Copyright Protection and Reverse Engineering of Software: Implementation and Effects of the EC Directive Symposium

Japp H. Spoor
Vrije University

COPYRIGHT PROTECTION AND REVERSE ENGINEERING OF SOFTWARE: IMPLEMENTATION AND EFFECTS OF THE EC DIRECTIVE

*Jaap H. Spoor**

I. Introduction

Reverse engineering and decompilation of software have received considerable attention in recent years. The stakes are high in the area of computer technology and programming. On the one hand, software needs to be protected against copying and unfair imitation. On the other hand, most software also needs to possess the capability of interacting with other software. To that end, computer programs must be carefully adapted to other programs' interface specifications. The interface specifications often may only be obtained through analysis of the relevant parts of those other programs.

From a technological viewpoint, the United States and Europe have similar, if not identical, interests concerning software. Some might argue that most of the commercially successful software programs were developed by U.S. companies, who therefore might be more interested in protection than in competition. One should take into account, however, that the U.S. software industry consists of more than just a few major players. If MS-DOS or Windows have become a success, it is largely because millions of IBM PC compatible computers were produced, and because hundreds of independent software developers created thousands of application programs for those platforms. Moreover, even IBM, Microsoft, Lotus, and other market leaders are at times followers. Consequently, they also need to continuously adapt and enhance their products and stay compatible with important new developments. In other words, each program developer must make and keep his products compatible as well as protect them from infringement. Of course, Europe is not altogether without major players either.

Most likely the differences between the United States and the European Community ("EC") do not lie as much in the facts as they do in the law. Unlike the U.S. Copyright Act, the EC Software Directive (the "Directive") explicitly addresses the issue of software reverse engi-

* Professor of Intellectual Property Law, Vrije Universiteit, and Attorney, Trenité Van Doorne, Amsterdam, The Netherlands. M.A. 1966, Vrije Universiteit; J.D. 1976, University of Utrecht.

neering, thereby obligating the EC member countries to do the same in their respective national copyright laws. This Article discusses the software reverse engineering and compatibility issues from a continental European viewpoint, with the Directive as a central point of focus.

II. Software Protection In The EC Countries: General Remarks

A. *Copyright in the EC: A Matter of National Laws*

Seen from a distance, the EC[1] may appear to be a U.S.E. (United States of Europe), certainly different from the U.S., but nevertheless a kind of federation. Tempting as the comparison may be, such a picture would be rather far from present day reality. Unlike the U.S., which has had a federal government for over two centuries, the EC countries at best are on their way to forming a federation. The EC, however, has a long way to go and success is as yet far from guaranteed. Currently in the EC, notwithstanding the impact of European Community law, national law prevails in most fields. Such is the case with copyright law.

While U.S. copyright law is regulated by the Federal Copyright Act, EC copyright is administered by twelve separate national laws. This is true, in spite of the fact that several EC Directives, including the Software Directive, are aimed at harmonizing these issues. All EC Member countries have been parties to the Berne Convention ("BC") for a considerable time. In spite of the fact that their membership to that Convention has certainly had a harmonizing effect on their national copyright laws, differences in the laws still abound. Such differences arise not only between the common-law countries (i.e., U.K. and Ireland), on the one hand and the ten continental, so called civil-law countries on the other, but the law also varies from one civil-law country to another. In fact, the civil-law concept refers mainly to certain corresponding tendencies in legal thought, the structure of laws, and the administration of justice. Civil law in no way guarantees any uniformity beyond that level. In consequence, the differences between continental European national laws are as varied and major as are those between U.K. and U.S. laws, if not more so.

If the EC has at least one problem which does not have a counterpart in the U.S., it is the language barrier. Between them, the ten con-

1. As a result of the enactment of the Maastricht Treaty on November 1, 1993, the denomination "European Union" ("EU") has been introduced as an alternative to the more traditional "European Community" ("EC") or "European Economic Community" ("EEC"). As it is not yet clear which expression will prevail, the expression "European Community" ("EC") will be used in this paper.

tinental EC countries boast eight different national languages.[2] This fact alone makes the harmonization of national laws in the EC a far more laborious task than the harmonization of state laws in the U.S.[3]

B. *Software Protection Before the Directive*

Even before the Directive came into force in 1991, all EC member states granted copyright protection to computer software under their national copyright laws. In most countries, the national copyright legislation had already established this protection. In other countries such as the Netherlands, the protection followed from a large body of reported court decisions. Much less harmony, however, existed as regards the conditions and effects of such protection.

