THE EC DIRECTIVE ON THE LEGAL PROTECTION OF COMPUTER SOFTWARE: NEW LAW GOVERNING SOFTWARE DEVELOPMENT

Alan K. Palmer*
Thomas C. Vinje**

On May 14, 1991, the Council of Ministers (Council) of the European Community (EC) officially adopted the Directive on the Legal Protection of Computer Programs.[1] Adoption of the Software Directive completed a three-year process that involved intense lobbying efforts in Brussels, the central seat of government for the EC, and in Strasbourg, the seat of the European Parliament. The process culminated in the adoption of a Software Directive that specifically allows certain types of reverse engineering.[2] This paper traces the legislative history of the Software Directive, a history which clearly indicates the EC intent not to frustrate competitive forces in the computer industry.

With the promulgation of the Software Directive, the EC surpassed the United States and Japan, two world leaders in computer technology, in the specificity with which important issues of legal protection for computer software are addressed.[3] It may well be that future developments in

* Partner, Morrison & Foerster Law Firm, Washington D.C. Office. A.B., Dartmouth College; J.D., Harvard Law School.

** Partner, Morrison & Foerster Law Firm, Brussels Office. B.A., University of Washington; J.D., Columbia University School of Law.

1. Council Directive 91/250 of 14 May 1991 on the Legal Protection of Computer Programs, 1991 O.J. (L 122) 42 [hereinafter Software Directive].

2. The term reverse engineering, which often is used interchangeably with reverse analysis, generally refers to a process by which a product is systematically broken down into its component parts to analyze and discover the composition of the product. The United States Supreme Court defined reverse engineering in Kewannee Oil Co. v. Bicron Corp., 416 U.S. 470 (1974) as "starting with the known product and working backward to divine the process which aided in its development or manufacture." *Id.* at 476. Reverse engineering techniques in the computer context include line traces, test runs, memory dumps, and disassembly. Because all these techniques may be used to analyze a computer program to learn about its interfaces, they are more properly grouped under the term reverse analysis, and that is the term that will be used hereinafter to refer to these techniques. For an excellent discussion of reverse analysis in relation to computer software, see Andy Johnson-Laird, *Reverse Engineering: Separating Legal Mythology From Modern Day Technology*, 5 TEKBRIEFS 7, 8-9 (Jan.-Feb. 1991). See *infra* part I.B.2. for a more complete discussion.

3. The status of United States and Japanese law was of concern to EC decision makers because it was generally deemed unwise for the EC to stray too far from the legal regimes applicable to

EC legislation will significantly affect the law governing reverse analysis of software and the degree of protection given to a computer program's interfaces[4] in the United States and Japan. Thus, the origin and evolution of the Software Directive is of potential worldwide interest to observers of computer law and intellectual property issues.

I. INTRODUCTION

A. The Legal Protection of Computer Software

The legal system of the United States grappled with various computer-related issues before most other jurisdictions because of its leadership role in the development and widespread use of computers and associated products. One of the first questions to arise in the United States regarding computer software was whether software should be protected either under copyright law or under a separate intellectual property regime.[5] Copyright was an obvious candidate since computer programs are written in lines of code resembling textual material; which traditionally is covered by copyright law.[6]

Copyright and software, however, are not a perfect fit:

> Computer software, by its very nature as . . . intended to serve utilitarian purposes, defies easy categorization within our intellectual property system. The copyright law has traditionally served as the principal source of legal protection for original literary work, while the patent system and trade secret law have been the primary means for protecting novel utilitarian works.[7]

Largely for this reason, many academics have argued that software protection should be governed by a *sui generis* legal regime developed for that purpose, partaking of some elements of patent law as well as various aspects of copyright law.[8] Nonetheless, the United States Copyright Office, the federal courts, and the United States Congress have decided that computer software is copyrightable subject matter.[9]

software in the other two major jurisdictions whose laws govern major innovations in computer technology.

4. Interfaces refer to the aspects of a program that permit its interaction with other components of a computer system. See *infra* part I.B.1. for a more detailed discussion.

5. *See generally* UNITED STATES NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS (CONTU) (1979).

6. *See id.* at 15.

7. Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 STAN. L. REV. 1045, 1046 (1989).

8. *See, e.g.*, Pamela Samuelson, *Creating a New Kind of Intellectual Property: Applying the Lessons of the Chip Law to Computer Programs*, 70 MINN. L. REV. 471 (1986).

9. PAUL GOLDSTEIN, COPYRIGHT § 2.15.2, at 205 n.46 (1989).

The debate concerning how software should be treated is not limited to the United States. European legislators also struggled with the issue of how the protection of computer software should be governed by intellectual property law. Since the late 1980s, however, it was clear that most EC member states accepted copyright law as the appropriate method of protecting computer software, albeit with some divergence.[10]

B. The Genesis of the Software Directive[11]

On June 7, 1988, the European Commission (Commission) issued a Green Paper[12] on Copyright and the Challenge of Technology, outlining the options available for the protection of computer software.[13] The Green Paper assessed existing European copyright law and discussed the possibility of *sui generis* protection.[14] It proposed that copyright law would continue as the principal vehicle for computer software protection, and it described alternative approaches to a number of copyright issues.[15]

After considering comments submitted by interested observers concerning the Green Paper, on January 5, 1989, the Commission[16] issued a

10. *See, e.g.*, Copyright, Designs & Patents Act, 1988, ch. 48 § 3(1)(b) (Eng.); Code Civil [C. Civ.] art. 543 (91st ed. Codes 1991-92) (Fr.).

11. Directives are binding as to the result to be achieved, but they leave to national authorities the choice of methods to be used. EEC TREATY art. 189. Directives must be implemented through national law in each member state to be effective. Therefore, the member states' legal regimes may differ with respect to how the directive is implemented. The process by which directives related to the coming EC single market are adopted is known as the cooperation procedure. *Id.* arts. 100a (as amended 1987), 149 (as amended 1987). For a detailed discussion of the EC's institutional organization and legislative process, see AUDREY WINTER ET AL., EUROPE WITHOUT FRONTIERS: A LAWYER'S GUIDE (BNA Corporate Practice Series, 1989). For a collection of EC legislation, see BLACKSTONE'S EEC LEGISLATION (Nigel G. Foster, ed., 1990).

12. Green Papers are working papers issued by the Commission that address substantive issues of proposed directives. See *infra* note 16, for a short explanation of how the European Commission operates.

13. Green Paper on Copyright and the Challenge of Technology: Copyright Issues Requiring Immediate Action COM(88)172 final (1988).

14. *Id.* § 5.3.5.

15. *Id.* § 5.5. Among the broader issues raised in the Green Paper were "the protection of the economic interests of the author or creator and the promotion and pursuit of cultural goals." *Id.* § 5.3.4. More specifically, the Green Paper addressed the interrelation of copyright and industrial designs, television by cable and satellite, semiconductors, computer technology, and new audio/visual recording techniques. *Id.* § 1.1.3.

