Computer Associations International, Inc., v. Altai, Inc.,

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

Before ALTIMARI, MAHONEY and WALKER, Circuit Judges.

* * * * * 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.

* * * * *

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." Computer Assocs., 775 F.Supp. at 550. 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.

Id. at 551. 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.

III 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.

* * * * *

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. See Novelty Textile Mills, Inc. v. Joan Fabrics Corp., 558 F.2d 1090, 1092 (2d Cir.1977); see also 3 Melville B. Nimmer & David Nimmer, Nimmer on Copyright s 13.01, at 13-4 (1991) (hereinafter "Nimmer"). 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. See Walker v. Time Life Films, Inc., 784 F.2d 44, 48 (2d Cir.), cert. denied, 476 U.S. 1159, 106 S.Ct. 2278, 90 L.Ed.2d 721 (1986).

For the purpose of analysis, the district court assumed that Altai had access to the ADAPTER code when creating OSCAR 3.5. See Computer Assocs., 775 F.Supp. at 558. 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, "[i]t 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, 51 S.Ct. 216, 75 L.Ed. 795 (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, 56 S.Ct. 835, 80 L.Ed. 1392 (1936) (play and motion picture); accord Stewart v. Abend, 495 U.S. 207, 238, 110 S.Ct. 1750, 1769, 109 L.Ed.2d 184 (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, 104 S.Ct. 690, 79 L.Ed.2d 158 (1984) (source and object code); Williams Elecs., 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 non-literal 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. See Lotus Dev. Corp. v. Paperback Software Int'l, 740 F.Supp. 37, 47-51 (D.Mass.1990); see also Whelan, 797 F.2d at 1240-42; Englund, at 885-90; Spivack, at 731-37. 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. See

Whelan, 797 F.2d at 1234 ("By analogy to other literary works, it would thus appear that the copyrights of computer programs can be infringed even absent copying of the literal elements of the program."). We have no reservation in joining the company of those courts that have already ascribed to this logic. See, e.g., Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir.1989); Lotus Dev. Corp., 740 F.Supp. at 54; Digital Communications Assocs., Inc. v. Softklone Distrib. Corp., 659 F.Supp. 449, 455-56 (N.D.Ga.1987); Q- Co Industries, Inc. v. Hoffman, 625 F.Supp. 608, 615 (S.D.N.Y.1985); SAS Inst., Inc. v. S & H Computer Sys., Inc., 605 F.Supp. 816, 829-30 (M.D.Tenn.1985). 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. See Data East USA, Inc. v. Epyx, Inc., 862 F.2d 204, 209 (9th Cir.1988). 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. See Baker v. Selden, 101 U.S. 99, 25 L.Ed. 841 (1879); Mazer v. Stein, 347 U.S. 201, 217, 74 S.Ct. 460, 470, 98 L.Ed. 630 (1954). 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.

17 U.S.C. §102(b). See also House Report, at 5670 ("Copyright does not preclude others from using ideas or information revealed by the author's 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." Id. at 5667; see also id. at 5670 ("Section 102(b) is intended ... to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of copyright law.").

Similarly, the National Commission on New Technological Uses of Copyrighted Works ("CONTU") established by Congress to survey the issues generated by the interrelationship of advancing technology

and copyright law, see Pub.L. No. 93-573, §201, 88 Stat. 1873 (1974), recommended, inter alia, that the 1976 Copyright Act "be amended ... to make it explicit that computer programs, to the extent that they embody the author's original creation, are proper subject matter for copyright." See National Commission on New Technological Uses of Copyrighted Works, Final Report 1 (1979) (hereinafter "CONTU Report"). To that end, Congress adopted CONTU's suggestions and amended the Copyright Act by adding, among other things, a provision to 17 U.S.C. s 101 which defined the term "computer program." See Pub.L. No. 96-517, §10(a), 94 Stat. 3028 (1980). CONTU also "concluded that the idea-expression distinction should be used to determine which aspects of computer programs are copyrightable." Lotus Dev. Corp., 740 F.Supp. at 54 (citing CONTU Report, at 44).

