COMPUTER ASSOCIATES INTERNATIONAL, INC., Plaintiff-Appellant-Cross-Appellee, v. ALTAI, INC., Defendant-Appellee-Cross-Appellant.

Docket Nos. 91-7893, 91-7935

UNITED STATES COURT OF APPEALS FOR THE SECOND CIRCUIT

982 F.2d 693; 1992 U.S. App. LEXIS 33369; 119 A.L.R. Fed. 741; 92 Cal. Daily Op. Service 10213

January 9, 1992, Argued
December 17, 1992, Filed

Before ALTIMARI, MAHONEY and WALKER, Circuit Judges. Judge Altimari concurs in part and dissents in part in a separate opinion.

WALKER, *Circuit Judge:*

In recent years, the growth of computer science has spawned a number of challenging legal questions, particularly in the field of copyright law. As scientific knowledge advances, courts endeavor to keep pace, and sometimes--as in the area of computer technology--they are required to venture into less than familiar waters. This is not a new development, though. "From its beginning, the law of copyright has developed in response to significant changes in technology."

Article I, section 8 of the Constitution authorizes Congress "to promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries." The Supreme Court has stated that "the economic philosophy behind the clause . . . is the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare . . . ." *Mazer v. Stein,* 347 U.S. 201, 219, 98 L. Ed. 630, 74 S. Ct. 460 (1954). The author's benefit, however, is clearly a "secondary" consideration. "The ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good."

Thus, the copyright law seeks to establish a delicate equilibrium. On the one hand, it affords protection to authors as an incentive to create, and, on the other, it must appropriately limit the extent of that protection so as to avoid the effects of monopolistic stagnation. In applying the federal act to new types of cases, courts must always keep this symmetry in mind.

Among other things, this case deals with the challenging question of whether and to what extent the "non-literal" aspects of a computer program, that is, those aspects that are not reduced to written code, are protected by copyright. While a few other courts have already grappled with this issue, this case is one of first impression in this circuit. As we shall discuss, we find the results reached by other courts to be less than satisfactory. Drawing upon long-standing doctrines of copyright law, we take an approach that we think better addresses the practical difficulties embedded in these types of cases. In so doing, we have kept in mind the necessary balance between creative incentive and industrial competition.

This appeal comes to us from the United States District Court for the Eastern District of New York, the Honorable George C. Pratt, *Circuit Judge,* sitting by designation. By Memorandum and Order entered August 12, 1991, Judge Pratt found that defendant Altai, Inc.'s ("Altai"), OSCAR 3.4 computer program had infringed plaintiff Computer Associates' ("CA"), copyrighted computer program entitled CA-SCHEDULER. Accordingly, the district court awarded CA $364,444 in actual damages and apportioned profits. Altai has abandoned its appeal from this award. With respect to CA's second claim for copyright infringement, Judge Pratt found that Altai's OSCAR 3.5 program was not substantially similar to a portion of CA-SCHEDULER called ADAPTER, and thus denied relief. Finally, the district court concluded that CA's state law trade secret misappropriation claim against Altai had been preempted by the federal copyright act. CA appealed from these findings.

Because we are in full agreement with Judge Pratt's decision and in substantial agreement with his careful reasoning regarding CA's copyright infringement claim, we affirm the district court's judgment on that issue. However, we vacate the district court's preemption ruling with respect to CA's trade secret claim, and remand the case to the district court for further proceedings.

BACKGROUND

We assume familiarity with the facts set forth in the district court's comprehensive and scholarly opinion. *See Computer Assocs. Int'l, Inc. v. Altai, Inc.,* 775 F. Supp. 544, 549-55 (E.D.N.Y. 1991). Thus, we summarize only those facts necessary to resolve this appeal.

I. COMPUTER PROGRAM DESIGN

Certain elementary facts concerning the nature of computer programs are vital to the following discussion. The Copyright Act defines a computer program as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101. In writing these directions, the programmer works "from the general to the specific."

The first step in this procedure is to identify a program's ultimate function or purpose. An example of such an ultimate purpose might be the creation and maintenance of a business ledger. Once this goal has been achieved, a programmer breaks down or "decomposes" the program's ultimate function into "simpler constituent problems or 'subtasks,'" which are also known as subroutines or modules. In the context of a business ledger program, a module or subroutine might be responsible for the task of updating a list of outstanding accounts receivable. Sometimes, depending upon the complexity of its task, a subroutine may be broken down further into sub-subroutines.

Having sufficiently decomposed the program's ultimate function into its component elements, a programmer will then arrange the subroutines or modules into what are known as organizational or flow charts. Flow charts map the interactions between modules that achieve the program's end goal.

In order to accomplish these intra-program interactions, a programmer must carefully design each module's parameter list. A parameter list, according to the expert appointed and fully credited by the district court, Dr. Randall Davis, is "the information sent to and received from a subroutine." The term "parameter list" refers to the form in which information is passed between modules (e.g. for accounts receivable, the designated time frame and particular customer identifying number) and the information's actual content (e.g. 8/91-7/92; customer No. 3). With respect to form, interacting modules must share similar parameter lists so that they are capable of exchanging information.

"The functions of the modules in a program together with each module's relationships to other modules constitute the 'structure' of the program." Additionally, the term structure may include the category of modules referred to as "macros." A macro is a single instruction that initiates a sequence of operations or module interactions within the program. Very often the user will accompany a macro with an instruction from the parameter list to refine the instruction (e.g. current total of accounts receivable (macro), but limited to those for 8/91 to 7/92 from customer No. 3 (parameters)).

In fashioning the structure, a programmer will normally attempt to maximize the program's speed, efficiency, as well as simplicity for user operation, while taking into consideration certain externalities such as the memory constraints of the computer upon which the program will be run. "This stage of program design often requires the most time and investment."