16. The Commission is the EC's executive or bureaucratic apparatus. It draws its personnel from the member states, but the personnel do not represent the interest of their countries of origin. Rather they function as civil servants, and respond to the interests of the Directorates-General who formulate the Commission's work. Directorates-General are the primary bureaucratic ministries of the Commission. Derek W. Urwin, *A-Z of European Communities*, *in* THE EUROPEAN COMMUNITIES ENCYCLOPEDIA & DIRECTORY 1992, 29 (1991). They ensure that the policies are carried out, and they administer various policy areas. The Directorates-General fall into 23 policy areas. The areas are typically referred to as DG and the corresponding Roman numeral (for example, DG-III).

proposal for a directive.[17] The Proposed Directive was drafted principally by the Directorate-General for Internal Market and Industrial Affairs (DG-III). Additionally, the Directorates-General for Competition (DG-IV) and for Telecommunications, Information Industries and Innovation (DG-XIII), which were so-called associated services, played a significant role in formulating Commission policy on the Proposed Directive. The interplay between these three Directorates-General is integral to understanding the development of the Software Directive.[18]

Intense lobbying of all three EC decision-making institutions (the Commission, the Council of Ministers, and the Parliament) began in mid-1989 and continued for nearly two years. The lobbyists were concerned with two aspects of the Proposed Directive: the scope of protection for software interfaces, which permit a program's interaction with other components of a computer system,[19] and the permissibility of reverse analysis. Companies with strong market positions generally found it in their interest to support copyright protection that would prevent or impede reverse analysis and that would also limit or prevent the use of their programs' interfaces. Other companies, with less powerful market positions, and software users, maintained largely opposite positions due to their ulterior concerns.

Publication of the Proposed Directive galvanized the latter group into action. This was because the Proposed Directive implied that under some circumstances a program's interface specifications could not lawfully be used by persons other than the program's developer. Furthermore, it incorporated an expansive prohibition on reproduction without an exception that would allow reverse analysis.[20]

1. *The Role of Interfaces.* Interfaces are the aspects of a program that permit its interaction with other components of a computer system.

17. Proposal for a Council Directive on the Legal Protection of Computer Programs, 1989 O.J. (C 91) 4 [hereinafter Proposed Directive] (includes accompanying Explanatory Memorandum).

18. *See infra* part II.

19. The Proposed Directive defined the term interface in its preamble as follows:

> Whereas . . . a logical and, where appropriate, physical interconnection and interaction is required to permit all elements of software and hardware to work with other software and hardware and with users in all the ways they are intended to function; *whereas the principles describing any such means of interconnection and interaction are generally known as 'an interface'* . . . .

Proposed Directive, *supra* note 17, pmbl., para. 12 (emphasis added). The Software Directive as eventually enacted had a shortened but similar definition of the term interface: "Whereas the parts of the program which provide for such interconnection and interaction between elements of software and hardware are generally known as 'interfaces' . . . ." Software Directive, *supra* note 1, pmbl., para. 15.

20. Proposed Directive, *supra* note 17, arts. 1(3), 4. *See also* European Committee for Interoperable Systems, Statement of Principles (1989) (on file with authors) [hereinafter ECIS Statement of Principles].

Conceptually, interfaces can be divided into interface specifications and interface implementations. Interface specifications are the rules and methods whereby computer products interact with one another. Interface implementations are the implementation of the interface specifications into a program's code.

In the past, computer systems largely functioned independently; one user's system had little need to interact with other users' systems. Some of the large computer companies, such as IBM, DEC, Burroughs, Sperry, and Apple, developed a unique set of interface specifications which also determined the way in which the various components of the computer systems that they marketed would interact with one another. A user who bought a system developed by one vendor was locked into an environment which conformed to that company's interface specifications. While this constrained the user's choice of products — and increased its costs when it sought to expand or upgrade its system — the existence of different sets of interface specifications did not seriously constrain its data processing capabilities because there was little need to interconnect with other systems.

Today, by contrast, computer technology is more pervasive and there is a significant need for interconnection between systems with different interface specifications. Within a given company, literally thousands of personal computers and work stations scattered across the globe may need to interact with each other and with a company's mainframes, as well as with the computers of other companies. With the advent of such computer networks, the lines between different systems have blurred. Thus, a user at one personal computer might access data from a disc drive attached to a second personal computer, process the data with software loaded on a third computer, and print out the result on a printer plugged into a fourth computer.

In the days of stand-alone systems, user dissatisfaction with being locked into a proprietary system led to a demand for third party products compatible with the proprietary systems. This, along with the movement toward computer networks, has dramatically accelerated the demand for interoperable products. To develop a new computer program that can attach to or replace an existing program, software developers must conform their new program to the interface specifications of the existing program. Otherwise, the new program simply could not function in the existing program environment.

Therefore, those who were concerned with the terms of the Proposed Directive regarded it as essential that no company should monopolize an interface. In their view, if the first companies in the market could copyright the interface specifications of their programs, those early entrants could use copyright law to lock in users and insulate themselves from competition.[21] The Proposed Directive could have been read to grant such exclusivity in some undefined circumstances.[22]

2. *The Role of Reverse Analysis.* Even if interface specifications are not copyrightable, a new entrant in the software market could face a second obstacle under the Proposed Directive in using the earlier entrant's interfaces. The object code format[23] in which most programs are distributed to the public cannot be read without the aid of computerized reverse analysis techniques. Thus, many of a program's characteristics, including its interface specifications, often are not discernible without reverse analysis of the program. Most types of reverse analysis, however, arguably involve the making of reproductions, since the mere act of running a program, in a technical sense, reproduces the program's code.[24] Reverse analysis, in the view of opponents of the Proposed Directive, is widely practiced throughout the world and is an entirely legitimate means of competition. However, reverse analysis would be barred by the extremely broad prohibition on reproduction in the Proposed Directive.[25]

3. *Interfaces and Reverse Analysis.* In the autumn of 1989, many opponents of the Proposed Directive, including Groupe Bull, Olivetti, NCR, Unisys, Fujitsu, and certain major users of software, joined together to form the European Committee for Interoperable Systems (ECIS).[26] ECIS began to lobby forcefully for a procompetition approach to the treatment of interfaces and reverse analysis in the Software Directive. In response to the formation of ECIS, a competing group was created. This group called itself the Software Action Group for Europe (SAGE), and its most

21. *See* Michel Colombe & Caroline Meyer, *Seeking Interoperability: An Industry Response*, 3 EUR. INTELL. PROP. REV. 79 (1990).

22. Proposed Directive, *supra* note 17, art. 1(3).

23. To oversimplify a complicated subject, object code consists of strings of ones and zeros that generally can be understood only by a computer; source code is human-readable code and is what a programmer first creates; and assembly code is an intermediate step between object code and high level source code.