Perhaps the most important difference between the EC continental copyright laws concerns the concept of originality. Since originality is of primary importance, for the subject of software copyright in general, and also for the compatibility issue, as well, it must be addressed.

C. *Originality Under National Law*

All European copyright laws require a work to be original, at least as a general principle, but the implementation of this concept varies greatly. All continental laws require a work, including software, to be the fruit of personal creative labor. This certainly does not exclude the use of automated tools in the creation of a work, but the personal aspect must not be wholly absent. On the other hand, in most countries the threshold for originality is probably fairly low. For instance, under French or Dutch law, originality can be described by the maxim that a work is original if it must be considered impossible, or even highly unlikely, that it could independently be created twice in more or less identical form. There are essentially two exceptions to this mainstream interpretation of the originality concept: the U.K. interpretation and the German interpretation.

The U.K. approach is generally understood to require a lesser amount of personal creativity than required by French law. In U.K. law, a work is considered to be original if it is the fruit of an author's

2. These include Danish, Dutch, French, German, Greek, Italian, Portuguese, and Spanish; not to count a number of regional languages. Fortunately, Belgium and Luxembourg have the courtesy to share languages with their neighbors.

3. Incidentally, the language barrier also offers a serious impediment to one trying to describe the existing situation. Many, if not most books and articles are in the relevant national languages. Probably few people and even fewer lawyers have a working knowledge of all of these languages. Moreover, libraries will rarely subscribe to professional magazines in foreign languages other than English and perhaps, French or German. For this and other reasons, this paper does not pretend to give a full or even representative overview of the situation in all ten continental EC member countries.

personal labor or even of an automated process. The work need not be in any way unique, provided that it is not itself a copy from another work. On the other hand, German law has always been much more demanding. German law concerning software requires a creative input which is above the ordinary programmer's skill.[4] The effect of this requirement is that only a small minority of all computer programs may enjoy copyright protection. The requirement can also severely limit the extent to which even copyrighted programs are protected. For example, certain program routines or modules may still fail to meet the threshold requirements.[5]

As a result of the German interpretation of originality, many computer programs which enjoyed copyright protection in most countries lacked such protection in Germany, thus affecting the trade of computer software within the EC.[6] One of the main concerns of the EC is to remove trade barriers between the member states. Harmonization of the originality concept with respect to software was certainly one of the primary motives for the Directive. Other significant aspects of software copyright which are addressed in the Directive include reverse engineering and compatibility.

III. THE DIRECTIVE: MAIN ASPECTS

The Directive focuses on harmonizing the main aspects of software copyright by standardizing the required level of originality, the notion of authorship, and the restricted acts, as well as the main exceptions thereto.

A. Originality

Article 1, paragraph 3 of the Directive states that "[a] computer program shall be protected if it is original in the sense that it is the author's own intellectual creation [and] [n]o other criteria shall be applied to determine its eligibility for protection." There is little doubt that this text is meant to give a fairly low threshold for protection that is comparable to the French or Dutch levels, and far below the very demanding standard set by the German Supreme Court. It should be noted, however, that several German commentators are reluctant to ac-

4. Judgment of May 9, 1986, BGH, *reprinted in* 8 EUR. INTELL. PROP. REV. 185 (1986).

5. To do justice to this German doctrine, it should be mentioned that the doctrine was not invented just to keep software out of copyright law, rather it is the long standing and prevailing doctrine with respect to all works of a more or less technical nature.