Once each necessary module has been identified, designed, and its relationship to the other modules has been laid out conceptually, the resulting program structure must be embodied in a written language that the computer can read. This process is called "coding," and requires two steps. First, the programmer must transpose the program's structural blue-print into a source code. This step has been described as "comparable to the novelist fleshing out the broad outline of his plot by crafting from words and sentences the paragraphs that convey the ideas." The source code may be written in any one of several computer languages, such as COBAL, FORTRAN, BASIC, EDL, etc., depending upon the type of computer for which the program is intended. Once the source code has been completed, the second step is to translate or "compile" it into object code. Object code is the binary language comprised of zeros and ones through which the computer directly receives its instructions.

After the coding is finished, the programmer will run the program on the computer in order to find and correct any logical and syntactical errors. This is known as "debugging" and, once done, the program is complete.

II. FACTS

CA is a Delaware corporation, with its principal place of business in Garden City, New York. Altai is a Texas corporation, doing business primarily in Arlington, Texas. Both companies are in the computer software industry--designing, developing and marketing various types of computer programs.

The subject of this litigation originates with one of CA's marketed programs entitled CA-SCHEDULER. CA-SCHEDULER is a job scheduling program designed for IBM mainframe computers. Its primary functions are straightforward: to create a schedule specifying when the computer should run various tasks, and then to control the computer as it executes the schedule. CA-SCHEDULER contains a sub-program entitled ADAPTER, also developed by CA. ADAPTER is not an independently marketed product of CA; it is a wholly integrated component of CA-SCHEDULER and has no capacity for independent use.

Nevertheless, ADAPTER plays an extremely important role. It is an "operating system compatibility component," which means, roughly speaking, it serves as a translator. An "operating system" is itself a program that manages the resources of the computer, allocating those resources to other programs as needed. The IBM System 370 family of computers, for which CA-SCHEDULER was created, is, depending upon the computer's size, designed to contain one of three operating systems: DOS/VSE, MVS, or CMS. As the district court noted, the general rule is that "a program written for

one operating system, e.g., DOS/VSE, will not, without modification, run under another operating system such as MVS." ADAPTER's function is to translate the language of a given program into the particular language that the computer's own operating system can understand.

The district court succinctly outlined the manner in which ADAPTER works within the context of the larger program. In order to enable CA-SCHEDULER to function on different operating systems, CA divided the CA-SCHEDULER into two components:

> --a first component that contains only the task-specific portions of the program, independent of all operating system issues, and
>
> --a second component that contains all the interconnections between the first component and the operating system.
>
> In a program constructed in this way, whenever the first, task-specific, component needs to ask the operating system for some resource through a "system call", it calls the second component instead of calling the operating system directly.
>
> The second component serves as an "interface" or "compatibility component" between the task-specific portion of the program and the operating system. It receives the request from the first component and translates it into the appropriate system call that will be recognized by whatever operating system is installed on the computer, *e.g.,* DOS/VSE, MVS, or CMS. Since the first, task-specific component calls the adapter component rather than the operating system, the first component need not be customized to use any specific operating system. The second, interface, component insures that all the system calls are performed properly for the particular operating system in use.

ADAPTER serves as the second, "common system interface" component referred to above.

A program like ADAPTER, which allows a computer user to change or use multiple operating systems while maintaining the same software, is highly desirable. It saves the user the costs, both in time and money, that otherwise would be expended in purchasing new programs, modifying existing systems to run them, and gaining familiarity with their operation. The benefits run both ways. The increased compatibility afforded by an ADAPTER-like component, and its resulting popularity among consumers, makes whatever software in which it is incorporated significantly more marketable.

Starting in 1982, Altai began marketing its own job scheduling program entitled ZEKE. The original version of ZEKE was designed for use in conjunction with a VSE operating system. By late 1983, in response to customer demand, Altai decided to rewrite ZEKE so that it could be run in conjunction with an MVS operating system.

At that time, James P. Williams ("Williams"), then an employee of Altai and now its President, approached Claude F. Arney, III ("Arney"), a computer programmer who worked for CA. Williams and Arney were longstanding friends, and had in fact been co-workers at CA for some time before Williams left CA to work for Altai's predecessor. Williams wanted to recruit Arney to assist Altai in designing an MVS version of ZEKE.

At the time he first spoke with Arney, Williams was aware of both the CA-SCHEDULER and ADAPTER programs. However, Williams was not involved in their development and had never

seen the codes of either program. When he asked Arney to come work for Altai, Williams did not know that ADAPTER was a component of CA-SCHEDULER.

Arney, on the other hand, was intimately familiar with various aspects of ADAPTER. While working for CA, he helped improve the VSE version of ADAPTER, and was permitted to take home a copy of ADAPTER'S source code. This apparently developed into an irresistible habit, for when Arney left CA to work for Altai in January, 1984, he took with him copies of the source code for both the VSE and MVS versions of ADAPTER. He did this in knowing violation of the CA employee agreements that he had signed.

Once at Altai, Arney and Williams discussed design possibilities for adapting ZEKE to run on MVS operating systems. Williams, who had created the VSE version of ZEKE, thought that approximately 30% of his original program would have to be modified in order to accommodate MVS. Arney persuaded Williams that the best way to make the needed modifications was to introduce a "common system interface" component into ZEKE. He did not tell Williams that his idea stemmed from his familiarity with ADAPTER. They decided to name this new component-program OSCAR.

Arney went to work creating OSCAR at Altai's offices using the ADAPTER source code. The district court accepted Williams' testimony that no one at Altai, with the exception of Arney, affirmatively knew that Arney had the ADAPTER code, or that he was using it to create OSCAR/VSE. However, during this time period, Williams' office was adjacent to Arney's. Williams testified that he and Arney "conversed quite frequently" while Arney was "investigating the source code of ZEKE" and that Arney was in his office "a number of times daily, asking questions." In three months, Arney successfully completed the OSCAR/VSE project. In an additional month he developed an OSCAR/MVS version. When the dust finally settled, Arney had copied approximately 30% of OSCAR's code from CA's ADAPTER program.

