even in cases of minimal copying.[118]

It would seem, then, that the apparent harmony evident in recent decisions is illusory. When the details of each case are examined, there appear to be significant differences between the results courts reach using the filtration test. While the courts have finally agreed on some first principles in software copyright law, much more obviously remains to be done. In the next section, I offer some suggestions.

IV. Beyond the Protection Debate-Proposals

To some extent, the differing outcomes courts have reached in applying the filtration test may simply reflect their political predispositions towards "high" or "low" protectionism.[119] Some courts may simply be more willing to find infringement than others given the same set of facts. But I do not think that application of the filtration test need be entirely ad hoc. In this section, I propose a unified approach to the abstraction-filtration-comparison test which I hope will resolve much of this difficulty.

A. The Abstract Nature of Abstraction

The "abstraction" part of the *Altai* test has caused a good deal of consternation among lawyers and courts attempting to apply the test. Lawyers argue over whether programs are in fact written using the levels of abstraction described in *Altai*. Courts adjust the test to fit their particular view of programming. Lawyers and courts then tangle again over *which level of abstraction* is the proper one to apply in their case, how the levels apply to unusual parts of programs (like graphical user interfaces and other screen displays), and even whether the copying at that level of abstraction was literal or not.[120]

All of this debate misconstrues and unnecessarily complicates the abstraction step of the *Altai* test. The problem is that lawyers and courts take the abstraction step too literally, forgetting that it is intended to be, in fact, an *abstraction*. It is easiest to see this problem in the context in which the abstraction test was first applied-a claim of infringement of a traditional literary work.[121]

Judge Hand suggested that "[u]pon any work . . . a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the [work] is about, and at times might consist only of its title . . . ."[122] For example, in the case of a book, the "lowest" level of abstraction is the text of the book itself. At higher levels of abstraction, the book might consist of a paragraph-by-paragraph outline or summary, a chapter-by-chapter outline or summary, a general description of plot and characters, and (at the highest level) the thesis of the book itself. If a book is alleged to infringe the copyrighted work, we first break the copyrighted book into "levels of abstraction"[123] and do a quick look at the accused work to see what it is that the plaintiff claims was copied.[124] If the actual text has been taken, the "abstractions" analysis merges into the test for literal infringement. But if the defendant has not copied any actual text, he may still be liable if he has copied the organization or the "texture" of the book at a higher level of abstraction.

Of course, in the literary context we all know that this "higher level of abstraction" is merely an artificial construct to help us decide the case. There may or may not be a "chapter outline" and a "paragraph outline" which were used in writing the copyrighted work. But it doesn't really matter. Certainly, the inquiry is not whether the defendant had access to and copied from the outline, but whether he copied *from the finished work at that particular level of generality*. Once we have identified the level (or levels) of generality at which copying is alleged to have occurred, we can proceed to the next step-filtration-to decide what parts of the copyrighted book are protectable at that level.

Unfortunately, in the computer cases everyone seems to have forgotten that the abstraction test is only a judicial construct. Programs are composed of object code, just as books are composed of words. If a copyright defendant has not copied the object code itself, the question becomes whether she has copied the program at a higher level of generality (or "abstraction"). But courts should not care whether the defendant actually had access to and copied the plaintiff's flow chart, just as they do not care whether the defendant had access to the plaintiff's paragraph outline in the literary context. The only relevant question is the level of generality at which the copying is alleged to have occurred. Once this level of abstraction is identified (and in many cases copying may be alleged at more than one level of abstraction), courts can proceed to decide how much protection the copyrighted program is entitled to at that level.

The First Circuit's decision in *Lotus v. Borland* seems to suffer from the problem of taking the abstraction analysis too literally. The *Lotus* court rejected the *Altai* test on the grounds that

> [w]hile the Altai test may provide a useful framework for assessing the alleged non-literal copying of computer code, we find it to be of little help in assessing whether the literal copying of a menu command hierarchy constitutes copyright infringement. In fact, we think that the Altai test in this context may actually be misleading because, in instructing courts to abstract the various levels, it seems to encourage them to find a base level that includes copyrightable subject matter that, if literally copied, would make the copier liable for copyright infringement . . . . We think that abstracting menu command hierarchies down to their individual word and menu levels and then filtering idea from expression at that stage, as both the Altai and the district court tests require, obscures the more fundamental question of whether a menu command hierarchy can be copyrighted at all.[125]

This argument misunderstands the *Altai* test. First, it is simply wrong to speak of "abstracting menu command hierarchies down to their individual word and menu levels." Indeed, it is really wrong to speak of what Borland did as "literal copying" of the menu hierarchy. Lotus distributed object code which, in certain combinations, produces physical images on a screen which represent certain words and which have certain effects. Borland did not copy that code; rather, it wrote its own code which produced similar images and had the same effects as the Lotus code. This may or may not be copyright infringement. It is not "literal" copying of the Lotus program any more than an accused book which uses the same organizational scheme as a copyrighted work, but different text in each chapter, has "literally" copied the original book. A proper application of the *Altai* approach in this context would identify the menu command hierarchy as the level of abstraction at which copying was alleged, and would then proceed to decide what *if anything* was copyrightable at that level.[126]

Of course, this is in effect precisely what the *Lotus* court did-although when it reached the "filtration" step, it determined that the entire program was unprotectable at the menu command hierarchy level. Thus, both the result and even the approach in *Lotus* are entirely consistent with the abstraction-filtration-comparison test. Unfortunately, however, the *Lotus* court's misinterpretation of that test led it to declare that it was taking a different approach.

B. Filtration and the Role of Software Patents

Having identified the level or levels of abstraction at which copying is alleged, the next step in applying the *Altai* test is for the court to "filter out" those parts of the program which are not copyrightable at that level of abstraction. There has not been serious judicial dispute over how to apply this part of the test, although there is certainly a political debate over what and how much material should be filtered out. The *Lotus* decision, however, does raise two issues of importance in conducting the filtration inquiry. First, the majority applies the "method of operation" limitation in section 102(b) to find the Lotus 1-2-3 menu command hierarchy uncopyrightable, in effect "filtering out" the entire program at that level. Second, Judge Boudin's concurring opinion focuses on the question of whether software ought to be protected by copyright law or by patent law. As it turns out, these two issues are related. To understand how they are related requires a brief digression into the history of software patents.

In the early days of computer software, program owners seeking legal protection had very few options. Computer programs were protectable either under copyright law (in accordance with the CONTU report) or under trade secret law. Problems inherent in trade secret protection,[127] however, left copyright the only realistic way for most companies to protect their programs.