6. In practice, this was often remedied by other forms of protection such as unfair competition law, or in the case of video games, by protecting the game as shown on the screen as a cinematographic work. Thomas Dreier, *Verletzung Urheberrechtlich geschützter Software nach der Umsetzung der EG-Richtlinie*, 95 GEWERBLICHER RECHTSSCHUTZ UND URHEBERRECHT 781, 782 (1993).

cept this conclusion. Broy and Lehmann[7] conclude that henceforth all but the most trivial programs will be protected. Dreier[8] stresses that the text of article 1 does not force this lower standard, although he admits that the discussions that led to the article indicate otherwise. Lesshaft and Ulmer[9] insist that not all programs, not even "the more complex software," will meet the standards. Moreover, they state that new trends and methods in software development, such as object-oriented programming, standardization, and the use of automated tools, will lead to software being less and less original. On the other side of the spectrum, it is not yet clear whether the U.K. will have to raise their originality threshold. The words "the author's own intellectual creation" leave room for different interpretations.

B. Reproduction

Article 4 of the Directive provides:

> The exclusive rights of the rightholder . . . shall include the right to do or to authorize [*inter alia*] the permanent or temporary reproduction of a computer program by any means and in any form, in part or in whole. Insofar as loading, displaying, running, transmission or storage of the computer program necessitate such reproduction, such acts shall be subject to authorization by the rightholder.

Although the text is somewhat circular, it is generally understood to mean that loading, displaying, running, transmission or storage is indeed reproduction. It is implemented more or less literally in most continental laws, including the German Copyright Act and the amended Dutch Bill.[10] Still, the idea that any copying, however transitory, is indeed to be considered reproduction is not universally accepted, especially by German scholars. While Bauer[11] and Haberstumpf[12] agree, others, such as Dreier[13] and especially Lehmann[14] do not. As Lehmann

---

7. M. Broy und M. Lehmann, *Die Schutzfähigkeit von Computerprogrammen nach dem neuen europäischen und Deutschen Urheberrecht*, 94 GEWERBLICHER RECHISSCHUTZ UND URHEBERRECHT 419, 423 (1992).

8. Thomas Dreier, *The Council Directive of 14 May 1991 on the Legal Protection of Computer Programs*, 12 EUR. INTELL. PROP. REV. 319, 578 (1991).

9. K. Lesshaft and D. Ulmer, *Urheberrechtliche Schutzwürdigkeit und Tatsächliche Schutzfähigkeit von Software*, 9 COMPUTER UND RECHT 607, 608 (1993).

10. The original text was even more explicit stating that *loading* is reproduction.

11. K. Bauer, *Reverse Engineering und Urheberrecht*, 6 COMPUTER UND RECHT 89, 91 (1990).

12. H. Haberstumpf, *Die Zulässigkeit des Reverse Engineering*, 7 COMPUTER UND RECHT 129, 133 (1991).

13. Dreier, *supra* note 8, at 580.

14. M. Lehmann, *Freie Schnittstellen ("interfaces") und freier Zugang zu den Ideen ("reverse engineering"). Schranken des Urheberrechtsschutzes von Software*, 5 COMPUTER UND RECHT 1057, 1062 (1989).

states, "reproduction is a legal concept which must be interpreted on a normative basis, and which should not depend on the technical state of affairs or the incidental way in which a computer happens to function."[15]

Incidentally, this is not a mere technicality as far as the law is concerned. It has been stressed time and again that software differs from other, more traditional copyright works, because it can hardly be used without being copied time and again. Consequently, the application of the broad traditional interpretation of "reproduction" of software leads to stronger protection and to users becoming more dependent on the copyright holder than is true in the case of traditional works. This author, therefore, agrees with Lehmann's statement that the concept of reproduction should not simply depend on the technical state of affairs. The way in which the traditional concept of reproduction operates in the software field is, however, not simply an anomaly which needs to be ironed out. Rather, it has come to play a key role in the whole structure of the software copyright system.

C. *Error Correction*

According to article 5, paragraph 1 of the Directive, "[i]n the absence of specific contractual provisions" the lawful user of a program may reproduce or alter the program if necessary for the use of the program in accordance with its intended purpose, including error correction. Although the text expressly provides for other contractual arrangements, it is debated whether error correction may be entirely forbidden by contract. Haberstumpf[16] asserts that under the Directive the right to use a program according to its purpose is of the essence, and that in consequence, error correction may not simply be ruled out under all conditions. The Portuguese Act altogether forbids contracting out of the provision,[17] as does the Dutch bill insofar as error correction is concerned.