The first generation of OSCAR programs was known as OSCAR 3.4. From 1985 to August 1988, Altai used OSCAR 3.4 in its ZEKE product, as well as in programs entitled ZACK and ZEBB. In late July 1988, CA first learned that Altai may have appropriated parts of ADAPTER. After confirming its suspicions, CA secured copyrights on its 2.1 and 7.0 versions of CA-SCHEDULER. CA then brought this copyright and trade secret misappropriation action against Altai.

Apparently, it was upon receipt of the summons and complaint that Altai first learned that Arney had copied much of the OSCAR code from ADAPTER. After Arney confirmed to Williams that CA's accusations of copying were true, Williams immediately set out to survey the damage. Without ever looking at the ADAPTER code himself, Williams learned from Arney exactly which sections of code Arney had taken from ADAPTER.

Upon advice of counsel, Williams initiated OSCAR's rewrite. The project's goal was to save as much of OSCAR 3.4 as legitimately could be used, and to excise those portions which had been copied from ADAPTER. Arney was entirely excluded from the process, and his copy of the ADAPTER code was locked away. Williams put eight other programmers on the project, none of whom had been involved in any way in the development of OSCAR 3.4. Williams provided the programmers with a description of the ZEKE operating system services so that they could rewrite the appropriate code. The rewrite project took about six months to complete and was finished in mid-November 1989. The resulting program was entitled OSCAR 3.5.

From that point on, Altai shipped only OSCAR 3.5 to its new customers. Altai also shipped OSCAR 3.5 as a "free upgrade" to all customers that had previously purchased OSCAR 3.4. While Altai and Williams acted responsibly to correct Arney's literal copying of the ADAPTER program, copyright infringement had occurred.

After CA originally instituted this action in the United States District Court for the District of New Jersey, the parties stipulated its transfer in March, 1989, to the Eastern District of New York where it was assigned to Judge Jacob Mishler. On October 26, 1989, Judge Mishler transferred the case to Judge Pratt who was sitting in the district court by designation. Judge Pratt conducted a six day trial from March 28 through April 6, 1990. He entered judgment on August 12, 1991, and this appeal followed.

DISCUSSION

While both parties originally appealed from different aspects of the district court's judgment, Altai has now abandoned its appellate claims. In particular, Altai has conceded liability for the copying of ADAPTER into OSCAR 3.4 and raises no challenge to the award of $ 364,444 in damages on that score. Thus, we address only CA's appeal from the district court's rulings that: (1) Altai was not liable for copyright infringement in developing OSCAR 3.5; and (2) in developing both OSCAR 3.4 and 3.5, Altai was not liable for misappropriating CA's trade secrets.

CA makes two arguments. First, CA contends that the district court applied an erroneous method for determining whether there exists substantial similarity between computer programs, and thus, erred in determining that OSCAR 3.5 did not infringe the copyrights held on the different versions of its CA-SCHEDULER program. CA asserts that the test applied by the district court failed to account sufficiently for a computer program's non-literal elements. Second, CA maintains that the district court erroneously concluded that its state law trade secret claims had been preempted by the federal copyright act, *see* 17 U.S.C. § 301(a). We shall address each argument in turn.

I. COPYRIGHT INFRINGEMENT

In any suit for copyright infringement, the plaintiff must establish its ownership of a valid copyright, and that the defendant copied the copyrighted work. The plaintiff may prove defendant's copying either by direct evidence or, as is most often the case, by showing that (1) the defendant had access to the plaintiff's copyrighted work and (2) that defendant's work is substantially similar to the plaintiff's copyrightable material.

For the purpose of analysis, the district court assumed that Altai had access to the ADAPTER code when creating OSCAR 3.5. Thus, in determining whether Altai had unlawfully copied protected aspects of CA's ADAPTER, the district court narrowed its focus of inquiry to ascertaining whether Altai's OSCAR 3.5 was substantially similar to ADAPTER. Because we approve Judge Pratt's conclusions regarding substantial similarity, our analysis will proceed along the same assumption.

As a general matter, and to varying degrees, copyright protection extends beyond a literary work's strictly textual form to its non-literal components. As we have said, "it is of course essential to any protection of literary property . . . that the right cannot be limited literally to the text, else a plagiarist would escape by immaterial variations." *Nichols v. Universal Pictures Co.,* 45 F.2d 119, 121 (2d Cir. 1930) (L. Hand, J.), *cert. denied,* 282 U.S. 902 (1931). Thus, where "the fundamental essence or structure of one work is duplicated in another," 3 Nimmer, § 13.03[A][1], at 13-24,

courts have found copyright infringement. *See, e.g., Horgan v. Macmillan,* 789 F.2d 157, 162 (2d Cir. 1986) (recognizing that a book of photographs might infringe ballet choreography); *Twentieth Century-Fox Film Corp. v. MCA, Inc.,* 715 F.2d 1327, 1329 (9th Cir. 1983) (motion picture and television series); *Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp.,* 562 F.2d 1157, 1167 (9th Cir. 1977) (television commercial and television series); *Sheldon v. Metro-Goldwyn Pictures Corp.,* 81 F.2d 49, 55 (2d Cir.), *cert. denied,* 298 U.S. 669, 80 L. Ed. 1392, 56 S. Ct. 835 (1936) (play and motion picture); *accord Stewart v. Abend,* 495 U.S. 207, 238, 109 L. Ed. 2d 184, 110 S. Ct. 1750 (1990) (recognizing that motion picture may infringe copyright in book by using its "unique setting, characters, plot, and sequence of events"). This black letter proposition is the springboard for our discussion.