Traditionally, patent protection has not been available for computer programs. Patent protection was rejected by the Supreme Court in *Gottschalk v. Benson* on the grounds that a computer program merely recites a mathematical algorithm,[128] and that mathematical algorithms are laws of nature that do not constitute patentable subject matter.[129] Between 1972 and 1981, during the "formative years" of intellectual property protection for computer software, therefore, software designers had to reckon without the possibility of patent protection for their inventions.[130]

In 1981, the Supreme Court decided the case of *Diamond v. Diehr.*[131] That case did not reverse *Benson*, but it did adopt an intermediate position with respect to the patenting of computer programs. *Diehr* held that a process was not unpatentable merely because it recited or applied a mathematical algorithm as one of its steps. Rather, the relevant question for the Court was whether the patent claimed *nothing more than* the algorithm, or whether it added other steps to the process (or, in the case of an apparatus claim, whether the claim was limited to the implementation of the algorithm in a particular physical structure).[132] Significantly, the *Diehr* Court indicated that a process or apparatus claim was patentable even though the only inventive element of the claim was the algorithm, as long as some physical structure or process steps were included.[133]

The effect of *Diehr* was to exalt artful drafting above the substance of the patented invention. A new mathematical algorithm was unpatentable if it was claimed as a mathematical algorithm, but the same algorithm could be patented by attorneys smart enough to include the proper claim elements in their application.[134] By the late 1980s, the rush to patent software was on.

Today, it seems reasonably clear that software is patentable. The formalistic rules required by *Diehr* have been progressively relaxed, allowing the patenting of algorithms with only the barest tie to physical structure or process.[135] The Patent Office has issued thousands of software patents, and continues to issue more each year.[136] The Patent Office has even granted patents on bare mathematical algorithms themselves.[137] The only remaining disputes primarily concern the form software patent claims must take.[138] And even those disputes may soon end, since GATT requires that member nations issue patents without discriminating against particular technologies.[139]

What does the availability of software patents mean for software copyright law? There is no question that patent and copyright law can coexist in the software context.[140] In the current legal environment, a savvy lawyer will no doubt seek both patent and copyright protection for an important program. But the overlap between patent and copyright protection highlights the importance of the idea-expression dichotomy in copyright law, which serves to distinguish copyright from patent. As the Federal Circuit explained in *Atari*:

> In addition, copyright protection does not "extend to any . . . procedure, process, system [or] method of operation." 17 U.S.C. section 102(b). In conformance with the standards of patent law, title 35 provides protection for the process or method performed by a computer in accordance with a program. Thus, patent and copyright laws protect distinct aspects of a computer program. Title 35 protects the process or method performed by a computer program; title 17 protects the expression of that process or method. While title 35 protects any novel, nonobvious, and useful process, title 17 can protect a multitude of expressions that implement that process. If the patentable process is embodied inextricably in the line-by-line instructions of the computer program, however, then the process merges with the expression and precludes copyright protection.[141]

The *Atari* court's final point is critical-the scope of expression protectable under copyright law varies with the scope of protection afforded ideas under patent law. In theory, this should not be true. The patent and copyright statutes are independent, and there is no reason to believe that changes in one statute mean changes in the other. But in practice, software copyright law has been protecting "patent-like" inventions for some time. The rules of section 102(b)-against copyrighting ideas, processes, systems, etc.-have not been strictly enforced in the software context, because as a practical matter no other form of protection for the ideas and processes of a computer program was available.[142]

Perhaps this was a bad idea from the outset. Certainly, many have argued persuasively that copyright has never been suited to protect computer software, offering numerous examples of the poor "fit" between the two.[143] Further, as Judge Boudin noted in his concurrence in *Lotus v. Borland*, "[i]t is no accident that patent protection has preconditions that copyright protection does not-notably, the requirements of novelty and non-obviousness-and that patents are granted for a shorter period than copyrights."[144] Altering copyright law rather than employing patent protection has arguably overprotected computer software, because copyright contains none of the safeguards of patent law's validity rules and examination procedures.

It doesn't really matter whether this expansion of copyright law was wise or not, however. As software patents gain increasingly broad protection,[145] whatever reasons there once were for broad copyright protection of computer programs disappear. Much of what has been considered the copyrightable "structure, sequence and organization" of a computer program will become a mere incident to the patentable idea of the program or of one of its potentially patentable subroutines. Broad copyright protection of the sort afforded in *Whelan* will become both unnecessary to software owners and problematic from the standpoint of intellectual property theory.

Arguably, the decision in *Lotus v. Borland* is merely an application of this principle. The court's decision adds a new item-"methods of operation"-to the list of things which must be filtered out before comparing the copyrighted and accused works. This decision is consistent with the idea suggested above: that courts have in the past been giving some copyright protection to ideas and functional program elements, but that they should cease doing so now that programs are eligible for patent protection.[146] Indeed, Judge Boudin's concurrence explicitly focuses on the patent-copyright distinction, arguing that the structure, sequence and organization of computer programs are particularly suited to patent rather than traditional copyright protection.

The availability of the patent option affects virtually all cases involving non-literal infringement or the copyright protection of program elements at a high level of abstraction. In these cases, the existence of software patents should make courts less willing to extend the coverage of copyright law to ideas and the functional elements of programs, and more willing to engage in a strict filtration analysis.

C. Comparison: Sliding Scales and Virtual Identity

The third step in evaluating a claim of copyright infringement is to compare the copyrightable elements of the program at the relevant level of abstraction (if any) with the corresponding elements in the accused program. The law in this area is underdeveloped. Two important factors in this analysis are the nature and amount of the copying that has occurred. "Copying" software can mean many different things. It might mean duplication of the object code of a program-the production of a true physical copy of the program. It might mean copying of source code to create a program that runs on a different operating system. It might mean copying of the basic instruction sets used by the original program. It might mean copying one or more subroutines from the program, or writing a program with a similar structure, or even writing a program to do the same thing as the original.[147]

To complicate matters further, *how much* is copied may vary along with the level of abstraction at which it is copied. Thus, an accused infringer may copy material verbatim, creating an identical work. Alternatively, he may make "immaterial variations" in the work, producing a virtually but not precisely identical copy. More significant differences between the two works may result in a "copy" which is substantially similar to the original, but which also contains nontrivial differences from the original program. Finally, an accused infringer may "copy" small pieces of a work as part of a wholly different work, creating minor similarities between otherwise different works. Different types of copying are represented on the axes of Figure 1. On the vertical axis, I have presented a variant of the "levels of abstraction" identified by *Altai* and *Gates Rubber* for use in software cases. On the horizontal axis, I have presented a rough measure of the amount of copying that has occurred.

None of these gradations are captured by the *Altai* filtration test. This is not because the filtration test is flawed, but because the test is largely directed at identifying protectable expression (the copyrightability portion of the infringement analysis), not at comparing a copyrighted work to an accused work. When the comparison step is considered, it becomes evident that the nature of the copying matters.

First, the level of abstraction at which the copying occurs is important, because computer programs generally receive more copyright protection at low levels of abstraction than at high levels. Other things being equal, a finding of copyright infringement is more likely against a company that copies *x* percent of the object code of a program than against a company that copies the same percentage of the basic structure of the program. This is true because object code is more likely to contain protectable expression than is basic program structure.

Second, it also matters how similar the two programs are. If the accused program is identical to the copyrighted program, it is almost certainly infringing, since if the copyrighted program contains *any* protectable expression, that expression has been copied. By contrast, if there are only minor similarities between the two programs, it is much less likely that a court will find infringement, since minor similarities are more often accidental or the result of unprotectable common elements.