24. The right to control reproduction in Article 4(1)(a) of the directive is fundamental to achieve adequate protection for computer programs. Unlike other forms of literary work, a computer program cannot serve its purpose unless it is 'reproduced'. The program may be re-created in part or in whole as part of the internal processes of the computer which runs it. No second permanent copy of the program is made during this process, although parts of the program will be 'reproduced' and stored in other parts of the memory of the computer during the operation of the program. These . . . operations may leave no trace once the operation of the machine has terminated. Thus 'copying' in the traditional sense of producing a second permanent version of an original does not normally take place unless a 'back-up' copy of the program is made.

Proposed Directive, *supra* note 17, at 10 (Explanatory memorandum discussing art. 4(a)).

25. *Id.* art. 4(a).

26. *See* ECIS Statement of Principles, *supra* note 20.

prominent members included IBM, DEC, Apple, Microsoft, and Lotus.[27] SAGE sharply challenged the basic arguments of ECIS, taking the view that a broad scope of protection for interfaces and a prohibition on reverse analysis were necessary to fight software piracy and to stimulate innovation in the software industry.[28] As contrasted with the procompetition orientation of ECIS, SAGE positioned itself as supporting and promoting the benefits of strong intellectual property protection.[29]

II. THE PROCEDURAL HISTORY OF THE DIRECTIVE

A. The Commission's Proposed Directive

With publication of the Proposed Directive and the formation of ECIS and SAGE, the stage was set for a debate of great significance in the computer industry with respect to both the scope of protection for interfaces and the permissibility of reverse analysis. As noted, the Proposed Directive could be read to restrict the use of interface specifications. Article 1(3) stated as follows:

> Protection in accordance with this Directive shall not extend to the ideas, principles, logic, algorithms or programming languages underlying the program. *Where the specification of interfaces constitutes ideas and principles which underlie the program*, those ideas and principles are not copyrightable subject matter.[30]

An inference may be drawn from this proposed language that interface specifications might sometimes constitute protected expression and thus

27. SAGE members signed the Industry Statement in Support of the Software Action Group for Europe (on file with the authors) [hereinafter Industry Statement in Support of SAGE]. The statement included the principles of SAGE and indicated that each signing entity was supportive of a proposal to protect computer programs as "literary works," that any exclusion of protection for logic, algorithms, programming languages, and interface specifications would severely damage protection for legitimate works, and that any amendment that would authorize copying under "research and analysis" techniques would deny U.S. software authors the level of protection in Europe to which their efforts are entitled. *Id.* *See also* Letter from Sybase, Inc., to Emery Simon, Director, Intellectual Property Policy, Office of the United States Trade Representative (Jan. 18, 1990) (on file with authors) (articulating that Sybase, Inc. supported Proposed Directive).

28. *See* Industry Statement in Support of SAGE, *supra* note 27.

29. SAGE took the position that the Proposed Directive was consistent with U.S. law, which, according to SAGE, prohibited reverse analysis. SAGE argued that because reverse analysis often required the making of technical reproductions, it violated the letter of the U.S. Copyright Act. *Id.* para. 5. *See generally* Hubco Data Products Corp. v. Management Assistance, Inc., 219 U.S.P.Q. 450 (D.C. Idaho 1983) (arguably supporting this view). ECIS, by contrast, countered that U.S. law *permitted* reverse analysis. ECIS Statement of Principles, *supra* note 20 at 9. Arguably, technical reproductions involved in reverse analysis are permitted in light of the fair use doctrine, 17 U.S.C. § 107 (1988), the limitation on exclusive rights with respect to computer programs contained in 17 U.S.C. § 117 (1988), and such cases as NEC Corp. v. Intel Corp., 10 U.S. P.Q.2d 1177 (N.D. Cal. 1989), E.F. Johnson Co. v. Uniden Corp. of America, 623 F. Supp. 1485, 1501 n.17 (D. Minn. 1986).

30. Proposed Directive, *supra* note 17, art. 1(3) (emphasis added).

could be used only by the copyright holder.[31] Moreover, Article 4 of the Proposed Directive gave the copyright owner the exclusive rights to reproduce or authorize reproduction of a program code, with reproduction defined in an expansive manner.[32] Since all reverse analysis of a program arguably requires reproduction of a program code, the above definition appeared to give copyright holders the right to prohibit such analysis. Furthermore, there was no exception permitting reverse analysis of a program.

ECIS vigorously protested the formulation of Articles 1(3) and 4. Its arguments were directed to the Commission, the Council of Ministers, and the European Parliament, which began active consideration of the Proposed Directive in the latter half of 1989.[33] At the Commission, the Directorate-General charged with enforcing EC competition law and the Directorate-General responsible for information technology policy were skeptical of the Proposed Directive's terms, and, therefore, were receptive to the ECIS arguments.[34] The Directorate-General primarily responsible for single market initiatives, whose staff included the principal authors of the Proposed Directive, was less sympathetic. At this stage, this Directorate-General appeared to be persuaded by arguments asserted by SAGE that reverse analysis was the weapon of software pirates. SAGE main-

31. ECIS acknowledged that interface implementations generally constituted protected expression.

> It is widely accepted that copyright law is intended to protect only the expression of ideas, and not the ideas themselves. In the case of interfaces this means that the implementation of an interface in the program code is protected, but the interface specifications underlying the program are not. The proposed Directive on the Legal Protection of Computer Programs should clearly make this distinction. Failure to do so would undermine the ability of competitors to offer interoperable computer products.

ECIS, The Distinction Between Interface Specifications and Implementations 1 (on file with authors).

32. For a definition of reproduction, see *supra* note 24.

33. Once a directive is proposed by the Commission, it is referred to the Council of Ministers and submitted to the European Parliament for the Parliament's first reading. EEC TREATY art. 149 (as amended 1987). The European Parliament operates on the basis of a committee system similar to that used by the United States Congress. Usually one committee plays a key role, while others occupy subsidiary roles. The lead committee for the Software Directive was the Committee on Legal Affairs & Citizens' Rights (Legal Affairs Committee). The Economic and Monetary Affairs Committee and the Energy, Research, and Technology Committee comprised the subsidiary committees. The Proposed Directive also was referred to the EC's Economic & Social Committee (ECOSOC). ECOSOC renders an opinion on proposed directives; however, its opinions generally carry little weight in the EC's decision making process.

34. The Directorate-General responsible for Competition was previously involved in a massive proceeding involving the extent to which IBM should be required to allow its competitors to use its interfaces. That proceeding was settled in 1984 when IBM entered into an undertaking to provide information to competitors about interfaces for its System/370 mainframe products. Commission of the European Communities, Press Release, No. IP (84) 290 (Aug. 2, 1984). DG-IV was concerned that the Software Directive's provisions on interfaces and reverse analysis might undermine the IBM undertaking and DG-IV's future ability to pursue actions similar to the IBM proceeding.

tained that if reverse analysis were permitted, it would impede innovation by reducing the rewards to copyright holders.[35]

B. ECIS, SAGE, and the European Parliament

Both lobbying groups brought computer experts to Brussels to explain complicated technical concepts to Commission officials. In addition, both sides sponsored seminars designed to educate EC decision makers and impress them with the widespread support they enjoyed among prominent and reputable companies.