A. *Copyright Protection for the Non-literal Elements of Computer Programs*

It is now well settled that the literal elements of computer programs, i.e., their source and object codes, are the subject of copyright protection. *See Whelan,* 797 F.2d at 1233 (source and object code); *CMS Software Design Sys., Inc. v. Info Designs, Inc.,* 785 F.2d 1246, 1247 (5th Cir. 1986) (source code); *Apple Computer, Inc. v. Franklin Computer Corp.,* 714 F.2d 1240, 1249 (3d Cir. 1983), *cert. dismissed,* 464 U.S. 1033 (1984) (source and object code); *Williams Electronics, Inc. v. Artic Int'l, Inc.,* 685 F.2d 870, 876-77 (3d Cir. 1982) (object code). Here, as noted earlier, Altai admits having copied approximately 30% of the OSCAR 3.4 program from CA's ADAPTER source code, and does not challenge the district court's related finding of infringement.

In this case, the hotly contested issues surround OSCAR 3.5. As recounted above, OSCAR 3.5 is the product of Altai's carefully orchestrated rewrite of OSCAR 3.4. After the purge, none of the ADAPTER source code remained in the 3.5 version; thus, Altai made sure that the literal elements of its revamped OSCAR program were no longer substantially similar to the literal elements of CA's ADAPTER.

According to CA, the district court erroneously concluded that Altai's OSCAR 3.5 was not substantially similar to its own ADAPTER program. CA argues that this occurred because the district court "committed legal error in analyzing [its] claims of copyright infringement by failing to find that copyright protects expression contained in the non-literal elements of computer software." We disagree.

CA argues that, despite Altai's rewrite of the OSCAR code, the resulting program remained substantially similar to the *structure* of its ADAPTER program. As discussed above, a program's structure includes its nonliteral components such as general flow charts as well as the more specific organization of inter-modular relationships, parameter lists, and macros. In addition to these aspects, CA contends that OSCAR 3.5 is also substantially similar to ADAPTER with respect to the list of services that both ADAPTER and OSCAR obtain from their respective operating systems. We must decide whether and to what extent these elements of computer programs are protected by copyright law.

The statutory terrain in this area has been well explored. The Copyright Act affords protection to "original works of authorship fixed in any tangible medium of expression . . . ." 17 U.S.C. § 102(a). This broad category of protected "works" includes "literary works," *id.* at § 102(a)(1), which are defined by the Act as

> works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film tapes, disks, or cards, in which they are embodied.

17 U.S.C. § 101. While computer programs are not specifically listed as part of the above statutory definition, the legislative history leaves no doubt that Congress intended them to be considered literary works. *See* H.R.Rep. No. 1476, 94th Cong., 2d Sess. 54, *reprinted in* 1976 U.S.C.C.A.N. 5659, 5667 (hereinafter *"House Report"); Whelan,* 797 F.2d at 1234; *Apple Computer,* 714 F.2d at 1247.

The syllogism that follows from the foregoing premises is a powerful one: if the non-literal structures of literary works are protected by copyright; and if computer programs are literary works, as we are told by the legislature; then the non-literal structures of computer programs are protected by copyright. We have no reservation in joining the company of those courts that have already ascribed to this logic. However, 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.

As a caveat, we note that our decision here does not control infringement actions regarding categorically distinct works, such as certain types of screen displays. These items represent products of computer programs, rather than the programs themselves, and fall under the copyright rubric of audiovisual works. If a computer audiovisual display is copyrighted separately as an audiovisual work, apart from the literary work that generates it (i.e., the program), the display may be protectable regardless of the underlying program's copyright status. *See Stern Elecs., Inc. v. Kaufman,* 669 F.2d 852, 855 (2d Cir. 1982) (explaining that an audiovisual works copyright, rather than a copyright on the underlying program, extended greater protection to the sights and sounds generated by a computer video game because the same audiovisual display could be generated by different programs). Of course, the copyright protection that these displays enjoy extends only so far as their expression is protectable. In this case, however, we are concerned not with a program's display, but the program itself, and then with only its non-literal components. In considering the copyrightability of these components, we must refer to venerable doctrines of copyright law.

1) *Idea vs. Expression Dichotomy*

It is a fundamental principle of copyright law that a copyright does not protect an idea, but only the expression of the idea. This axiom of common law has been incorporated into the governing statute. Section 102(b) of the Act provides:

> In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

Congress made no special exception for computer programs. To the contrary, the legislative history explicitly states that copyright protects computer programs only "to the extent that they incorporate authorship in programmer's expression of original ideas, as distinguished from the ideas themselves." ...

Drawing the line between idea and expression is a tricky business. Judge Learned Hand noted that "nobody has ever been able to fix that boundary, and nobody ever can." *Nichols,* 45 F.2d at 121. Thirty years later his convictions remained firm. "Obviously, no principle can be stated as to when an imitator has gone beyond copying the 'idea,' and has borrowed its 'expression,'" Judge Hand concluded. "Decisions must therefore inevitably be *ad hoc." Peter Pan Fabrics, Inc. v. Martin Weiner Corp.,* 274 F.2d 487, 489 (2d Cir. 1960).

The essentially utilitarian nature of a computer program further complicates the task of distilling its idea from its expression. In order to describe both computational processes and abstract ideas, its content "combines creative and technical expression." The variations of expression found in purely creative compositions, as opposed to those contained in utilitarian works, are not directed towards practical application. For example, a narration of Humpty Dumpty's demise, which would clearly be a creative composition, does not serve the same ends as, say, a recipe for scrambled eggs--which is a more process oriented text. Thus, compared to aesthetic works, computer programs hover even more closely to the elusive boundary line described in § 102(b).

The doctrinal starting point in analyses of utilitarian works, is the seminal case of *Baker v. Selden,* 101 U.S. 99, 25 L. Ed. 841 (1879). In *Baker,* the Supreme Court faced the question of "whether the exclusive property in a system of bookkeeping can be claimed, under the law of copyright, by means of a book in which that system is explained?" Selden had copyrighted a book that expounded a particular method of bookkeeping. The book contained lined pages with headings intended to illustrate the manner in which the system operated. Baker's accounting publication included ledger sheets that employed "substantially the same ruled lines and headings. . . ." *Id.* Selden's testator sued Baker for copyright infringement on the theory that the ledger sheets were protected by Selden's copyright.