These two principles interact. Identical copying of object code is unquestionably copyright infringement, except perhaps in the most unusual case.[148] By contrast, minor similarities between the purpose or basic structure of two programs almost certainly do not constitute copyright infringement. Between these extremes, analysis of copying must take place on a sliding scale: the higher the level of abstraction at which similarities are found, the closer the works must be before a court should find copyright infringement. I have depicted this sliding scale in a rough, stylized way as the diagonal line in Figure 1 located in the appendix at page 34.

This sliding-scale, "organic" analysis of copying finds significant support in other areas of copyright law. In *Hoehling v. Universal City Studios*,[149] for example, Hoehling was the author of a non-fiction book analyzing the destruction of the German zeppelin "Hindenburg" by fire in New Jersey in 1938. He advanced a particular theory to explain the Hindenburg explosion-that the Hindenburg was sabotaged by a crewman on board. When Universal made a movie of the explosion which featured a crewman-saboteur, Hoehling sued for copyright infringement.

The Second Circuit rejected Hoehling's claim:

> [W]here, as here, the idea at issue is an interpretation of an historical event, our cases hold that such interpretations are not copyrightable as a matter of law. In *Rosemont Enterprises, Inc. v. Random House, Inc.*,[150] we held that the defendant's biography of Howard Hughes did not infringe an earlier biography of the reclusive alleged billionaire. Although the plots of the two works were necessarily similar, there could be no infringement because of the "public benefit in encouraging the development of historical and biographical works and their public distribution." To avoid a chilling effect on authors who contemplate tackling an historical issue or event, broad latitude must be granted to subsequent authors who make use of historical subject matter, including theories or plots.[151]

The court went on to note a caveat, however:

> We are aware, however, that in distinguishing between themes, facts, and *scenes a faire* on the one hand and copyrightable expression on the other, courts may lose sight of the forest for the trees. By factoring out similarities based on non-copyrightable elements, a court runs the risk of overlooking wholesale usurpation of a prior author's expression. A verbatim reproduction of another work, of course, even in the realm of nonfiction, is actionable as copyright infringement. Thus, in granting or reviewing a grant of summary judgment for defendants, courts should assure themselves that the works before them are not virtually identical. In this case, it is clear that all three authors relate the story of the Hindenburg differently.[152]

Other courts have adopted the "virtual identity" standard as well, in the artistic as well as the factual context.[153]

Hoehling's two lessons-that we must not "lose sight of the forest for the trees," and that virtual identity may constitute copyright infringement even in circumstances in which substantial similarity does not-have direct applicability to the software context. Indeed, several courts have adopted a similar approach in applying the filtration test.

The most obvious example is the Ninth Circuit's decision in *Apple Computer v. Microsoft*.[154] There, the court engaged in abstraction and filtration of the graphical user interfaces in copyrighted and accused programs. After filtering licensed and other unprotectable elements out of Apple' Macintosh GUI, the court found that only a few isolated parts of Apple's interface were subject to copyright protection. The court also agreed that Apple was entitled to protect its "unique selection and arrangement" of its (largely uncopyrightable) features.[155]

However, the claim that Microsoft and Hewlett-Packard copied the *arrangement* of icons and menu items works at a higher level of abstraction than the claim that they copied the icons themselves. Rather than individual program elements, Apple is now claiming that the overall structure of the GUI has been copied. Because the alleged infringement occurs at a higher level of abstraction, the court held that it is not enough to show that the two interfaces were arranged in similar ways: "[w]hen the range of protectable and unauthorized expression is narrow, the appropriate standard for illicit copying is virtual identity."[156] Other courts have similarly acknowledged that copyright protection for computer programs must be judged on a sliding scale, with the degree of similarity required for infringement varying according to the level of abstraction at which copying is alleged to have occurred.[157]

The filtration of unprotectable elements out of a computer program is only part of the analysis in a copyright infringement case. Comparison of the two programs necessarily requires a detailed inquiry into how much copying occurred, and at what level of abstraction. Where the alleged similarities occurred at a high level of abstraction, at which the copyright owner is entitled to little protection, she must be able to show more striking similarities between the works than if the copying had occurred at a lower level of abstraction.

D. Justifications for Copying

Proof of copying under the abstraction-filtration-comparison test does not end the copyright analysis. There is a final step, in which the court must determine whether the copying was justified under one of several provisions giving rights to users of copyrighted programs. The copyright fair use doctrine has increasingly been interpreted to give accused infringers the right to engage in some copying of computer programs for a socially useful purpose. In particular, most courts to consider the issue have permitted the copying of software to the extent necessary to ensure compatibility or interoperability between computer programs, although the issue is not yet completely settled.[158] Similarly, section 117 of the Copyright Act gives users of computer programs the right to make certain copies and derivative works,[159] although some courts have tended to read section 117 fairly narrowly.[160]

The existence of these "justifications for copying" does not relate directly to the determination of copyrightability or infringement. However, because they affect the circumstances under and frequency with which copyright defendants will be held liable, these doctrines may interact with the determination of copyrightability in a somewhat surprising way. Courts may be more willing to confer copyright protection on programs if that protection does not foreclose copying under all circumstances. Judge Boudin's concurrence in *Lotus v. Borland* is quite explicit in identifying this tradeoff.[161]

Indeed, the tradeoff is a particularly important one in the *Lotus* case. The majority in *Lotus* decided that the menu command hierarchy of the 1-2-3 program was not copyrightable at all. This means that if Borland "had simply copied the Lotus menu (using different code []), contributed nothing of its own, and resold Lotus under the Borland label,"[162] it would still be off the hook under the majority's analysis. By contrast, the alternative suggested by Judge Boudin-that "Borland's use is privileged because . . . it is not seeking to appropriate the advances made by Lotus' menu"[163]-would protect Borland only in situations where it had contributed something of public value, or where it had not appropriated value from Lotus. The result in the actual case would be the same under this "privileged use" rationale, but Lotus would still be able to protect its menu structure against those who merely sought to counterfeit it.

V. Conclusions

For some time, judicial and academic debate on software copyright law has focused on selecting the appropriate test for copyrightability of computer programs. Today, this debate has largely been resolved. Since the adoption of the abstraction-filtration-comparison test by the Second Circuit in 1992, most courts to consider the issue have lined up behind this approach. However, this convergence on a legal standard for software copyright law has not produced corresponding agreement among the courts in the outcomes of copyright infringement cases. Rather, the terms of the debate seem to have changed.

In this Article, I suggest a way to apply the abstraction-filtration-comparison test which should help courts make consistent decisions regarding infringement. First, the abstraction test should be used only as an analytic guide, and not as a representation of reality. Viewed as a guide, it is possible to reconcile even cases such as *Lotus v. Borland* with the *Altai* approach to infringement. Second, the filtration analysis must take account of software patents. The ready availability of software patents means that broad copyright protection for the structural and functional elements of computer software is less important today than it was ten years ago. It also means that courts should be careful to respect the idea-expression dichotomy in applying the filtration test. Third, copying is not a unitary phenomenon. In comparing two works, courts should be sensitive to both the extent of the similarity between the works, and the level of abstraction at which the similarity is tested. To account for these factors, courts should adopt a sliding-scale approach to copyright infringement. Finally, a finding of infringement is not necessarily the end of the story. Copyright fair use doctrine provides a number of possible justifications for copying that courts should consider closely.