Similar efforts at persuasion, including petitions and massive letter-writing campaigns, were directed at the European Parliament. In Parliament, Committee on the Energy, Research, and Technology and the Economic and Monetary Affairs Committee considered the Proposed Directive and issued Opinions on November 8, 1989, and March 19, 1990, respectively.[36] Generally, the Economic and Monetary Affairs Committee[37] was supportive of the SAGE position.[38] On the other hand, the Energy, Research, and Technology Committee[39] strongly supported the procompetitive forces that stood behind ECIS and concluded that a ban on reverse analysis "would have . . . a serious restrictive effect on innovation and competition within the EC. . . ."[40] The Legal Affairs Committee[41] served as the lead committee on this proposal, taking into account opinions issued by both the Committee on Energy, Research and Technology and the Economic and Monetary Affairs Committee.[42]

35. SAGE, Questions and Answers on the Legal Protection of Computer Programs 2 (on file with the authors). *See also* Industry Statement in Support of SAGE, *supra* note 27, para. 5.

36. In addition, ECOSOC issued an Opinion on the Proposed Directive on October 6, 1989. EUR. PARL. DOC. (CES 538) 89 (1989).

37. Mr. Karel Pinxten of Belgium is the chair of the Economic and Monetary Affairs Committee.

38. *See, e.g.*, Report Drawn Up on Behalf of the Committee on Legal Affairs and Citizens Rights on the Proposal from the Commission to the Council for a Directive on the Legal Protection of Computer Programs, EUR. PARL. DOC. (SYN 183-A 3-173/90/Part A) Annex I (1990) (Opinion of the Committee on Economic and Monetary Affairs and Industrial Policy).

39. Mr. Amedee Turner, a British intellectual property lawyer, chairs the Energy, Research and Technology Committee.

40. Report Drawn Up on Behalf of the Committee on Legal Affairs and Citizens Rights on the Proposal from the Commission to the Council for a Directive on the Legal Protection of Computer Programs, EUR. PARL. DOC. (SYN 183-A 3-173/90/Part A) Annex I, para. 4 (Opinion of the Committee on Energy, Research and Technology) (1990).

41. Mme. Margarida Salema, a Portuguese lawyer and law professor, chaired the Legal Affairs Committee.

42. *See supra* note 36.

C. The Commission Services Proposal

In April 1990, the Commission Services[43] agreed that the Proposed Directive unduly restricted reverse analysis, and, therefore, did not adequately accommodate the need to achieve interoperability. Although the Commission did not adopt a provision permitting reverse engineering without limitation, the Commission Services agreed upon a proposal (Commission Services Proposal) allowing reverse analysis with certain limitations.[44]

The Commission Services Proposal had two components, each of which addressed a separate type of reverse analysis. The first component dealt with all types of reverse analysis except decompilation.[45] Specifically, the Commission Services Proposal added a provision to the existing language of the Proposed Directive, which addressed a broad category of analysis known as black box analysis.[46] The Proposed Directive provided that:

> [T]he legitimate user of a copy of a program shall be entitled, without the authorization of the right holder, to observe, study or test the functioning of the program in order to determine the ideas, principles and other elements which underlie the program and which are not protected by copyright if he did so while loading, displaying, running, transmitting or storing the program.[47]

This provision was intended to permit any form of reverse analysis short of decompilation.

The second part of the new proposal permitted decompilation, but only to the extent that it was necessary for interoperability. This provision provided that:

> When a modification of the form of the code in which a copy of a program has been made available is indispensable to ensure that interoperable programs can be created, can be maintained or can function, and insofar as such a modification performed by the legitimate possessor of a copy of that program has been strictly limited to the parts of the program necessary to attain this goal, such a modification shall be authorized notwithstanding contractual provisions to the contrary, unless the rightsholder proves that such a modification unreasonably

43. The Commission Services for the Proposed Directive included the three Directorates-General who considered the proposal, the Directorate-General for External Affairs, and the Commissions Legal Service.

44. Commission Draft Proposal, art. 5.3(A) (unpublished proposal, on file the author).

45. There is no standard meaning for decompilation; however, this term was used in the Software Directive debates to cover any analytical technique involving the translation of a program's object code into something akin to assembly code or high-level source code. *See* Software Directive, *supra* note 1.

46. Black box analysis includes test runs, communication line traces, storage media dumps, and studying hexadecimal object code on computer screens.

47. *See* Commission Draft Proposal, *supra* note 44.

prejudices his legitimate interests or that it conflicts with a normal exploitation of the computer program.[48]

These two new provisions legitimized the basic thrust of the procompetition group's position, and they also gave rise to new debates on some important issues. First among these issues was whether the exception for interoperability would apply only to the development by third parties of programs that would attach to the program being studied, or also encompass the development of programs that serve the same function as the software being analyzed. If the latter interpretation was adopted, new programs could, essentially, compete with the more established programs.[49]

The Legal Affairs Committee and the European Parliament based their amendments relating to reverse analysis on the Commission Services Proposal, and both retreated to politically safe ground with respect to interfaces by taking a middle position between SAGE (which largely though not entirely defended the original draft) and ECIS (which wanted it made explicit that interface specifications are not protected). The Parliamentary amendment on interfaces reverted to the general proposition that only expression, and not ideas, was protected by copyright:

> Protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any aspect of a program, *including its interfaces*, shall not be protected by copyright under this Directive.[50]

Nonetheless, this language was an advance for ECIS, since it originally was suggested in the Proposed Directive that interface specifications might in some circumstances constitute protected expression.[51]

During Parliament's first reading[52] of the Proposed Directive in July 1990,[53] several amendments that were intended to prohibit reverse analysis for the creation of competing products were defeated. Instead, Parliament adopted an amendment intended to permit reverse analysis.[54] The

48. *Id.* art. 5bis(1).

49. *See supra* note 46.

50. EUR. PARL. DOC. (A3/173/1) 3 (1990) (Amendment No. 3 tabled by the Committee on Legal Affairs and Citizens' Rights, text amended by Parliament) (emphasis added) [hereinafter Legal Affairs Committee Amendments].

51. Proposed Directive, *supra* note 17, art. 1(3).

52. European Parliament's First Reading of the Proposal for a Directive on the Legal Protection of Computer Programs, EUR. PARL. DOC. (A3) 173 (July 5, 1990).

53. *Id.* amend. 34.

54. The Commission Services Proposal had provided that decompilation could not be engaged in for purposes of creating a "substantially similar program." Commission Draft Proposal, *supra* note 47, art. 5.3(A). The European Parliament provided instead that "the information retrieved [through decompilation] may not be used to create or market a substantially similar program." Legal Affairs Committee Amendments, *supra* note 50, at 34. This change was intended to permit the development of competing products through decompilation while prohibiting piracy. The Parliament also defeated

amendments favorable to reverse analysis were supported by all political groups, except part of the Christian Democratic Group.