The Supreme Court found nothing copyrightable in Selden's bookkeeping system, and rejected his infringement claim regarding the ledger sheets. The Court held that:

> The fact that the art described in the book by illustrations of lines and figures which are reproduced in practice in the application of the art, makes no difference. Those illustrations are the mere language employed by the author to convey his ideas more clearly. Had he used words of description instead of diagrams (which merely stand in the place of words), there could not be the slightest doubt that others, applying the art to practical use, might lawfully draw the lines and diagrams which were in the author's mind, and which he thus described by words in his book.
>
> The copyright of a work on mathematical science cannot give to the author an exclusive right to the methods of operation which he propounds, or to the diagrams which he employs to explain them, so as to prevent an engineer from using them whenever occasion requires.

Id. at 103.

To the extent that an accounting text and a computer program are both "a set of statements or instructions . . . to bring about a certain result," 17 U.S.C. § 101, they are roughly analogous. In the former case, the processes are ultimately conducted by human agency; in the latter, by electronic means. In either case, as already stated, the processes themselves are not protectable. But the holding in *Baker* goes farther. The Court concluded that those aspects of a work, which "must necessar-

ily be used as incident to" the idea, system or process that the work describes, are also not copyrightable. Selden's ledger sheets, therefore, enjoyed no copyright protection because they were "necessary incidents to" the system of accounting that he described. *Id.* at 103. From this reasoning, we conclude that those elements of a computer program that are necessarily incidental to its function are similarly unprotectable.

While *Baker v. Selden* provides a sound analytical foundation, it offers scant guidance on how to separate idea or process from expression, and moreover, on how to further distinguish protectable expression from that expression which "must necessarily be used as incident to" the work's underlying concept. In the context of computer programs, the Third Circuit's noted decision in *Whelan* has, thus far, been the most thoughtful attempt to accomplish these ends.

The court in *Whelan* faced substantially the same problem as is presented by this case. There, the defendant was accused of making off with the non-literal structure of the plaintiff's copyrighted dental lab management program, and employing it to create its own competitive version. In assessing whether there had been an infringement, the court had to determine which aspects of the programs involved were ideas, and which were expression. In separating the two, the court settled upon the following conceptual approach:

> The line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question. In other words, *the 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 the 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.

797 F.2d at 1236 (citations omitted). The "idea" of the program at issue in *Whelan* was identified by the court as simply "the efficient management of a dental laboratory." *Id.* at n.28.

So far, in the courts, the *Whelan* rule has received a mixed reception. While some decisions have adopted its reasoning, *see, e.g., Bull HN Info. Sys., Inc. v. American Express Bank, Ltd.,* 1990 Copyright Law Dec. (CCH) P 26,555 at 23,278 (S.D.N.Y. 1990); *Dynamic Solutions, Inc. v. Planning & Control, Inc.,* 1987 Copyright Law Dec. (CCH) P 26,062 at 20,912 (S.D.N.Y. 1987); *Broderbund Software Inc. v. Unison World, Inc.,* 648 F. Supp. 1127, 1133 (N.D.Cal. 1986), others have rejected it. *See Plains Cotton Co-op v. Goodpasture Computer Serv, Inc.,* 807 F.2d 1256, 1262 (5th Cir.), *cert. denied,* 484 U.S. 821, 108 S. Ct. 80, 98 L. Ed. 2d 42 (1987); *cf. Synercom Technology, Inc. v. University Computing Co.,* 462 F. Supp. 1003, 1014 (N.D.Tex. 1978) (concluding that order and sequence of data on computer input formats was idea not expression).

Whelan has fared even more poorly in the academic community, where its standard for distinguishing idea from expression has been widely criticized for being conceptually overbroad. The leading commentator in the field has stated that, "the crucial flaw in *[Whelan's]* reasoning is that it assumes that only one 'idea,' in copyright law terms, underlies any computer program, and that once a separable idea can be identified, everything else must be expression." 3 Nimmer § 13.03[F], at 13-62.34. This criticism focuses not upon the program's ultimate purpose but upon the reality of its structural design. As we have already noted, a computer program's ultimate function or purpose is the composite result of interacting subroutines. Since each subroutine is itself a program, and thus,

may be said to have its own "idea," *Whelan's* general formulation that a program's overall purpose equates with the program's idea is descriptively inadequate.

Accordingly, we think that Judge Pratt wisely declined to follow *Whelan.* In addition to noting the weakness in the *Whelan* definition of "program-idea," mentioned above, Judge Pratt found that *Whelan's* synonymous use of the terms "structure, sequence, and organization," demonstrated a flawed understanding of a computer program's method of operation. Rightly, the district court found *Whelan's* rationale 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.

2) *Substantial Similarity Test for Computer Program Structure: Abstraction-Filtration-Comparison*

We think that *Whelan's* approach to separating idea from expression in computer programs relies too heavily on metaphysical distinctions and does not place enough emphasis on practical considerations. As the cases that we shall discuss demonstrate, a satisfactory answer to this problem cannot be reached by resorting, *a priori,* to philosophical first principals.

As discussed herein, we think that district courts would be well-advised to undertake a three-step procedure, based on the abstractions test utilized by the district court, in order to determine whether the non-literal elements of two or more computer programs are substantially similar. This approach breaks no new ground; rather, it draws on such familiar copyright doctrines as merger, *scenes a faire,* and public domain. In taking this approach, however, we are cognizant that computer technology is a dynamic field which can quickly outpace judicial decisionmaking. Thus, in cases where the technology in question does not allow for a literal application of the procedure we outline below, our opinion should not be read to foreclose the district courts of our circuit from utilizing a modified version.