Application of these principles will help ensure consistent judicial decision making, as well as a copyright doctrine which is both fair and consistent with the rest of intellectual property law.

† Assistant Professor, University of Texas School of Law. I would like to thank Rose Hagan for her advice, and Lisa Launer at HTLJ for her useful suggestions.

1. Numerous articles have tackled these issues. Among them are: David Bender, Computer Associates v. Altai: *Rationality Prevails*, COMPUTER LAW**.**, Aug. 1992, at 1; Jack E. Brown, *"Analytical Dissection" of Copyrighted Computer Software-Complicating the Simple and Confounding the Complex*, 25 ARIZ. ST. L.J. 801 (1993); Anthony L. Clapes & Jennifer M. Daniels, *Revenge of the Luddites: A Closer Look at* Computer Associates v. Altai, COMPUTER LAW**.**, Nov. 1992, at 11; Douglas Derwin, *It is Time to Put "Look and Feel" out to Pasture*, 15 HASTINGS COMM. & ENT. L.J. 605 (1993); Audrey F. Dickey, Computer Associates v. Altai *and* Apple v. Microsoft*: Two Steps Back from* Whelan*?*, 8 SANTA CLARA COMPUTER & HIGH TECH. L.J. 379 (1993); Donald F. McGahn, *Copyright Infringement of Protected Computer Software: An Analytical Method to Determine Substantial Similarity*, 21 RUTGERS COMPUTER & TECH. L.J. 88 (1995); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1074, 1082 (1989); Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV. 977, 1004-05 (1993); J.H. Reichman, *Electronic Information Tools: The Outer Edge of World Intellectual Property Law*, 17 U. DAYTON L. REV. 797 (1992); Jack Russo & Jamie Nafziger, *Software "Look and Feel" Protection in the 1990s*, 15 HASTINGS COMM. & ENT. L.J. 571 (1993); Symposium, *Has Look and Feel Crashed?*, 11 CARDOZO ARTS & ENT. L.J. 721 (1993); Julian Velasco, *The Copyrightability of Nonliteral Elements of Computer Programs*, 94 COLUM. L. REV. 242 (1994); Aram Dobalian, Comment, *Copyright Protection for the Non-Literal Elements of Computer Programs: The Need for Compulsory Licensing*, 15 WHITTIER L. REV. 1019 (1994); Stephen H. Eland, Note, *The Abstraction-Filtration Test: Determining Non-Literal Copyright Protection for Software*, 39 VILL. L. REV. 665 (1994); 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, 881 (1990); Thomas M. Gage, Note, Whelan Associates v. Jaslow Dental Laboratories: *Copyright Protection for Computer Software Structure-What's the Purpose?*, 1987 WISC. L. REV. 859, 860-61; W. H. Baird Garrett, Note, *Toward a Restrictive View of Copyright Protection for Nonliteral Elements of Computer Programs: Recent Developments in the Federal Courts*, 79 VA. L. REV. 2091 (1993); Mark T. Kretschmer, Note, *Copyright Protection for Software Architecture: Just Say No!*, 3 COLUM. BUS. L. REV. 823, 837-39 (1988); David A. Lowe, Comment, *A Square Peg in a Round Hole: The Proper Substantial Similarity Test for Nonliteral Aspects of Computer Programs*, 68 WASH. L. REV. 351 (1993); Adam E. McKinney, Note, *Copyright Protection for Functional Works: Where Does the Fifth Circuit Draw the Line Between Idea and Expression?*, 47 BAYLOR L. REV. 249 (1995); Peter G. Spivack, Comment, *Does Form Follow Function? The Idea/Expression Dichotomy in Copyright Protection of Computer Software*, 35 UCLA L. REV. 723, 747-55 (1988); Jon S. Wilkins, Note, *Protecting Computer Programs as Compilations Under* Computer Associates v. Altai, 104 YALE L.J. 435 (1994).

2. *See* discussion *infra* section III.A.

3. 982 F.2d 693 (2d Cir. 1992). For convenience, I will periodically refer to this as the "filtration" or "dissection" approach.

4. Indeed, the one court which has rejected *Altai*'s filtration approach did so because it thought the protection afforded by *Altai* was too broad, rather than too narrow. *See* Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 815 (1st Cir. 1995).

5. *See* discussion *infra* section III.B.

6. 49 F.3d at 807.

7. 35 F.3d 1435 (9th Cir. 1994).

8. To be sure, it is possible to "reverse engineer" object code in some cases to create a kind of rough estimate of what must have been in the original source code. The process, however, is demanding and time-consuming even for expert programmers. *See, e.g.*, Andrew Johnson-Laird, *Reverse Engineering of Software: Separating Legal Mythology from Actual Technology*, 5 SOFTWARE L.J. 331 (1992).

9. Data General Corp. v. Digital Computer Controls, Inc., 297 A.2d 433, 436 (Del. Ch. 1971) (program with 500 copies sold still qualifies as a trade secret), *aff'd*, 297 A.2d 437 (Del. 1972). Courts continue to reach this result. Data General Corp. v. Grumman Systems Support Corp., 825 F. Supp. 340, 354-55 (D. Mass. 1993); Management Science of Am. v. Cyborg Systems, Inc., 1977-1 TRADE CAS. ¶ 61472 (N.D. Ill. 1977).

10. Most software contracts at this time contained proprietary rights provisions which asserted that the information contained in the accompanying computer software was proprietary to the vendor, and could not be copied or disclosed without the vendor's permission.

11. On the shortcomings of trade secret protection, *see* discussion *infra* note 127.

12. *See* ROBERT MERGES, PETER MENELL, MARK LEMLEY, & THOMAS JORDE, INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE ch. IV, at 1-3 (discussing historical development of copyright law).

13. *See* 17 U.S.C. § 106(1) (1995).

14. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

15. *See* Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157, 1166-67 (9th Cir. 1977).

16. *Id.* at 1167.

17. Gaste v. Kaiserman, 863 F.2d 1061, 1066 (2d Cir. 1988).

18. *See* National Comics Pubs. v. Fawcett Pubs., 191 F.2d 594, 600 (2d Cir. 1951) ("[N]o one infringes, unless he descends so far into what is concrete as to invade . . . [its] expression.").

19. *See* Baker v. Selden, 101 U.S. 99 (1879). Section 102(b) of the Copyright Act provides that "[i]n 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." 17 U.S.C. § 102(b) (1995).

20. Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980).

21. Nichols v. Universal Pictures Crop., 45 F.2d 119, 121 (2d Cir. 1930).

22. Feist Publication v. Rural Telephone Service, 111 S. Ct. 1282 (1991).

23. *See* Jarvis v. A & M Records, 827 F. Supp. 282, 288 (D.N.J. 1993).