D. Amended Proposal and Common Position

1. *Before the Council of Ministers.* Pursuant to the cooperation procedure,[55] the European Parliament forwarded its opinion[56] on the first reading to the EC Council of Ministers. During the debates in the Council, France, Germany, and Greece led the group favoring the views of ECIS, while Ireland and, particularly, Denmark favored the views articulated by the SAGE group.

The Commission exerted significant influence on the Council during the debate. In the autumn of 1990, the Commission drafted and submitted to the Council an Amended Proposal for the Software Directive (Amended Proposal), which endorsed — or at least acquiesced in — the substance of most of the amendments recommended by the European Parliament.[57] In particular, the Amended Proposal prohibited the use of information obtained through decompilation when it was used for the creation of programs "substantially similar *in . . . expression*" to the original program.[58] This provision would prohibit developers from creating programs that violated the copyright of the original program, but it would not prohibit decompilation for developing noninfringing competing products.

In October 1990, the United Kingdom proposed an amendment to the decompilation provision of the Amended Proposal.[59] The amendment, if adopted, would have permitted decompilation only to achieve interoperability "with the original program."[60] This language would support the argument that permitted decompilation only for the development of attaching, but not competing, products.

---

amendments that would have prohibited decompilation except to achieve interoperability with the decompiled program. *See, e.g.*, Legal Affairs Committee Amendments, *supra* note 50, at 17-18, 21, 23 (Amendments submitted to the European Parliament by Messrs. Jannsen van Raay and Garcia Amigo).

55. *See supra* note 11.

56. Upon the first reading, Parliament issues an opinion which typically proposes amendments to the draft directive. The opinion is submitted to the Commission and Council of Ministers for further consideration. EEC TREATY art. 140.

57. *See* Amended Proposal for a Council Directive on the Legal Protection of Computer Programs, COM(90) 509 final-SYN 183.

58. *Id.* art. 5a(2)(c) (emphasis added).

59. Consolidated Text Proposed Council Working Group, Oct. 25, 1990 (confidential document, copy on file with the authors).

60. *Id.* at 5.

The procompetition forces prevailed in the intense lobbying efforts that followed the proposal submitted by the United Kingdom.[61] On December 13, 1990, the Council of Ministers adopted a Common Position[62] which replaced the controversial language from the United Kingdom proposal (i.e., "with the original program") with a provision authorizing decompilation where necessary to achieve interoperability with other programs.[63]

2. *Parliament's Second Reading.* The Common Position was sent to the Parliament for the second reading. Accompanying the Common Position was a communication to the Parliament from the Commission.[64] The language of the communication made it clear that decompilation would be permitted for the purpose of creating competing products. The communication stated that:

> [d]ecompilation is permitted by Article 6 to the extent necessary to ensure the interoperability of an independently created computer program. Such a program may connect to the program subject to decompilation. *Alternatively it may compete with the decompiled program and in such cases will not normally connect to it.*[65]

By the time of the second reading,[66] it was evident that the momentum of the debate had shifted to the procompetition group. There were no serious efforts by SAGE or others to propose significant restrictions on the decompilation provision. It was clear that the compromise reached on interfaces by the Parliament on first reading (and incorpo-

61. Some members of SAGE broke ranks and supported the deletion of the controversial four words from the U.K. amendment. *See* Business Software Alliance, BSA Supports European Commission Software Proposal and Calls for Compromise to Complete EC Directive (Press Release, Nov. 6, 1990).

62. The Council of Ministers considers the Parliament's opinion and reaches a common position. The common position is then communicated to the European Parliament for a second reading. EEC TREATY arts. 100a (as amended 1987), 148 (as amended 1987), 149 (as amended 1987). Of the 76 total votes, 54 are necessary to achieve a qualified majority. Votes in the Council are roughly weighed according to population, with the larger member states (U.K., Germany, Italy, and France) having 10 votes, while the other member states have between 2 and 8 votes. *Id.* art. 148 (as amended 1987).

63. Council Common Position 10652/1/90 with a View to the Adoption of a Directive on the Legal Protection of Computer Programs, art. 6.

64. Communication from the Commission to the European Parliament, EUR. PARL. DOC. (SEC 87 final-SYN 183) (1991).

65. *Id.* at 5 (emphasis added).

66. On the second reading, under Article 149(2), the Parliament may reject the common position. The Parliament may propose amendments to the common position but only by vote of an absolute majority of its members. Any amendments proposed by Parliament may be adopted in the final directive only if the Commission and a qualified majority of the Council of Ministers agree with the Parliament. If the Commission disagrees with Parliament's amendments, the amendments may be adopted only by unanimity in the Council of Ministers. In practice, therefore, determination of the common position is critical and the Commission generally has a veto over the amendments adopted by the Parliament's second reading. EEC TREATY art. 149(2)(c) (as amended 1987).

rated into the Common Position), would remain unchanged; none of the EC decision makers wished to take a more definitive stand. They preferred to let the member states resolve the issues concerning the scope of protection for interfaces. Although the Legal Affairs Committee proposed several clarifying amendments to the Common Position, none were approved by an absolute majority of the European Parliament as required by the EEC Treaty.[67] Subsequently, the Council of Ministers, with the full support of the Commission, enacted the Common Position on May 14, 1991.[68]

III. THE SOFTWARE DIRECTIVE PROVISIONS

As enacted, the Software Directive begins with the rights given to a copyright owner and then establishes certain exceptions to those rights. These exceptions are the key to the Directive's real impact. Article 4(a) provides that the owner of a program has the exclusive right to engage in or to authorize "the permanent or temporary reproduction of a computer program by any means and in any form, in part or in whole."[69] It further provides that "[i]nsofar as loading, displaying, running, transmision [sic] or storage of the computer program necessitates such reproduction, such acts shall be subject to authorization by the rightsholder."[70]

A. Black Box Analysis

Black box analysis, which consists of reverse analysis techniques short of decompilation, generally involves some of the acts mentioned in Article 4 of the Software Directive. For example, studying object code requires displaying the code on a screen or on paper. However, Article 4 does not say that all acts of loading, displaying, running, transmission, and storage constitute reproductions, and it is unclear precisely when particular acts of this nature are to be considered reproductions.[71]

67. *Cf.* Recommendation of the Committee on Legal Affairs and Citizens' Rights on the Common Position Established by the Council with a View to the Adoption of a Directive on the Legal Protection of Computer Programs, EUR. PARL. DOC. (A3-0083) 2 (1991).

68. Software Directive, *supra* note 1.

69. *Id.* art. 4(a).