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

Step One: Abstraction

As the district court appreciated, *see Computer Assocs.,* 775 F. Supp. at 560, the theoretic framework for analyzing substantial similarity expounded by Learned Hand in the *Nichols* case is helpful in the present context. In *Nichols,* we enunciated what has now become known as the "abstractions" test for separating idea from expression:

> Upon 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; but 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," to which, apart from their expression, his property is never extended.

Nichols, 45 F.2d at 121.

While the abstractions test was originally applied in relation to literary works such as novels and plays, it is adaptable to computer programs. In contrast to the *Whelan* approach, the abstractions test "implicitly recognizes that any given work may consist of a mixture of numerous ideas and expressions."

As applied to computer programs, the abstractions test will comprise the first step in the examination for substantial similarity. Initially, in a manner that resembles reverse engineering on a theoretical plane, a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it. This process begins with the code and ends with an articulation of the program's ultimate function. Along the way, it is necessary essentially to retrace and map each of the designer's steps--in the opposite order in which they were taken during the program's creation. *See* Background: Computer Program Design, *supra.*

As an anatomical guide to this procedure, the following description is helpful:

> At the lowest level of abstraction, a computer program may be thought of in its entirety as a set of individual instructions organized into a hierarchy of modules. At a higher level of abstraction, the instructions in the lowest-level modules may be replaced conceptually by the functions of those modules. At progressively higher levels of abstraction, the functions of higher-level modules conceptually replace the implementations of those modules in terms of lower-level modules and instructions, until finally, one is left with nothing but the ultimate function of the program. . . . A program has structure at every level of abstraction at which it is viewed. At low levels of abstraction, a program's structure may be quite complex; at the highest level it is trivial.

Step Two: Filtration

Once the program's abstraction levels have been discovered, the substantial similarity inquiry moves from the conceptual to the concrete. Professor Nimmer suggests, and we endorse, a "successive filtering method" for separating protectable expression from non-protectable material. This process entails examining the structural components at each level of abstraction to determine whether their particular inclusion at that level was "idea" or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself; or taken from the public domain and hence is nonprotectable expression. The structure of any given program may reflect some, all, or none of these considerations. Each case requires its own fact specific investigation.

Strictly speaking, this filtration serves "the purpose of defining the scope of plaintiff's copyright." By applying well developed doctrines of copyright law, it may ultimately leave behind a "core of protectable material." Further explication of this second step may be helpful.

(a) *Elements Dictated by Efficiency*

The portion of *Baker v. Selden,* discussed earlier, which denies copyright protection to expression necessarily incidental to the idea being expressed, appears to be the cornerstone for what has developed into the doctrine of merger. *See Morrissey v. Procter & Gamble Co.,* 379 F.2d 675, 678-79 (1st Cir. 1967) (relying on *Baker* for the proposition that expression embodying the rules of a sweepstakes contest was inseparable from the idea of the contest itself, and therefore were not protectable by copyright). The doctrine's underlying principle is that "when there is essentially only one way to express an idea, the idea and its expression are inseparable and copyright is no bar to copying that expression." Under these circumstances, the expression is said to have "merged" with the idea itself. In order not to confer a monopoly of the idea upon the copyright owner, such expression should not be protected. …

Furthermore, when one considers the fact that programmers generally strive to create programs "that meet the user's needs in the most efficient manner," the applicability of the merger doctrine to computer programs becomes compelling. In the context of computer program design, the concept of efficiency is akin to deriving the most concise logical proof or formulating the most succinct mathematical computation. Thus, the more efficient a set of modules are, the more closely they approximate the idea or process embodied in that particular aspect of the program's structure.

While, hypothetically, there might be a myriad of ways in which a programmer may effectuate certain functions within a program,--i.e., express the idea embodied in a given subroutine--efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable options. Of course, not all program structure is informed by efficiency concerns. *See* Menell, at 1052 (besides efficiency, simplicity related to user accommodation has become a programming priority). It follows that in order to determine whether the merger doctrine precludes copyright protection to an aspect of a program's structure that is so oriented, a court must inquire "whether the use of *this particular set* of modules is necessary efficiently to implement that part of the program's process" being implemented. If the answer is yes, then the expression represented by the programmer's choice of a specific module or group of modules has merged with their underlying idea and is unprotected.

Another justification for linking structural economy with the application of the merger doctrine stems from a program's essentially utilitarian nature and the competitive forces that exist in the software marketplace. Working in tandem, these factors give rise to a problem of proof which merger helps to eliminate.

Efficiency is an industry-wide goal. Since, as we have already noted, there may be only a limited number of efficient implementations for any given program task, it is quite possible that multiple programmers, working independently, will design the identical method employed in the allegedly infringed work. Of course, if this is the case, there is no copyright infringement.

Under these circumstances, the fact that two programs contain the same efficient structure may as likely lead to an inference of independent creation as it does to one of copying. Thus, since evidence of similarly efficient structure is not particularly probative of copying, it should be disregarded in the overall substantial similarity analysis. …

(b) *Elements Dictated By External Factors*

We have stated that where "it is virtually impossible to write about a particular historical era or fictional theme without employing certain 'stock' or standard literary devices," such expression is not copyrightable. *Hoehling v. Universal City Studios, Inc.,* 618 F.2d 972, 979 (2d Cir.), *cert. de-*

nied, 449 U.S. 841, 66 L. Ed. 2d 49, 101 S. Ct. 121 (1980). For example, the *Hoehling* case was an infringement suit stemming from several works on the Hindenberg disaster. There we concluded that similarities in representations of German beer halls, scenes depicting German greetings such as "Heil Hitler," or the singing of certain German songs would not lead to a finding of infringement because they were "'indispensable, or at least standard, in the treatment of'" life in Nazi Germany. This is known as the *scenes a faire* doctrine, and like "merger," it has its analogous application to computer programs.