Several commentators have pointed out that the error correction provision is rather vague, especially as it does not give any definition of the term "error." Indeed, it is remarkable that this subject has not led to more debate in the European Parliament. After all, it will often be impossible to correct errors without some form of decompilation. It fol-

15. M. Lehmann, *Erwiderung. Reverse Engineering ist keine Vervielfältigung i.S.d. §§ 16, 53 UrhG*, 6 Computer und Recht 94 (1990).

16. H. Haberstumpf, *Das Software-Urhebervertragsrecht im Lichte der bevorstehenden Umsetzung der EG-Richtlinie über den Rechtsschutz von Computerprogrammen*, 1992 Gewerblicher Rechtsschutz und Urheberrecht Internationaler Teil 715 (1992).

17. J.A. Veloso, *La Protección del Software en Portugal*, Derecho de la Alta Tecnología, Mar. 1993, at 14, 19.

lows from the provision that such decompilation may not exceed what is needed in order to correct the error. Both the vagueness of this notion, and the problem of knowing beforehand exactly where the correction will need to be made, may well lead to larger parts being decompiled than where decompilation takes place under the much debated article 6.

Admittedly, decompilation for error correction as a rule will not have the same impact. The user may adapt only the licensed program copies he uses, while interface information obtained under the interoperability provision will probably find its way to a commercial software product. Still, it is striking that the legislators have so perfunctorily dealt with the subject of error correction. Even a provision forbidding any further use of the information obtained in the course of permissible error correction is lacking from the law.

D. *Permissible Studying*

Article 5, paragraph 3 expressly provides:

> [The rightful user] shall be entitled, without the authorization of the rightholder, 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.

In other words, the user may try to find out from the outside how the program works.

Although it is presented as a right of the user, most commentators stress that article 5 permits something which is almost evident,[18] which can hardly be forbidden, and more important, which is of little avail to someone trying to develop compatible software. In short, the article mainly seems to serve cosmetic purposes, and the fact that in most, if not all, member countries it may not be excluded by contract can hardly alter that fact.

E. *Decompilation for Compatibility Purposes*

Article 6 permits decompilation, or rather "reproduction of the code and translation of its form," if it is "indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs." This provision

18. A. Lucas, *The Council Directive of 14 May 1991 Concerning the Legal Protection of Computer Programs and its Implications in French Law*, 14 EUR. INTELL. PROP. REV. 28, 31 (1992). "The reservation in Article 5, paragraph 3 . . . adds nothing, as regards French law, to the exception covered by Article 5, paragraph 1." *Id.*

is the essence of article 6, although the article addresses several other topics: decompilation, permissible use of information, and interpretation of article 6. Decompilation may only be performed by or on behalf of a rightful user, and only if the information "has not previously been readily available" to them. Decompilation must remain limited to "the parts of the original program necessary to achieve interoperability." Furthermore, the permissible use of the information obtained is subject to several restrictions. Finally, by its third paragraph the interpretation of article 6 may not unreasonably prejudice the rightholder's legitimate interests or conflict with a normal exploitation of the computer program. This last provision explicitly refers to the Berne Convention, and clearly echoes article 9, paragraph 2 thereof.

No article of the Directive has given rise to more debate and lobbying than article 6. Major software producers feared a protection gap, while independent software developers were afraid they would be at the copyright owners' mercy when trying to develop compatible software. As a result of this debate, the drafters made an important change in the earlier proposed text of article 6. The early text only permitted decompilation for the purpose of enabling a newly developed program to interoperate with the decompiled program. The final text permits decompilation for interoperability with other programs as well.[19] The end result has not received universal approval, to put it mildly. Of course, many of the disfavorable comments are essentially based on policy considerations. Thus, Kindermann (IBM Germany) states that article 6 will enable third parties to produce cheap imitation products. He, therefore, considers it contrary to article 9, paragraph 2 of the Berne Convention.[20] On the other hand, Vinje concludes that "while it might be argued that article 6 contains unnecessary restrictions, it does not appear that it will impede existing reverse analysis practices."[21]

A more serious problem may well lie in the complexity of the text of article 6. As Schulte puts it, much of article 6's wording is based on compromises which do not solve the conflicts of interests but rather leave them to the European Court of Justice to solve.[22] Consequently, article 6 leaves many questions to be resolved.[23] For example, the Directive gives no guidance as to how much code one may actually

19. Thomas C. Vinje, *Interoperable Product Development under the EC Software Directive*, 8 COMPUTER L. & PRAC. 190, 193 (1992).