24. *See, e.g.*, Apple Computer v. Franklin Computer, 545 F. Supp. 812 (E.D. Pa. 1982), *rev'd*, 714 F.2d 1240 (3d Cir. 1983).

25. *See, e.g.*, H.R. Rep. No. 1307, 96th Cong., 2d Sess. 23 (1980) U.S.C.A.A.N. 6460, 6482. 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 the users of computer programs. CONTU did not, however, add computer programs to the list of protectable works, instead considering them to be both literary and audiovisual works.

26. Williams Electronics v. Artic Int'l, 685 F.2d 870, 876-77 (3d Cir. 1982).

27. *Franklin Computer*, 714 F.2d at 1247-49.

28. NEC Corp. v. Intel Corp., 10 U.S.P.Q.2d 1177, 1178-80 (N.D. Cal. 1989).

29. *Williams Electronics*, 685 F.2d at 870.

30. *Franklin Computer*, 714 F.2d at 1240.

31. Stern Electronics v. Kaufman, 669 F.2d 852, 857 (2d Cir. 1982); *see also* Manufacturers Technologies, Inc. v. Cams, Inc., 706 F. Supp. 984, 992-96 (D. Conn. 1989).

32. Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960) ("Obviously, no principle can be stated as to when an imitator has gone beyond copying the 'idea', and has borrowed its 'expression'. Decisions must therefore inevitably be *ad hoc*."); *see also* Nichols v. Universal Pictures Corp., 45 F.2d 119 (2d Cir. 1930) (discussing levels of abstraction at which copyrightability may be tested).

33. *See, e.g.*, Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1074, 1082 (1989).

34. 462 F. Supp. 1003 (N.D. Tex. 1978).

35. *Id.* at 1012-14.

36. *Id.* at 1014 (emphasis in original).

37. 807 F.2d 1256 (5th Cir. 1987).

38. *Plains Cotton*, 807 F.2d at 1262. The court also refused to follow the Third Circuit's decision granting broad protection to non-literal elements in Whelan Assoc. Inc. v. Jaslow Dental Laboratory Inc., 797 F.2d 1222 (3d Cir. 1986), discussed *infra* notes 41-57 and accompanying text.

39. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

40. *See e.g.*, Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp., 562 F.2d 1157 (9th Cir. 1977).

41. 797 F.2d 1222 (3d Cir. 1986).

42. *Id.* at 1234.

43. *Id.* at 1240.

44. While acknowledging the difficulty of such an undertaking, and Judge Hand's skepticism that the task could be accomplished at all, *see* Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960), the Third Circuit believed that a review of copyright law would "enable [them] to formulate a rule applicable in this case." *Whelan*, 797 F.2d at 1235.

45. *Whelan*, 797 F.2d at 1236 (emphasis omitted).

46. *Id.* at 1237. *Whelan*'s emphasis on allowing computer programmers to recoup their investment in writing the program is of questionable vitality in light of the Supreme Court's rejection of the "sweat of the brow" doctrine in Feist Publication Inc. v. Rural Telephone Service Co, Inc., 499 U.S. 340, 360 (1991). In that case, the Supreme Court made it clear that the fact that a copyright plaintiff had invested substantial effort in developing a work did not automatically entitle the plaintiff to copyright protection. Investment of time and energy could not convert unprotectable elements such as facts and ideas into protectable expression. Similarly, under *Feist*, the fact that programmers invest substantial time and effort in developing the ideas or functional aspects of a program ought not give them control over those ideas or functional aspects.

47. *Whelan*, 797 F.2d at 1236 n.28.

48. *Id*. at 1236.

49. *Id*. at 1239-40.

50. *See, e.g.*, Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37, 63 (D. Mass. 1990) (protecting Lotus 1-2-3 user interface); Manufacturers Technologies v. Cams, Inc., 706 F. Supp. 984, 993 (D. Conn. 1989) (protecting user interface); Digital Communications Assocs., Inc. v. Softklone Distrib. Corp., 659 F. Supp. 449, 455-56 (N.D. Ga. 1987) (protecting "total concept and feel" of program, but rejecting protection for screen displays); Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127,

1133 (N.D. Cal. 1986) (protecting the "overall structure" of the program); SAS Inst. Inc. v. S & H Computer Systems, Inc., 605 F. Supp. 816, 829-30 (M.D. Tenn. 1985) (copying of program structure constitutes copyright infringement).

Many of the cases which accepted *Whelan*'s first premise-that the non-literal elements of computer programs were protectable-did not go on to find infringement, however. These cases suggested a much narrower approach to copyright protection, based on the functional nature of most software. *See, e.g.*, Telemarketing Resources v. Symantec Corp., 12 U.S.P.Q.2d 1991 (N.D. Cal. 1989) (no infringement of screen displays); Q-Co Industries, Inc. v. Hoffman, 625 F. Supp. 608, 615-16 (S.D.N.Y. 1985) (similarities between program structures were dictated by function, and did not constitute copyright infringement).

51. *See* Kepner-Tregoe, Inc. v. Leadership Software, Inc., 12 F.3d 527, 536 n.20 (5th Cir. 1994).

52. *See, e.g.*, 3 MELVILLE NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.03(F), at 13-62.34; Douglas K. Derwin, *It Is Time to Put "Look and Feel" Out to Pasture*, 15 HASTINGS COMM & ENT. L.J. 605 (1993); Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1074, 1082-83 (1989); 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, 881-82 (1990); Thomas M. Gage, Note, Whelan Associates v. Jaslow Dental Laboratories: *Copyright Protection for Computer Software Structure-What's the Purpose?*, 1987 WIS. L. REV. 859, 860-61; W.H. Baird Garrett, Note, *Toward a Restrictive View of Copyright Protection for Nonliteral Elements of Computer Programs: Recent Developments in the Federal Courts*, 79 VA. L. REV. 2091 (1993); Mark T. Kretschmer, Note, *Copyright Protection for Software Architecture: Just Say No!*, 1988 COLUM. BUS. L. REV. 823, 836-39; Peter G. Spivack, Comment, *Does Form Follow Function?: The Idea/Expression Dichotomy in Copyright Protection of Computer Software*, 35 UCLA L. REV. 723, 747-55 (1988). The American Committee for Interoperable Systems, an organization of computer companies opposed to broad copyright protection, filed an amicus brief in Computer Associates v. Altai critical of the *Whelan* approach. *See* Brief Amicus Curiae of American Committee for Interoperable Systems, Computer Associates v. Altai, 982 F.2d 693 (2d Cir. 1992) (No. 91-7893).

In addition to these contemporaneous sources, numerous commentators writing after the Second Circuit's decision in Computer Assoc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992), have endorsed that decision and rejected *Whelan*. For a partial list of these commentators, *see infra* note 78.