Professor Nimmer points out that "in many instances it is virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques." This is a result of the fact that a programmer's freedom of design choice is often circumscribed by extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program [*710] is intended to run; (2) compatibility requirements of other programs with which 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.

Courts have already considered some of these factors in denying copyright protection to various elements of computer programs. … Building upon this existing case law, we conclude that a court must also examine the structural content of an allegedly infringed program for elements that might have been dictated by external factors.

(c) *Elements taken From the Public Domain*

Closely related to the non-protectability of *scenes a faire,* is material found in the public domain. Such material is free for the taking and cannot be appropriated by a single author even though it is included in a copyrighted work. We see no reason to make an exception to this rule for elements of a computer program that have entered the public domain by virtue of freely accessible program exchanges and the like. Thus, a court must also filter out this material from the allegedly infringed program before it makes the final inquiry in its substantial similarity analysis.

Step Three: Comparison

The third and final step of the test for substantial similarity that we believe appropriate for non-literal program components entails a comparison. Once a court has sifted out all elements of the allegedly infringed program which are "ideas" or are dictated by efficiency or external factors, or taken from the public domain, there may remain a core of protectable expression. In terms of a work's copyright value, this is the golden nugget. At this point, the court's substantial similarity inquiry focuses on whether the defendant copied any aspect of this protected expression, as well as an assessment of the copied portion's relative importance with respect to the plaintiff's overall program. *See* 3 Nimmer § 13.03[F][5]; *Data East USA,* 862 F.2d at 208 ("To determine whether similarities result from unprotectable expression, analytic dissection of *similarities* may be performed. If . . . all similarities in expression arise from use of common ideas, then no substantial similarity can be found.").

3) *Policy Considerations*

We are satisfied that the three step approach we have just outlined not only comports with, but advances the constitutional policies underlying the Copyright Act. Since any method that tries to distinguish idea from expression ultimately impacts on the scope of copyright protection afforded to

a particular type of work, "the line [it draws] must be a pragmatic one, which also keeps in consideration 'the preservation of the balance between competition and protection. . . .'"

CA and some *amici* argue against the type of approach that we have set forth on the grounds that it will be a disincentive for future computer program research and development. At bottom, they claim that if programmers are not guaranteed broad copyright protection for their work, they will not invest the extensive time, energy and funds required to design and improve program structures. While they have a point, their argument cannot carry the day. The 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.

In this respect, our conclusion is informed by Justice Stewart's concise discussion of the principles that correctly govern the adaptation of the copyright law to new circumstances. In *Twentieth Century Music Corp. v. Aiken,* he wrote:

> The limited scope of the copyright holder's statutory monopoly, like the limited copyright duration required by the Constitution, reflects a balance of competing claims upon the public interest: Creative work is to be encouraged and rewarded, but private motivation must ultimately serve the cause of promoting broad public availability of literature, music, and the other arts.
>
> The immediate effect of our copyright law is to secure a fair return for an "author's" creative labor. But the ultimate aim is, by this incentive, to stimulate artistic creativity for the general public good. . . . When technological change has rendered its literal terms ambiguous, the Copyright Act must be construed in light of this basic purpose.

422 U.S. 151, 156, 95 S. Ct. 2040, 45 L. Ed. 2d 84 (1975) (citations and footnotes omitted).

Recently, the Supreme Court has emphatically reiterated that "the primary objective of copyright is not to reward the labor of authors. . . ." *Feist Publications, Inc. v. Rural Tel. Serv. Co.,* 113 L. Ed. 2d 358, 111 S. Ct. 1282, 1290 (1991) (emphasis added). While the *Feist* decision deals primarily with the copyrightability of purely factual compilations, its underlying tenets apply to much of the work involved in computer programming. *Feist* put to rest the "sweat of the brow" doctrine in copyright law. The rationale of that doctrine "was that copyright was a reward for the hard work that went into compiling facts." The Court flatly rejected this justification for extending copyright protection, noting that it "eschewed the most fundamental axiom of copyright law--that no one may copyright facts or ideas."

Feist teaches that substantial effort alone cannot confer copyright status on an otherwise uncopyrightable work. As we have discussed, despite the fact that significant labor and expense often goes into computer program flow-charting and debugging, that process does not always result in inherently protectable expression. Thus, *Feist* implicitly undercuts the *Whelan* rationale, "which allowed copyright protection beyond the literal computer code . . . [in order to] provide the proper incentive for programmers by protecting their most valuable efforts. . . ." We note that *Whelan* was decided prior to *Feist* when the "sweat of the brow" doctrine still had vitality. In view of the Supreme Court's recent holding, however, we must reject the legal basis of CA's disincentive argument.

Furthermore, we are unpersuaded that the test we approve today will lead to the dire consequences for the computer program industry that plaintiff and some amici predict. To the contrary, serious students of the industry have been highly critical of the sweeping scope of copyright protection engendered by the *Whelan* rule, in that it "enables first comers to 'lock up' basic programming techniques as implemented in programs to perform particular tasks."

To be frank, the exact contours of copyright protection for non-literal program structure are not completely clear. We trust that as future cases are decided, those limits will become better defined. Indeed, it may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes. This results from the hybrid nature of a computer program, which, while it is literary expression, is also a highly functional, utilitarian component in the larger process of computing.

Generally, we think that copyright registration--with its indiscriminating availability--is not ideally suited to deal with the highly dynamic technology of computer science. Thus far, many of the decisions in this area reflect the courts' attempt to fit the proverbial square peg in a round hole. The district court and at least one commentator have suggested that patent registration, with its exacting up-front novelty and non-obviousness requirements, might be the more appropriate rubric of protection for intellectual property of this kind. In any event, now that more than 12 years have passed since CONTU issued its final report, the resolution of this specific issue could benefit from further legislative investigation--perhaps a CONTU II.