20. Manfred Kindermann, *Reverse Engineering von Computerprogrammen. Vorschläge des Europäischen Parlaments*, 6 COMPUTER UND RECHT 638, 638 (1990).

21. Vinje, *supra* note 19, at 195.

22. D. Schulte, *Der Referentenlwurf eines Zweiten Gesetzes zur Aenderung des Urheberrechtsgesetzes. Ausgewählte Auslegungsfragen der EG-Richtlinie über den Rechtsschutz von Computerprogrammen*, 8 COMPUTER UND RECHT 648, 653 (1992).

23. M. Vivant & C. Le Stanc, LAMY DROIT DE L'INFORMATIQUE 525 (1992).

decompile in search for interfaces,[24] or is at least unclear as to how one can actually know what to decompile.[25] Furthermore, the article does not mention the subject of compatibility with hardware, while it is uncertain whether a solution can always be reached through software.[26] Finally and perhaps most important, what kind of interfaces does article 6 cover: official interfaces only, that is, such parts of a program as are considered by the rightholder as interfaces, unofficial interfaces, especially those which have gained acceptance as interfaces in the industry; or anything which may serve as an interface, that is, any spot in a program where data are transmitted without being altered? This is a much debated issue. Rightholders may fear that the reputation of their programs will be compromised by the fact that new releases may not interface with certain independently developed programs because the parts that served as an unofficial interface were changed in the new release. The other side may claim it needs to use the most efficient interfaces it can find and its creativity, as well as new developments, is impaired if it may only use what the developer of the original program thinks is appropriate.

IV. Implementation and Effects of the Directive

A. General

As previously mentioned, the Directive does not contain directly applicable law. Its provisions must be implemented in the national laws. This should have been completed in all countries by January 1, 1993, but most member states failed to meet that deadline.[27] More important, although the national implementations may not be contrary to the Directive, it is possible that the provisions will be implemented in different forms and will have different effects in various countries. It will take years for the European Court of Justice to bring these national interpretations into line. Moreover, as only part of the copyright laws is harmonized, differences may continue to exist in the areas of common restrictions to copyright, the notion of adaptation, and the in-

24. Dreier, *supra* note 8, at 582.

25. M. de Cock Buning, *Auteursrecht en 'reverse engineering'*, 9 Intellectuele Eigendom en Reclamerecht 129, 134 (Tjeenk Willink 1993).

26. Schulte, *supra* note 22, at 654.

27. In December 1993, The Netherlands still had not updated its Copyright Act, although the bill was passed by the Second Chamber of Parliament on October 12, 1993, and was expected to be approved by the Senate soon.

cidence of moral rights.[28] Such differences may also have an impact on software protection.

B. *Article 6*

Notwithstanding incidental text differences, which may result from translation problems or particularities of the respective national systems, the national implementations have followed the Directive rather closely. Still, this does not exclude differences in protection under national law. Two important questions raised by article 6 demonstrate this fact. First, can the right to decompile for compatibility purposes be restricted in other ways, for instance by patent or trade secret law? Second, what exactly may be used after permissible decompilation has taken place? May only interface "information" be utilized, or is literal interface "code" also acceptable?

1. Incidence of National Trade Secret Law

According to article 9, paragraph 2 of the Directive, decompilation for compatibility purposes may not be excluded by contract. The German Copyright Act, in Section 69 g, paragraph 2, follows suit and explicitly states that contracting out of decompilation is not permitted. Moritz[29] argues, however, that the decompilation provisions do not affect trade secret law. In this respect he points out that article 9, paragraph 1 explicitly leaves trade secret law intact, and that paragraph 2 forbids only contracting out. Consequently, even if decompilation is not seen as copyright infringement and may not be forbidden by contract, the use of the information might still violate a trade secret. Others tend to disagree,[30] though admitting that interfaces of licensed software can indeed be considered as trade secrets.[31] Even Moritz admits that, in any event, the refusal to use interface information might violate antitrust law, which therefore must be taken into account. Still, it cannot be excluded off-hand that national trade secret law will apply to some extent, if perhaps not for the decompilation as such, then at least as

28. Bodin v. l'Agospap (Paris Dist. Ct. Jan. 20, 1993), *reprinted in* 1993 EXPERTISES DES SYSTÈMES D'INFORMATION 187 (1993) (changes in a computer program by a licensee were not only found to infringe the license, but the programmer's moral rights as well).