70. *Id.*

71. Some commentators consider that an appropriate guideline for interpreting the term "reproduction" would be the interest of the rightsholders in participating in the legitimate economic benefits resulting from the use of their programs. Thus, an act which might technically be considered to be a reproduction, but which would not lead to an increased use of the program — in the sense of using a multiplicity of copies — would not be considered a reproduction for purposes of copyright law. *See, e.g.*, Thomas Dreier, *The Council Directive of 14 May 1991, on the Legal Protection of Computer Programs*, 9 EUR. INTELL. PROP. REV. 319, 321 (1991); Michael Lehmann, *Der Neue Europaïsche Rechtschutz von Computerprogrammen*, 34 NEUE JURISTISCHE WOCHENSCHRIFT 2112 (1991). Michael Leh-

Even if the process of black box analysis involves reproductions within the scope of Article 4, those acts generally will be permitted by Article 5(3) of the Directive. Article 5(3) provides:

> The person having a right to use a copy of a computer program shall be entitled, without the authorization of the rightsholder, to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.[72]

Article 5(3) was proposed with the intent of permitting all forms of reverse analysis short of the more controversial practice of decompilation. An examination of the individual elements of Article 5(3) will illustrate how it can be interpreted with this original intent in mind.

First, Article 5(3) requires a developer to have "a right to use a copy of the computer program . . ." that he intends to analyze.[73] This requirement should not present any barrier to legitimate interoperable product developers, as it only prohibits the use of a pirated copy of the program. While it may be expected that most interoperable product developers will be licensees of the program they intend to analyze, it would be permissible to analyze a copy of the program in the legitimate possession of someone else, provided that the interoperable product developer was granted the right to use the program.

Second, Article 5(3) permits the software engineer to "observe, study, or test the functioning of the program. . . ."[74] This is precisely what an engineer does when conducting black box analysis. For example, when performing line traces, the engineer causes messages to be transmitted from the analyzed program to another program or device and uses a line tracer device to determine how the program interacts or functions with the other program or device. Similarly, an engineer will observe and study screen displays of the hexadecimal object code of a program to determine how the program functions with other programs or devices.

Third, Article 5(3) allows the software engineer to determine the "ideas and principles which underlie any element of the program. . . ."[75] While Article 5(3) is not limited to determining interface specifications, the language of the article would allow a developer to ascertain them because they constitute unprotected ideas and principles.

---

mann, *Die Europaïsche Richtline uber den Schutz von Computerprogrammen*, GRUR INT., May 1991, at 327.

72. Software Directive, *supra* note 1, art. 5(3).

73. *Id.*

74. *Id.*

75. *Id.*

Fourth, Article 5(3) permits the software engineer to observe, study, or test the functioning of the program while "loading, displaying, running, transmitting or storing the program. . . ."[76] While the particular acts involved should be evaluated in each individual case, it appears that all forms of black box analysis may be said to be conducted while performing one or more of the procedures listed in Article 5(3). For example, studying an object code on a computer screen takes place while displaying the program. Likewise, conducting a test run involves running the program.

Finally, Article 5(3) provides that one must be "entitled to do" the acts involved.[77] Although awkwardly worded, the section is an attempt to ensure that a developer adheres to the spirit of any pertinent license agreement, preventing the illegitimate expansion of otherwise authorized acts. For example, one may not seek to use a program on a machine not designated in a license agreement on the ground that this was merely being done in order to "observe, study or test the functioning of the program."[78]

It is important to note that the exception provided for in Article 5(3) cannot be eliminated by contract.[79] Thus, a license restriction limiting the running of a program to data processing purposes, and prohibiting the running of the program for purposes of determining the program's underlying ideas and principles, is unenforceable. As long as one is entitled to run the program, one may not be prohibited from running it in conjunction with observing, studying, and testing the functioning of the program to determine its underlying ideas and principles.

B. Decompilation

Article 6 of the Software Directive addresses the issue of decompilation.[80] Article 6 permits decompilation for purposes of developing competing as well as attaching programs. As the Commission stated in its

76. *Id.*

77. *Id.*

78. *Id.*

79. Article 9(1) provides, *inter alia*, that "[a]ny contractual provisions contrary to Article 6 or to the exceptions provided for in Article 5(2) and (3) shall be null and void." *Id.* art. 9(1).

80. Article 6 states:

Decompilation

1. The authorization of the rightsholder shall not be required where reproduction of the code and translation of its form within the meaning of Article 4(a) and (b) are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs, provided that the following conditions are met:

(a) these acts are performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorized to do so;

(b) the information necessary to achieve interoperability has not previously been readily available to the persons referred to in subparagraph (a); and

Twentieth Report on Competition Policy, a program developed through decompilation "may compete with the decompiled program and in such cases will not normally connect to it."[81] The Software Directive does not, however, permit a developer to decompile a program simply because the developer wishes to create a product to compete with the program in some general way unrelated to interoperability. In particular, a developer may not decompile a program solely to research its underlying ideas, such as a novel algorithm, and then implement those ideas in a program that competes with the decompiled program.[82]

Moreover, pursuant to Article 6, the decompilation must be indispensable and must meet the conditions of subparagraphs 1(a), (b), and (c). However, the term indispensable is not defined in the Software Directive.[83] While it was clearly meant to impose a strict requirement of technical necessity, it must be assumed that the courts will strike a reasonable balance and will interpret the term in light of existing practice. Presumably the courts will not require that developers engage, for example, in endless amounts of black box reverse analysis before decompilation is justified. Such a requirement would place European developers at a disadvantage in relation to their overseas competitors and would impede economic efficiency.[84]

(c) these acts are confined to the parts of the original program which are necessary to achieve interoperability.

2. The provisions of paragraph 1 shall not permit the information obtained through its application:

(a) to be used for goals other than to achieve the interoperability of the independently created computer program;

(b) to be given to others, except when necessary for the interoperability of the independently created computer program; or

(c) to be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright.

3. In accordance with the provisions of the Berne Convention for the protection of Literary and Artistic Works, the provisions of this Article may not be interpreted in such a way as to allow its application to be used in a manner which unreasonably prejudices the rightsholder's legitimate interests or conflicts with a normal exploitation of the computer program.

Software Directive, *supra* note 1, art. 6.

81. Commission of the European Communities, Twentieth Report on Competition Policy (1991).

82. Jean-François Verstrynge, *Protecting Intellectual Property Rights Within the New Pan-European Framework—Computer Software*, 5 Int'l Computer L. Adviser (forthcoming June 1991) [hereinafter *Protecting Intellectual Property Rights*] (manuscript at 13, on file with authors).

83. *Id.* at 12. Nor is there any clear indication of who bears the burden of proof on this or the other issues raised by Article 6. The question of burden of proof is presumably left to each member state's law.