Drawing the line between idea and expression is a tricky business. Judge Learned Hand noted that "[n]obody 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. See SAS Inst., 605 F.Supp. at 829; cf. Englund, at 893. In order to describe both computational processes and abstract ideas, its content "combines creative and technical expression." See Spivack, at 755. 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 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:

> [T]he 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. * * * * * 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. * * * * *

Accordingly, we think that Judge Pratt wisely declined to follow Whelan. See Computer Assocs., 775 F.Supp. at 558-60. 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," see Whelan, 797 F.2d at 1224 n. 1, demonstrated a flawed understanding of a computer program's method of operation. See Computer Assocs., 775 F.Supp. at 559-60 (discussing the distinction between a program's "static structure" and "dynamic structure"). 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. Cf. Apple Computer, 714 F.2d at 1253 (rejecting certain commercial constraints on programming as a helpful means of distinguishing idea from expression because they did "not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged"). 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 possible kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program. The result of this comparison will determine whether the protectable elements of the programs at issue are substantially similar so as to warrant a finding of infringement. 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." 3 Nimmer §13.03[F], at 13-62.34-63.

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.

Englund, at 897-98; cf. Spivack, at 774.

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. See generally 3 Nimmer §13.03[F]. 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. See also Kretschmer, at 844-45 (arguing that program features dictated by market externalities or efficiency concerns are

unprotectable). 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." Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1475 (9th Cir.) (endorsing "analytic dissection" of computer programs in order to isolate protectable expression), cert. denied, --- U.S. ----, 113 S.Ct. 198, 121 L.Ed.2d 141 (1992). By applying well developed doctrines of copyright law, it may ultimately leave behind a "core of protectable material." 3 Nimmer §13.03[F][5], at 13-72. 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. Proctor & 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); see also Digital Communications, 659 F.Supp. at 457. The doctrine's underlying principle is that "[w]hen there is essentially only one way to express an idea, the idea and its expression are inseparable and copyright is no bar to copying that expression." Concrete Machinery Co. v. Classic Lawn Ornaments, Inc., 843 F.2d 600, 606 (1st Cir.1988). 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. See Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir.1971).

CONTU recognized the applicability of the merger doctrine to computer programs. In its report to Congress it stated that:

> [C]opyrighted language may be copied without infringing when there is but a limited number of ways to express a given idea.... In the computer context, this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement.

CONTU Report, at 20. While this statement directly concerns only the application of merger to program code, that is, the textual aspect of the program, it reasonably suggests that the doctrine fits comfortably within the general context of computer programs.

Furthermore, when one considers the fact that programmers generally strive to create programs "that meet the user's needs in the most efficient manner," Menell, at 1052, 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. See 3 Nimmer §13.03[F][2], at 13-63; see also Whelan, 797 F.2d at 1243 n. 43 ("It is true that for

certain tasks there are only a very limited number of file structures available, and in such cases the structures might not be copyrightable...."). 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. Englund, at 902. 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. Id. at 902-03.

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. See Kretschmer, at 842. 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. See Roth Greeting Cards v. United Card Co., 429 F.2d 1106, 1110 (9th Cir.1970); Sheldon, 81 F.2d at 54.

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. See 3 Nimmer §13.03[F][2], at 13-65; cf. Herbert Rosenthal Jewelry Corp., 446 F.2d at 741 (evidence of independent creation may stem from defendant's standing as a designer of previous similar works). Thus, since evidence of similarly efficient structure is not particularly probative of copying, it should be disregarded in the overall substantial similarity analysis. See 3 Nimmer §13.03[F][2], at 13-65.

We find support for applying the merger doctrine in cases that have already addressed the question of substantial similarity in the context of computer program structure. Most recently, in Lotus Dev. Corp., 740 F.Supp. at 66, the district court had before it a claim of copyright infringement relating to the structure of a computer spreadsheet program. The court observed that "the basic spreadsheet screen display that resembles a rotated 'L' ..., if not present in every expression of such a program, is present in most expressions." Id. Similarly, the court found that "an essential detail present in most if not all expressions of an electronic spreadsheet--is the designation of a particular key that, when pressed, will invoke the menu command system." Id. Applying the merger doctrine, the court denied copyright protection to both program elements.