29. H. Moritz, *Softwarelizenzverträge. Rechtslage nach der Harmonisierung durch die EG-Richtlinie über den Rechtsschutz von Computerprogrammen*, 9 COMPUTER UND RECHT 341, 350 (1993).

30. Schulte, *supra* note 22, at 656.

31. Thomas C. Vinje, *Softwarelizenzen im Lichte von art. 85 des EWG-Vertrages*, 9 COMPUTER UND RECHT 401, 408 (1993); A. Wiebe, *Reverse Engineering und Geheimnisschutz von Computerprogrammen*, 8 COMPUTER UND RECHT 134, 137 (1992); *see also* J. Taeger, *Softwareschutz durch Geheimnisschutz*, 7 COMPUTER UND RECHT 449, 451 (1991).

regards the exact use which may be made of the information thus obtained.

2. Incidence of National Patent Law

The incidence of patent law may go even further. Article 6 permits decompilation in order to obtain certain information. It does not, however, permit the use of such information if that use is covered by a patent. It could be argued conversely that, to the extent the use of the interface is covered by a patent, even the decompilation will not be allowed. Patent information must, legally speaking, be considered "previously readily available." To the extent that the patent will prevent one from "achieving interoperability," one of the first conditions of article 6 has not been satisfied.

In the EC, patent law is still partly a matter of national law. Most patents today are granted by the European Patent Office ("EPO"), though a patent is not necessarily granted for all EC countries. The applicant has a free choice as to which countries he wishes to designate. More important, the European Patent System is not exclusive. It therefore remains possible to obtain patents through the national patent offices, and there is a wide variety in the national patent laws in the EC as regards software patentability. Admittedly, where software patents are concerned, most of these laws are even more restrictive than the European Patent Convention ("EPC"), which certainly does not provide for unlimited granting of software patents either.[32] Some countries, however, especially The Netherlands, have a very liberal software patent system, the possibilities of which exceed the EPC. Furthermore, some countries provide patents with respect to interfaces, and this could affect decompilation rights.

3. Permissible Use of the Interface as Such

In a classic sensory research experiment, Titchener asked test persons in a dark room to give their first impressions of objects handed to them. He then handed them a coin, and expected to be told it was something round, cold, hard, etc. Instead, the standard immediate reaction turned out to be "a nickel" (or whatever it was).[33] The experiment shows that it may be hard to split data from conclusions, or what one learns from what one is entitled to know. This, of course, applies to interfaces as well. Eventually, decompilation leads to interface information, but first of all it leads to the interface itself. Currently, the Direc-

32. A.P. Meijboom, *Software Protection in "Europe 1992,"* 16 RUTGERS COMPUTER & TECH. L.J. 407, 409 (1990).

33. P. VROON, BEWUSTZIJN, HERSENEN EN GEDRAG 31 (1976).

tive and the national laws only allow use of interface information, yet one may ask whether this will exclude every use of the interface as such. In this author's view, this question must be answered according to the relevant copyright rules, which means that one must first check whether the interface (or a set of interfaces) is original, and if so, whether an exception to copyright applies.

a. Interface Originality

Originality forms the "threshold" for a computer program to qualify as a work, and thereby to be protected by copyright laws. Originality is also a decisive factor in determining the scope of protection since parts or aspects of computer programs will only be protected on the condition that they are original. This holds true at least in France, Germany, and The Netherlands. For example, if someone copies part of a module from an original computer program, it must not only be found that the program was original to constitute infringement, but it must also be demonstrated that the copied part was equally original. Consequently, the required originality level is of direct importance for the protection of interfaces, and thereby for the whole of the compatibility issue, since interfaces may only (though even then not necessarily) enjoy copyright protection if they are original.[34]