84. In practice, the indispensability requirement should not present a problem, since decompilation requires great sophistication, time, and expense. As noted by the Commission in the Explanatory Memorandum accompanying its proposal for the Software Directive, "[a]lthough it is technically possible to decompile a program in order to find out information concerning access protocols and interfaces this is a lengthy, costly and inefficient procedure." Proposed Directive, *supra* note 17,

With respect to the other conditions contained in Article 6(1), subparagraph (a) requires that a developer be a licensee of the program or be authorized to use the program by a licensee or by someone else entitled to use the program.[85] This condition is not unreasonable and will be met by all legitimate developers of compatible products.

Subparagraph (b) simply requires that the necessary interface information "has not previously been readily available" to the developer.[86] While this condition will have to be interpreted by the courts on a case-by-case basis, it is unlikely to impose an unreasonable burden on developers. As noted by the Commission, decompilation is lengthy, costly, and inefficient.[87] Therefore, decompilation is only engaged in as a last resort. It will not, in practice, be conducted if the interface information is readily available. While the rightsholder of a program may offer to sell interface information relating to his program, it does not appear necessary for a developer to request the information from the rightsholder before engaging in decompilation.[88] Several formulations imposing such a requirement were considered by all three EC decision making institutions, and each was rejected.[89] As long as the information is not contained in published documentation that is easily accessible to a developer, he very likely will meet the condition imposed by subparagraph (b).

Pursuant to subparagraph (c), a developer must confine his decompilation to those parts of the analyzed program which contain the interface information he requires to make his own independently created, interoperable program.[90] This will again require a case-by-case analysis. It should not, however, place an unreasonable restriction on the developer of interoperable products. The expense, inefficiency, and difficulty of decompilation will inhibit decompilation of portions of a program that do not contain information necessary to achieve a developer's interoperability objectives.

Article 6(2) imposes limits on what may be done with the results of decompilation. First, subparagraph (a) provides that information gained

para. 3.14 (Explanatory Memorandum). Thus, software engineers generally will not engage in decompilation unless it is indeed "indispensable."

85. Software Directive, *supra* note 1, art. 6(1)(a).

86. *Id.* art. 6(1)(b).

87. Proposed Directive, *supra* note 17, para. 3.14 (Explanatory Memorandum).

88. "Nothing in the Directive prevents the parties from entering into an Agreement for the supply of the information, *even if access has to be granted under the conditions of Article 6.*" *Protecting Intellectual Property Rights*, *supra* note 82, at 12 (emphasis added). This suggests that access must be granted, assuming that other conditions of Article 6 are met, apart from any attempts to reach agreement on provision of the information.

89. *See, e.g.*, the proposal made by the Irish delegation to the Council Working Group dated May 2, 1990 (SN/2382/90) (PI).

90. Software Directive, *supra* note 1, art. 6(1)(c).

through decompilation (unlike the information obtained pursuant to reverse analysis conducted under Article 5(3)) may be used only "to achieve the interoperability of the independently created computer program."[91] In other words, a developer may use the information obtained from decompiling a program, implement it in a new program, and allow the new program to operate with other programs in the same way as the original program. The developer may not, as noted above, decompile the original program in the course of general research, using any information gained thereby for purposes unrelated to interoperability.[92]

Second, subparagraph (b) provides that a developer may not give any of the interface information derived through decompilation to others, except when necessary for the interoperability of his independently created program.[93] For example, a developer may include in his customer manuals interface information which could be otherwise obtainable by users through decompilation; users could then implement this prepackaged information into their own operating system — to the extent customers need such information for purposes of ensuring that their application programs will run with the new operating system. While this restriction appears to be unnecessary, it should not prove to be an obstacle to interoperable product developers, who have no need to provide interface information to others for purposes unrelated to the interoperability of their products.

Third, subparagraph (c) does not permit information obtained through decompilation to be used "for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright."[94] The formulation of this paragraph is extremely important. The groups which sought to prohibit decompilation for purposes of creating products that would compete with the decompiled program attempted during the pre-enactment process to impose the requirement that decompilation would not be allowed to create a "substantially similar" program.[95] A competing product (such as a compatible operating system) will arguably be similar in function to the program with which it operates, and therefore, such a formulation might not have permitted information obtained through decompilation to be

91. *Id.* art. 6(2)(a).

92. *See id.* art. 6(2)(a). Some may regard the limitation in subparagraph (a) as inconsistent with traditional copyright law because, by precluding access to them, it results in the *de facto* protection of ideas (at least ideas unrelated to interoperability). However, the limitation should not hinder the development of interoperable products, where decompilation is necessary only to obtain interface information.

93. Software Directive, *supra* note 1, art. 6(2)(b).

94. *Id.* art. 6(2)(c).

95. *See, e.g.*, Legal Affairs Committee Amendments, *supra* note 50, at 21, 23.

used to create competing products. The attempts to impose this formulation were defeated, however, and the traditional copyright test, one of substantial similarity in expression, was retained.

The third paragraph of Article 6 merely provides, in effect, that Article 6 may not be interpreted inconsistently with Article 9(2) of the Berne Convention for the Protection of Literary and Artistic Works.[96] While the inclusion of this language from the Berne Convention appears to be surplusage,[97] it was apparently intended to limit undue extensions of the boundaries imposed by the text of Article 6.[98] While paragraph 3 is a potential wild card, it seems unlikely to present any serious problems to interoperable product developers.[99]

One final aspect of Article 6 should be noted. As with Articles 5(2) and 5(3), "[a]ny contractual provisions contrary to Article 6 . . . shall be null and void."[100] The purpose of this provision is straightforward. It is designed to prevent companies that have market power or similar leverage from undoing the delicate balance reached in the Software Directive by routinely including in their license provisions overriding Article 6.[101]

C. The Scope of Interface Protection

The provisions of the Software Directive on interfaces are far less explicit than those governing reverse analysis. Interfaces are defined as "the parts of the program which provide for . . . interconnection and interaction between elements of software and hardware. . . ."[102] Further, Article 1(2) of the Software Directive restates the traditional copyright doctrine that "[i]deas and principles which underlie any element of a computer program, *including those which underlie its interfaces*, are not protected by copyright. . . ."[103]

96. Software Directive, *supra* note 1, art. 6(3) (referring to the Berne Convention for the Protection of Literary and Artistic Works, Sept. 9, 1896, as revised at Paris, July 4, 1971, 168 CONSOL. T.S. 185). Article 9(2) of the Berne Convention states that:

> It shall be a matter for legislation in the countries of the Union to permit the reproduction of such works in certain special cases, provided that such reproduction does not conflict with a normal exploitation of the work and does not unreasonably prejudice the legitimate interests of the author.

Berne Convention, *supra*, art. 9(2).

97. *See* Dreier, *supra* note 71, at 325.

98. *See Protecting Intellectual Property Rights*, *supra* note 82, at 14 ("We do not want to see the boundaries which we have drawn extended in a way which would cause the exception to fall outside the scope of Article 9.2 of the Berne Convention.").

99. *See* W.R. Cornish, *Computer Program Copyright and the Berne Convention*, 4 EUR. INTELL. PROP. REV. 129 (1990) (reverse analysis is consistent with the Berne Convention as long as the fruits of reverse analysis are not used to reproduce substantially the expression of the analyzed program).