Indeed, even some advocates of broad protectionism have failed to endorse *Whelan*, opting instead for other approaches. *See, e.g.*, Jon S. Wilkins, Note, *Protecting Computer Programs as Compilations Under* Computer Associates v. Altai, 104 YALE L.J. 435 (1994) (suggesting that computer programs might be protected as original compilations of unprotectable elements). An argument such as Wilkins' in effect rejects the *Whelan* approach, since it acknowledges the wisdom of evaluating individual elements for copyrightability.

53. *See, e.g.*, Gates Rubber Co. v. Bando Chem. Indus. Ltd., 9 F.3d 823, 836-38 (10th Cir. 1993). For a more detailed discussion of these issues, *see infra* section III.A.

54. *See Whelan*, 797 F.2d at 1236-37 (acknowledging the rule that scenes a faire and facts are unprotectable, but concluding that this is true only when the facts or the scenes a faire "merge" with the single idea of the program).

55. *Id*.

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

57. *Whelan*, 797 F.2d at 1238-39. Of course, it does not follow from the argument that the protection granted in *Whelan* was too broad that the *result* in that case was necessarily incorrect. It may well be that, had the court defined the idea more narrowly, it would still have found the programs substantially similar in their copyrighted expression.

58. 982 F.2d 693 (2d Cir. 1992).

59. *Altai*, 982 F.2d at 699-700. On appeal, Altai did not contest the district court's finding that its old program infringed CA's copyright, or its liability for damages for this past infringement. *See id*. at 701.

60. *Id*. at 702-03.

61. *Id*. at 703 (emphasis added).

62. *Id*. at 706.

63. *Id*. at 706-711.

64. *See* Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1475 (9th Cir. 1990) (endorsing "analytic dissection" of computer programs).

65. 45 F.2d 119, 121 (2d Cir. 1930).

66. *See generally* Englund, *supra* note 1, at 897-98.

67. *Altai*, 982 F.2d at 707-10. It is worth noting that *Altai* does not simply choose one level of abstraction as appropriate. Rather, it evaluates the program at *each* level of abstraction in order to identify any protectable expression residing at that level. At the same time, as Judge Hand noted in *Nichols*, "there is a point in this series of abstractions where they are no longer protected, since otherwise the [author] could prevent the use of his 'ideas.'" 45 F.2d at 121.

68. *Altai*, 982 F.2d at 707-08; *see also* Menell, *supra* note 1, at 1052 (copyright protection does not extend to portions of a program dictated by efficiency).

69. *Altai*, 982 F.2d at 709-10. The five factors are: "(1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs which with a program is designed to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry." *Id.*, citing 3 NIMMER ON COPYRIGHT, *supra* note 52, at 13-66 to 13-71.

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

71. *Id*. The court also refers to this core expression as the "golden nugget." *Id.*

72. *Id.* at 710-11.

73. 111 S. Ct. 1282 (1991).

74. *Altai*, 982 F.2d at 711.

75. *Id.*

76. Clapes, *supra* note 1, at 11.

77. Brown, *supra* note 1, at 801.

78. In addition, the weight of academic commentary on *Altai* has been favorable as well. *See, e.g.*, Brief Amicus Curiae of Copyright Law Professors, Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807 (1st Cir. 1995) No. 93-2214 (24 copyright law professors endorsing *Altai* decision), *reprinted in* 16 HASTINGS COMM. & ENT L.J. 657 (1994); Bender, *supra* note 1, at 1; Derwin, *supra* note 1, at 605; McGahn, *supra* note 1, at 131; Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 COLUM. L. REV. 2644 (1994); Reichman, *supra* note 1, at 831; Velasco, *supra* note 1, at 242; Dobalian, *supra* note 1, at 1073; Eland, *supra* note 1, at 699.

79. In addition to *Altai* itself, see Softel v. Dragon Medical, 1992 WL 168190 (S.D.N.Y. 1992).

80. Atari Games Corp. v. Nintendo of Am., 975 F.2d 832, 839 (Fed. Cir. 1992).

81. 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).

82. Apple Computer v. Microsoft Corp., 35 F.3d 1435, 1442-43 (9th Cir. 1994).

83. 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).

84. Mitek Holdings v. Arce Eng. Co., 864 F.Supp. 1568, 1577-1578 (M.D. Fla. 1994); CMAX/Cleveland, Inc. v. UCR, Inc., 804 F. Supp. 337, 352-54 (M.D. Ga. 1992).

85. Delrina Corp. v. Triolet Sys., 9 B.L.R.2d 140 (Ont. Ct. Just. 1993); Matrox Elect.. Sys. v. Gaudreau, 1993 R.J.Q. 2449, 2457-58 (Mont. Ct. Just. July 28, 1993).

86. John Richardson Computers v. Flanders & Chemrec Ltd., [1993] 497 (Ct. Chanc. Feb. 19, 1993).

87. Computer Assoc. Int'l v. S.A.R.L. Faster, No. 519/95, reported in COMP. L. RPTR., May 1995.

88. Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 815 (1st Cir. 1995).

89. Gates Rubber v. Bando Chem. Indus., 9 F.3d 823, 834 (10th Cir. 1993).

90. *Id*. at 835.

91. *Id*. at 836. The court noted that the basic idea of a program, and the individual modules themselves, will almost never be protected by copyright law. *Id*.

92. *Id*. at 836-37. The court cited the legislative history accompanying section 102(b) of the Copyright Act, which clearly indicates that the actual processes or methods used in a computer program are not entitled to copyright protection. *Id.* The court noted that processes were most commonly found in the program architecture, module structure, and algorithms used. *Id*. at 837.

93. *Id*. Facts are most commonly found in a program's data structures, or in the actual code used to implement a particular process. *Id*.

94. *Id*. at 837-38.

95. *Id.* at 838.

96. *Id*. This is the "scenes a faire" doctrine. The court held that such standard devices in computer programs include "those elements of a program that have been dictated by external factors," such as hardware and software standards, specifications, or compatibility requirements, customer design specifications, and basic industry practices. *Id*.

97. *Id*. at 839.

98. *Id*. at 839 n.15.

99. *See*, *e.g.*, Engineering Dynamics v. Structural Software Inc., 26 F.3d 1335, 1342-43 (5th Cir. 1994) (adopting the Tenth Circuit approach explicitly); *cf.* Apple v. Microsoft, 35 F.3d at 1443 (applying the Ninth Circuit's own version of the filtration test, called "analytic dissection," relying on Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1475 (9th Cir. 1992), *cert. denied*, 113 S. Ct. 198 (1992)).

100. *Engineering Dynamics*, 26 F.3d at 1343.

101. *See infra* notes 108-111 and accompanying text.

102. Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 815 (1st Cir. 1995).

103. Computer Associates Int'l v. Altai, Inc., 982 F.2d 693, 714 (2d Cir. 1992).

104. *Id.* at 714-15.

105. Gates Rubber v. Bando Chem. Indus., 9 F.3d 823, 842-46 (10th Cir. 1993).

106. Apple Computer v. Microsoft Corp., 35 F.3d 1435, 1446-47 (9th Cir. 1994). In *Apple*, the court's copyright analysis was complicated by the fact that Apple had licensed many of the features of its Macintosh GUI to Microsoft and Hewlett-Packard. *Id*. at 1439-41. As a result, the Ninth Circuit had no opportunity to consider whether many elements of the GUI were protectable at all.