In the meantime, Congress has made clear that computer programs are literary works entitled to copyright protection. Of course, we shall abide by these instructions, but in so doing we must not impair the overall integrity of copyright law. While incentive based arguments in favor of broad copyright protection are perhaps attractive from a pure policy perspective, ultimately, they have a corrosive effect on certain fundamental tenets of copyright doctrine. If the test we have outlined results in narrowing the scope of protection, as we expect it will, that result flows from applying, in accordance with Congressional intent, long-standing principles of copyright law to computer programs. Of course, our decision is also informed by our concern that these fundamental principles remain undistorted.

B. *The District Court Decision*

We turn now to our review of the district court's decision in this particular case. …

2) *Evidentiary Analysis*

The district court had to determine whether Altai's OSCAR 3.5 program was substantially similar to CA's ADAPTER. We note that Judge Pratt's method of analysis effectively served as a road map for our own, with one exception Judge Pratt filtered out the non-copyrightable aspects of OSCAR 3.5 rather than those found in ADAPTER, the allegedly infringed program. We think that our approach--i.e., filtering out the unprotected aspects of an allegedly infringed program and then comparing the end product to the structure of the suspect program--is preferable, and therefore believe that district courts should proceed in this manner in future cases

We opt for this strategy because, in some cases, the defendant's program structure might contain protectable expression and/or other elements that are not found in the plaintiff's program. Since it is extraneous to the allegedly copied work, this material would have no bearing on any potential substantial similarity between the two programs. Thus, its filtration would be wasteful and unnecessar-

ily time consuming. Furthermore, by focusing the analysis on the infringing rather than on the infringed material, a court may mistakenly place too little emphasis on a quantitatively small misappropriation which is, in reality, a qualitatively vital aspect of the plaintiff's protectable expression.

The fact that the district court's analysis proceeded in the reverse order, however, had no material impact on the outcome of this case. Since Judge Pratt determined that OSCAR effectively contained no protectable expression whatsoever, the most serious charge that can be levelled against him is that he was overly thorough in his examination.

The district court took the first step in the analysis set forth in this opinion when it separated the program by levels of abstraction. The district court stated:

> As applied to computer software programs, this abstractions test would progress in order of "increasing generality" from object code, to source code, to parameter lists, to services required, to general outline. In discussing the particular similarities, therefore, we shall focus on these levels.

While the facts of a different case might require that a district court draw a more particularized blueprint of a program's overall structure, this description is a workable one for the case at hand.

Moving to the district court's evaluation of OSCAR 3.5's structural components, we agree with Judge Pratt's systematic exclusion of non-protectable expression. With respect to code, the district court observed that after the rewrite of OSCAR 3.4 to OSCAR 3.5, "there remained virtually no lines of code that were identical to ADAPTER." Accordingly, the court found that the code "presented no similarity at all."

Next, Judge Pratt addressed the issue of similarity between the two programs' parameter lists and macros. He concluded that, viewing the conflicting evidence most favorably to CA, it demonstrated that "only a few of the lists and macros were similar to protected elements in ADAPTER; the others were either in the public domain or dictated by the functional demands of the program." *Id.* As discussed above, functional elements and elements taken from the public domain do not qualify for copyright protection. With respect to the few remaining parameter lists and macros, the district court could reasonably conclude that they did not warrant a finding of infringement given their relative contribution to the overall program. *See Warner Bros., Inc. v. American Broadcasting Cos., Inc.,* 720 F.2d 231, 242 (2d Cir. 1983) (discussing *de minimis* exception which allows for literal copying of a small and usually insignificant portion of the plaintiff's work); 3 Nimmer § 13.03[F][5], at 13-74. In any event, the district court reasonably found that, for lack of persuasive evidence, CA failed to meet its burden of proof on whether the macros and parameter lists at issue were substantially similar.

The district court also found that the overlap exhibited between the list of services required for both ADAPTER and OSCAR 3.5 was "determined by the demands of the operating system and of the applications program to which it [was] to be linked through ADAPTER or OSCAR. . . ." In other words, this aspect of the program's structure was dictated by the nature of other programs with which it was designed to interact and, thus, is not protected by copyright.

Finally, in his infringement analysis, Judge Pratt accorded no weight to the similarities between the two programs' organizational charts, "because [the charts were] so simple and obvious to anyone exposed to the operation of the programs." CA argues that the district court's action in this regard "is not consistent with copyright law"--that "obvious" expression is protected, and that the district

court erroneously failed to realize this. However, to say that elements of a work are "obvious," in the manner in which the district court used the word, is to say that they "follow naturally from the work's theme rather than from the author's creativity." This is but one formulation of the *scenes a faire* doctrine, which we have already endorsed as a means of weeding out unprotectable expression.

CA argues, at some length, that many of the district court's factual conclusions regarding the creative nature of its program's components are simply wrong. Of course, we are limited in our review of factual findings to setting aside only those that we determine are clearly erroneous. *See* Fed. R. Civ. P. 52. Upon a thorough review of the voluminous record in this case, which is comprised of conflicting testimony and other highly technical evidence, we discern no error on the part of Judge Pratt, let alone clear error.

Since we accept Judge Pratt's factual conclusions and the results of his legal analysis, we affirm his dismissal of CA's copyright infringement claim based upon OSCAR 3.5. We emphasize that, like all copyright infringement cases, those that involve computer programs are highly fact specific. The amount of protection due structural elements, in any given case, will vary according to the protectable expression found to exist within the program at issue. …

[The court's analysis of the issue of trade secret preemption is omitted.]

CONCLUSION

In adopting the above three step analysis for substantial similarity between the non-literal elements of computer programs, we seek to insure two things: (1) that programmers may receive appropriate copyright protection for innovative utilitarian works containing expression; and (2) that non-protectable technical expression remains in the public domain for others to use freely as building blocks in their own work. At first blush, it may seem counter-intuitive that someone who has benefited to some degree from illicitly obtained material can emerge from an infringement suit relatively unscathed. However, so long as the appropriated material consists of non-protectable expression, "this result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science and art."