100. Software Directive, *supra* note 1, art. 9(1).

101. *See Protecting Intellectual Property Rights*, *supra* note 82, at 14.

102. Software Directive, *supra* note 1, pmbl., para. 15.

103. *Id.* art. 1(2) (emphasis added).

Under Article 1(2), a developer will have to determine in every case where to draw the line between idea and expression. In general, as long as only the rules and methods of interoperability established by the interface are used and implemented independently in the program code, the program should be held to be noninfringing under the Software Directive.[104]

There is, however, one further twist: to make an interface work, meaning to make a product interoperable, it is sometimes necessary to use small portions of a program code that are very similar or identical to expressions found in the program code of existing copyrighted products. The Software Directive does not explicitly address this issue, and there is room for divergence among the member states' laws on this point. However, while the precise legal theory employed may vary from country to country, it seems likely that such similarities in expression will be deemed noninfringing. There are at least three theories potentially available to justify such similarities.

First, *de minimis* uses of identical code may not amount to infringement under a member state's copyright law. Such uses would not render a program substantially similar to another program in the copyright sense.[105] Second, under continental laws in particular, interface implementations may not, given their typically short length and arbitrariness, demonstrate sufficient originality to warrant copyright protection.[106] In this case, compatible developers would be free to use the code to achieve interoperability without copyright infringement. This may be particularly true where the interface implementation is function-dictated, for example by the need for a compatible operating system to achieve interoperability with third party applications. Third, under the so-called merger doctrine,

104. It would be prudent to continue using clean room methods for the development of competing products. Under the clean room approach to development, one software engineer disassembles the original program and determines its interface specifications. This engineer then prepares a document detailing the interface specifications, which he gives to a clean engineer, who has had no access to the analyzed program's code, either in object code or otherwise. This engineer then independently implements the interfaces in his own noninfringing code. This procedure provides an access defense. The developer can defend against claims of infringement by demonstrating a lack of access by the clean engineer to the original program's code, and can justify similarities in code on the basis that they were independently developed and hence not copied. For more detailed descriptions of clean room procedure and their legal basis, see David L. Hayes, *Acquiring and Protecting Technology: The Intellectual Property Audit*, COMPUTER LAW., Apr. 1991, at 1, 18-19; Douglas K. Kerwin & Daniel R. Siegel, *Microcode Copyright Infringement*, COMPUTER LAW., Apr. 1987, at 1, 7-8.

105. It is of course true that substantial similarity is judged on the basis not of the *quantity*, but of the *quality* of expression reproduced. *See* Ladbroke (Football) Ltd. v. William Hill (Football) Ltd., 1 W.L.R. 273 (Eng. 1964); L.B. (Plastics) Ltd. v. Swish Products Ltd., 1979 R.P.C. 551 (Eng.). However, this doctrine is likely to work in the compatible developer's favor, as most interfaces, whether intended for use as such by the original developer or not, are relatively arbitrary and lacking in creativity. Compatible developers use similar expression not because it embodies any particular creativity, but because its use is functionally dictated by the need to achieve interoperability.

106. Dreier, *supra* note 71, at 325.

which the Commission has acknowledged is a general concept of copyright law,[107] idea and expression may have merged. Therefore, using the required code will not infringe the copyright in the original program. In particular, merger will occur when "the constraints of the interface are such that in the circumstances no different implementation is possible" to achieve interoperability.[108]

Given the commitment of the EC to promote interoperability, as evidenced by the very existence of the decompilation provision, it can be expected that the member states' courts will interpret the national legislation adopted under Article 1(2) in a relatively liberal manner. It would be incongruous for the EC to have gone to the very substantial effort of devising a decompilation provision, only to render it ineffective by according an overly expansive scope of protection to interfaces.

IV. CONCLUSION

Adoption of the Software Directive was an extended and hard fought procedural exercise that introduced the EC to American-style lobbying on a massive scale.[109] There are a number of substantive questions that currently have less than complete answers, however, in some respects, the contours of the Software Directive will be filled in during the implementation process in the various member states and through court decisions. While a rough balance has been achieved, the Software Directive clearly avoids adopting measures which restrain competition that had been proposed at various points during the process. On a broad scale the Directive, as enacted, is far more favorable to the procompetition forces than was the original draft Directive. Any implication that interface specifications are capable of being protected is absent from the final version of the Software Directive. Moreover, reverse analysis is explicitly authorized, and even decompilation is permitted where it is necessary to achieve

107. In stating that it was unnecessary to adopt a Parliamentary amendment on the merger doctrine, Vice President of the Commission Martin Bangemann set forth the Commission's view that the proposed amendment "is supposed to reflect the 'merger doctrine,' according to which there is no copyright protection where an idea and its expression cannot be separated. . . . [This doctrine] is a permanent feature of copyright law [and] we do not need to mention it in the directive." EUR. PARL. DEB. (3-404) 56, 57 (Apr. 16, 1991). In effect, Mr. Bangemann merely confirmed what the Commission said in the Explanatory Memorandum accompanying its original proposed directive:

> If similarities in the code which implements the ideas, rules or principles occur as between inter-operative programs, due to the inevitability of certain forms of expression, where the constraints of the interface are such that in the circumstances no different implementation is possible, then no copyright infringement will normally occur, because in these circumstances it is generally said that idea and expression have merged.

Proposed Directive, *supra* note 17, para. 3.13 (Explanatory Memorandum).

108. Proposed Directive, *supra* note 17, para. 3.13 (Explanatory Memorandum).

109. *See, e.g.*, Alan Cane, *Computer Users Fight EC Software Directive*, FIN. TIMES, Sept. 10, 1990, at 4.

interoperability. In light of the Commission's official statements (both in its communication to the European Parliament and in its Competition Policy Report), the critical competing-products issue also has been resolved so as to permit the development of such products through any and all types of reverse analysis where necessary.

The impact of these results on the future development of the law in the United States and Japan should not be ignored. Relatively few judicial decisions in either jurisdiction address, for instance, the lawfulness of reverse analysis. When cases presenting this issue arise in the future, one side or the other is likely to cite to the Software Directive as useful precedent for their position, since it deals with the permissibility of reverse analysis in various circumstances and for various purposes. In this respect, the battle over the Software Directive and its outcome may affect the application of copyright law to software worldwide as well as in the EC itself.[110]

110. The International Bureau of the World Intellectual Property Organization, in a July 18, 1991 memorandum, proposed the use of language similar to that of the Software Directive in a possible Protocol to the Berne Convention for the Protection of Literary and Artistic Works. World Intellectual Property Organization, Committee of Experts on a Possible Protocol to the Berne Convention for the Protection of Literary and Artistic Works, First Session, November 4 to 8, 1991, Questions Concerning A Possible Protocol to the Berne Convention, *Part I*, July 18, 1991 (prepared by the International Bureau) (on file with authors).