107. *Id. Cf.* Wilkins, *supra* note 1, at 462-64 (suggesting such a test for protectable compilations, but with an extremely easy threshold for finding infringement).

108. CMAX/Cleveland, Inc. v. UCR, Inc., 804 F. Supp. 337, 354-55 (M.D. Ga. 1992).

109. Kepner-Tregoe v. Leadership Software, 12 F.3d 527, 535-36 (5th Cir. 1994). *See* McKinney, *supra* note 1, at 258-263 (criticizing the Fifth Circuit approach in detail).

110. Atari Games Corp. v. Nintendo of America, 975 F.2d 832, 840 (Fed. Cir. 1992).

111. *Id.*

112. Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 815 (1st Cir. 1995).

113. *Id.*

114. *Compare* Apple v. Microsoft, 35 F.3d at 1447 *and Lotus*, 49 F.3d at 817-18 (interfaces found unprotectable in whole or part) *with Engineering Dynamics*, 26 F.3d at 1347 *and CMAX*, 804 F. Supp. at 355-56 (interfaces found protectable).

115. *Compare Altai*, 982 F.2d at 714-15 (after filtration, virtually no part of the CA-ADAPTER program was protectable) *with Atari*, 975 F.2d at 845 (Nintendo lockout chip contains nonfunctional, protectable expression).

116. *Compare Altai*, 982 F.2d at 707-09 *and Gates Rubber*, 9 F.3d at 836-37 (program structures uncopyrightable to the extent that they are dictated by function) *with Kepner-Tregoe*, 12 F.3d at 536-37 (program structures protectable if multiple ways of structuring the program exist). *See* McKinney, *supra* note 1, at 259 (criticizing *Kepner-Tregoe* for failing to engage in a detailed application of the abstraction test).

117. *See, e.g., Lotus*, 49 F.3d at 817-18; *Altai*, 982 F.2d at 714-15; *Gates Rubber*, 9 F.3d at 842.

118. *See Atari*, 975 F.2d at 845; *CMAX*, 804 F. Supp. at 355-56.

119. On these terms, see Paul Goldstein, *Derivative Rights and Derivative Works in Copyright*, 30 J. COPYRIGHT OFF. SOC'Y 209, 211 (1983).

120 *See* Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 815 (1st Cir. 1995).

121. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

122. *Id.*

123. Actually, *Altai* abstracted the *accused* rather than the copyrighted work. *See Altai*, 982 F.2d at 707. This seems backwards given

that the inquiry in copyright infringement cases should focus on finding protectable expression in the *copyrighted* work.

124. On this initial determination of what is alleged to be copied, *see* Gates Rubber v. Bando Chem. Indus., 9 F.3d 823, 833 (10th Cir. 1993).

125. Lotus Dev. Corp. v. Borland Int'l., 49 F.3d 807, 815 (1st Cir. 1975).

126. The *Lotus* court is wrong to suggest that the abstraction test implies that literal copying is always copyright infringement. In the first place, as noted above, Borland did not engage in literal copying as that term is used in *Altai*. Second, the filtration analysis must be applied at any given level of abstraction, and it is quite possible that the result of that analysis will be that *no* part of the program will be copyrightable at that level. *See, e.g.*, Apple Computer v. Microsoft Corp., 35 F.3d 1435, 1446 (9th Cir. 1994); *Gates Rubber*, 9 F.3d at 842; *Altai*, 982 F.2d at 714-15 (all finding virtually no protectable elements of the copyrighted programs at the particular levels examined).

127. Trade secret protection normally ends when a product is publicly distributed. While some companies have attempted to evade this legal rule on the grounds that the secret is contained in unreadable object code, trade secret protection is problematic for many computer companies because it does not prohibit the reverse engineering of legally-obtained computer disks. Uniform Trade Secrets Act, § 1(4) (1985). In part because of the ability of competitors to reverse engineer a secret product, the Supreme Court has acknowledged the inherent weakness of trade secrecy when compared to other forms of intellectual property protection. Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 489-90 (1974).

128. 409 U.S. 63, 72 (1972).

129. *Id.* at 67. 35 U.S.C. § 101 by its terms contains no such limitation on patenting mathematical algorithms. Indeed, section 100 of the Patent Act would seem to extend patent protection to any "invention or discovery." Nonetheless, a venerable line of cases has denied patent protection to "phenomena of nature, . . . mental processes, and abstract intellectual concepts." *Benson*, 409 U.S. at 63.

130. To be sure, even during this period there were efforts to patent computer programs. *See, e.g.*, Parker v. Flook, 437 U.S. 584 (1978). However, these attempts were generally unsuccessful. *See* Pamela Samuelson, Benson *Revisited: The Case Against Patent Protection for Algorithms and Other Computer-Related Inventions*, 39 EMORY L.J. 1025 (1990).

131. 450 U.S. 175 (1981).

132. *Id.* at 185-188.

133. *Id.*; *see also In re* Iwahashi, 888 F.2d 1370 (Fed. Cir. 1989); *In re* Grams, 888 F.2d 835 (Fed. Cir. 1989); *In re* Abele, 684 F.2d 902 (C.C.P.A. 1982); *see* Samuelson, *supra* note 130, at 1089-90.

134. *Iwahashi* is a good example. In that case, the court held that a mathematical algorithm was patentable because the patent claimed it only as implemented in ROM format. 888 F.2d at 1375. There is no question that the algorithm alone would have been unpatentable; it is unclear today whether implementation in RAM would have been sufficient.

135. *See*, *e.g.*, *id.* at 1375 (implementation in ROM); Arrhythmia Research Technology v. Corazonix Corp., 958 F.2d 1053, 1059 (Fed. Cir. 1992) (invention patentable because it results in the manipulation of physical quantities); *In re* Alappat, 33 F.2d 1526, 1545 (Fed. Cir. 1994) (en banc) (suggesting mathematical formula is patentable if it is capable of being used in a general purpose computer).

Even more recently, the Patent Office has taken the position that any computer-readable memory (including ROM, RAM, magnetic disk drives, and CD-ROMs) are statutory "articles of manufacture" if they "can be used to direct a computer to function in a particular manner." United States Dept. of Com. *Proposed Examination Guidelines for Computer-Implemented Inventions*, PTO Docket no. 95053144-5144-01 (May 31, 1995).

136. By one estimate, there are approximately 14,000 United States patents in existence that cover software. *See* Steve G. Steinberg, *Growth of US Software Patents*, WIRED, January 1995, at 54. And the Patent Office is currently issuing approximately 4,000 new software patents per year. *See* Simson L. Garfinkel, *Patently Absurd*, WIRED, July 1994, at 104, 106.

137. The most commonly cited example is Bracewell's patent on the discrete Bracewell transform, an improvement on Fast Fourier Transforms. United States Patent # 4,646,256 (Feb. 24, 1987).