In Manufacturers Technologies, Inc. v. Cams, Inc., 706 F.Supp. 984, 995-99 (D.Conn.1989), the infringement claims stemmed from various alleged program similarities "as indicated in their screen displays." Id. at 990. Stressing efficiency concerns in the context of a merger analysis, the court determined that the program's method of allowing the user to navigate within the screen displays was not protectable because, in part, "the process or manner of navigating internally on any specific screen displays ... is limited in the number of ways it may be simply achieved to facilitate user comfort." Id. at 995.

The court also found that expression contained in various screen displays (in the form of alphabetical and numerical columns) was not the proper subject of copyright protection because it was "necessarily

incident to the idea[s]" embodied in the displays. Id. at 996-97. Cf. Digital Communications, 659 F.Supp. at 460 (finding no merger and affording copyright protection to program's status screen display because "modes of expression chosen ... are clearly not necessary to the idea of the status screen").

We agree with the approach taken in these decisions, and conclude that application of the merger doctrine in this setting is an effective way to eliminate non-protectable expression contained in computer programs.

(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. denied, 449 U.S. 841, 101 S.Ct. 121, 66 L.Ed.2d 49 (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. Id. (quoting Alexander v. Haley, 460 F.Supp. 40, 45 (S.D.N.Y.1978)). This is known as the scenes a faire doctrine, and like "merger," it has its analogous application to computer programs. Cf. Data East USA, 862 F.2d at 208 (applying scenes a faire to a home computer video game).

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." 3 Nimmer §13.03[F][3], at 13-65. 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 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. Id. at 13-66-71.

Courts have already considered some of these factors in denying copyright protection to various elements of computer programs. In the Plains Cotton case, the Fifth Circuit refused to reverse the district court's denial of a preliminary injunction against an alleged program infringer because, in part, "many of the similarities between the ... programs [were] dictated by the externalities of the cotton market." 807 F.2d at 1262.

In Manufacturers Technologies, the district court noted that the program's method of screen navigation "is influenced by the type of hardware that the software is designed to be used on." 706 F.Supp. at 995. Because, in part, "the functioning of the hardware package impact[ed] and constrain[ed] the type of navigational tools used in plaintiff's screen displays," the court denied copyright protection to that aspect of the program. Id.; cf. Data East USA, 862 F.2d at 209 (reversing a district court's finding of audiovisual work infringement because, inter alia, "the use of the Commodore computer for a karate game intended for home consumption is subject to various constraints inherent in the use of that computer").

Finally, the district court in Q-Co Industries rested its holding on what, perhaps, most closely approximates a traditional scenes a faire rationale. There, the court denied copyright protection to four

program modules employed in a teleprompter program. This decision was ultimately based upon the court's finding that "the same modules would be an inherent part of any prompting program." 625 F.Supp. at 616.

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. See E.F. Johnson Co. v. Uniden Corp. of America, 623 F.Supp. 1485, 1499 (D.Minn.1985); see also Sheldon, 81 F.2d at 54. 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. See 3 Nimmer §13.03[F][4]; see also Brown Bag Software, 960 F.2d at 1473 (affirming the district court's finding that " '[p]laintiffs may not claim copyright protection of an ... expression that is, if not standard, then commonplace in the computer software industry.' "). 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. See Brown Bag Software, 960 F.2d at 1475. 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.").

* * * * *

II. TRADE SECRET PREEMPTION

* * * * * *

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 benefitted 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, "[t]his result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science and art." Feist, --- U.S. at ----, 111 S.Ct. at 1290.

Furthermore, we underscore that so long as trade secret law is employed in a manner that does not encroach upon the exclusive domain of the Copyright Act, it is an appropriate means by which to secure compensation for software espionage.

Accordingly, we affirm the judgment of the district court in part; vacate in part; and remand for further proceedings. The parties shall bear their own costs of appeal, including the petition for rehearing.

ALTIMARI, Circuit Judge, concurring in part and dissenting in part:

Because I believe that our original opinion, see Computer Assoc. Int'l v. Altai, Nos. 91-7893(L), (2d Cir. June 22, 1992), is a reasoned analysis of the issues presented, I adhere to the original determination and therefore concur in Part 1 and respectfully dissent from Part 2 of the amended opinion.