138. *Compare In re* Trovato, 42 F.3d 1376 (Fed. Cir. 1994) (means-plus-function claim invalid because it did not disclose specific hardware embodiments in the specification) *and In re* Warmerdam, 33 F.3d 1354 (Fed. Cir. 1994) (rejecting process claims directed to manipulation of data structure) *with In re* Lowry, 32 F.3d 1579 (Fed. Cir. 1994) (virtual data "structure" resident in computer memory constitutes patentable subject matter).

139. General Agreement on Tariffs and Trade, Agreement on Trade Related Aspects of Intellectual Property Rights (GATT TRIPS), April 15, 1994, Art. 27(1) ("patents shall be available and patent rights enjoyable without discrimination as to . . . the field of technology").

140. *See, e.g.*, Atari Games Corp. v. Nintendo of Am., 975 F.2d 832, 839 (Fed. Cir. 1992).

141. *Id.* at 839-849 (citations omitted).

142. *See, e.g.*, Lotus Dev. Corp. v. Borland Int'l., 49 F.3d 807, 819 (9th Cir. 1994) (Boudin, J., concurring) ("The problem presented by computer programs is fundamentally different [from that presented by literary works] in one respect. The computer program is a means for causing something to happen; it has a mechanical utility, an instrumental role, in accomplishing the world's work. Granting protection, in other words, can have some of the consequences of patent protection in limiting other people's ability to perform a task in the most efficient manner.").

That this expanded view of copyright law in the software context has become institutionalized is amply demonstrated by the arguments of certain copyright scholars, who decry the recent trend away from broad copyright protection on the grounds that it will fail to reward programming efficiency. *See* Anthony Clapes, *Confessions of an Amicus Curiae: Technophobia, Law and Creativity in the Digital Arts*, 19 U. DAYTON L. REV. 903 (1994); Miller, *supra* note 1, at 1004-05. Copyright law, of course, was never intended to reward efficiency.

143. *See, e.g.*, Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329 (1987); A. Samuel Oddi, *An Uneasier Case for Copyright Than for Patent Protection of Computer Programs*, 72 NEB. L. REV. 351 (1993); Pamela Samuelson et al, *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308 (1994); Mary Mills, Note, *New Technology and the Limits of Copyright Law: An Argument for Finding Alternatives to Copyright Legislation in an Era of Rapid Technological Change*, 65 CHI.-KENT L. REV**.** 307 (1989).

144. *Lotus*, 49 F.3d at 819 (Boudin, J., concurring).

145. While most software patent cases to date have dealt with the issue of patentability under 35 U.S.C. § 101, the language of these cases is relevant to issues of patent scope as well. In particular, software patents written as means-plus-function claims have breadth in direct proportion to the range of hardware implementations that are allowed as "equivalents" to the structure disclosed in the patent specification. As that range of equivalents expands, arguably including virtual data "structures" resident entirely in computer memory, *see In re* Lowry, 32 F.3d 1579, 1583 (Fed. Cir. 1994), software patents will gradually expand to cover the implementation of an idea in any digital form.

The clearest example of this trend is the saga of *In re* Beauregard, 53 F.3d 1583 (Fed. Cir. 1995). IBM sought to patent "software contained on a floppy disk" as an article of manufacture. The Board of Patent Appeals and Interferences originally rejected this claim as directed to nonstatutory subject matter, but later admitted before the Federal Circuit that it now considered such matter statutory. For a description of the history of this case, see Richard H. Stern, *Solving the Algorithm Conundrum: After 1994 The Federal Circuit Patent Law Needs a Radical Algorithmectomy*, 22 AIPLA Q.J. 167, 195-206 (1994).

146. Certainly, the buttons on a VCR (the analogy the *Lotus* court uses to describe the menu command hierarchy, *see Lotus*, 49 F.3d at 817) would be eligible for patent but not copyright protection.

147. *Cf.* Gates Rubber v. Bando Chem. Indus., 9 F.3d 823 (10th Cir. 1993) (discussing the levels of abstraction of computer programs).

148. For example, it is possible that two instructions written in different high-level computer languages will produce identical object

code sequences, since the high-level languages must be translated into machine-readable form, and the instructions to the computer will be the same. In this way, it is possible (though unlikely) that different source code will produce identical object code. Of course, in that case there has not really been "copying" at all, but rather independent creation resulting in striking similarity.

149. 618 F.2d 972 (2d Cir. 1980).

150. 366 F.2d 303 (2d Cir. 1966), *cert. denied*, 385 U.S. 1009 (1967).

151. *Hoehling*, 618 F.2d at 978 (citations omitted).

152. *Id.* at 979-80 (citation omitted).

153. *See, e.g.*, Sid & Marty Krofft Television Prods., Inc., v. McDonald's Corp., 562 F.2d 1157, 1167 (9th Cir. 1977).

154. 35 F.3d 1435 (9th Cir. 1994).

155. *Id*. at 1446.

156. *Id*.

157. *See, e.g.*, Engineering Dynamics v. Structural Software, Inc., 26 F.3d 1335, 1348 (5th Cir. 1994); 3 NIMMER ON COPYRIGHT, *supra* note 52, § 13.03[B][2][b].

158. Courts and commentators favoring copying to achieve compatibility include Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807, 817-18 (1st Cir. 1995); Atari Games Corp. v. Nintendo of Am., 975 F.2d 832, 843-44 (Fed. Cir. 1992); Sega, Inc. v. Accolade, 977 F.2d 1510, 1527-28 (9th Cir. 1992); Vault v. Quaid, 847 F.2d 255, 270 (5th Cir. 1988); JONATHAN BAND & MASANOBU KATOH, INTERFACES ON TRIAL (1995); Julie Cohen, *Reverse Engineering and the Rise of Electronic Vigilantism: Intellectual Property Implications of "Lock-Out" Technologies*, 68 S. CAL. L. REV. 1091 (1995); Dennis S. Karjala, *Copyright Protection of Computer Documents, Reverse Engineering, and Professor Miller*, 19 **U. DAYTON L. REV.** 975, 1016-18 (1994); David A. Rice, *Sega and Beyond: A Beacon for Fair Use Analysis . . . At Least as Far as It Goes*, 19 U. DAYTON L. REV**.** 1131, 1168 (1994).

Some commentators remain opposed to compatibility as a justification for copying, and at least one old court decision rejecting compatibility remains the law of that Circuit. *See* Apple Computer v. Franklin Computer, 714 F.2d 1240, 1253 (3d Cir. 1983); Anthony Clapes, *Confessions of an Amicus Curiae: Technophobia, Law and Creativity in the Digital Arts*, 19 U. DAYTON L. REV. 903 (1994); Arthur Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 HARV. L. REV**.** 977, 1029-32 (1993).

159. Aymes v. Bonnelli, 47 F.3d 23 (2d Cir. 1995).

160. *See, e.g.*, MAI v. Peak, 991 F.2d 511, 518-19 (9th Cir. 1993).

161. Lotus Dev. Corp. v. Borland Int'l., 49 F.3d 807, 821 (9th Cir. 1994) (Boudin, J., concurring).

162. *Id.*

163. *Id*.