Although originality is one of the points which the Directive specifically intended to harmonize, and this harmonization roused strong feelings in several countries, it can be doubted whether the final wording of the Directive is sufficiently unambiguous to lead to a uniform interpretation. Given the reluctance of the German doctrine to accept the lower threshold of the Directive, German courts probably will not consider any and all interfaces original. On the other hand, it must equally be doubted whether an English court, if convinced that the interface was made by the programmer himself, will feel inclined to apply further tests for originality. In this author's view, no German court, nor indeed a Dutch one, would consider original such interfaces as ones merely consisting of the letters "SEGA" or "IBM."[35] Such interfaces therefore could be freely used (provided of course the other conditions have been met) and, one might add, considerably larger interfaces probably could be freely used as well. Of course, it may be a matter of policy whether one dares take the risk, or whether one prefers to follow

34. Dreier, *supra* note 8, at 580, 583; Haberstumpf, *supra* note 12, at 138; Vinje, *supra* note 19, at 196.

35. Vinje, *supra* note 19, at 196 (agreeing that interfaces of such a small size will generally be non-original, yet stating that "the term 'IBM' arguably constitutes expression"). This may be true in some jurisdictions, yet not in countries such as Belgium, France, Germany, or The Netherlands.

the hard but more secure way of clean room re-engineering of the interface code. Vinje[36] certainly is right when suggesting it may be a wise thing to do, but it will perhaps not always be necessary.

Incidentally, it must be noted that some countries may offer protection even to non-original interfaces. This may, for example, be the case in Denmark, where such interfaces could be protected by the so-called "catalogue rule." The catalogue rule is a doctrine known in the Nordic countries[37] that offers a ten-year protection against copying catalogues and other factual information products, including "list-like elements of software."[38]

b. Exceptions to Copyright Which May Apply to Interfaces

Even if the interface is original, copying may still be permitted by an exception to copyright. Article 6 is, of course, such an exception, as it permits the reproduction and decompilation of certain parts of a computer program. It does not, however, extend to the further use of an original interface as such and, in particular, to the insertion thereof into a new computer program. Still, the national copyright laws may contain exceptions that might cover such an insertion in certain situations. The exceptions to copyright, even if they often find a legal basis in the Berne Convention, vary a great deal from one country to another. Perhaps the most important is the "quotation right" as stated in article 10, paragraph 1 BC, and its variations in the different countries.

According to Bauer,[39] article 51 of the German Copyright Act, which deals with the right of quotation, does not directly apply to interfaces, nor indeed does any other existing exception. In his view an extensive interpretation might perhaps be justifiable, but it is unlikely to be accepted by the German Supreme Court, given that court's usual narrow interpretation of such exceptions. Dreier, however, does not consider the application of the quotation right to be excluded, although he believes it must remain restricted to "the extent necessary for the purpose."[40]

The bill to amend the Dutch Copyright Act originally provided that none of the ordinary exceptions to copyright would apply to software. This provision, however, was limited by Parliament to the effect that only the exception which allows the copying of a work for private use shall not apply to software. Although the right of quotation

36. Vinje, *supra* note 19, at 195.
37. Denmark, Finland, Iceland, Norway, and Sweden.
38. Paul Nielsen, *Software Copyright in the Nordic Countries*, 9 COMPUTER LAWYER 1 (Mar. 1992).
39. Bauer, *supra* note 11, at 92.
40. Dreier, *supra* note 6, at 784.

will continue to apply to software, article 15a is worded in such a way that it can hardly apply to the copying of interfaces into other software.[41] In comparison, article L. 122-5 of the French Act seems to be worded somewhat more broadly, making quotation permissible provided that mention is made of author and source, and the use involves "brief quotations which are justified by [*inter alia*] the scientific or informational character of the work into which they are incorporated."[42] Whether this will be sufficient to cover interfaces remains to be seen.

Probably the most extensive text, however, can be found in the Berne Convention itself. According to article 10, paragraph 1, "it shall be permissible to make quotations from a work which has already been made lawfully available to the public, provided that their making is compatible with fair practice, and that their extent does not exceed that justified by the purpose." It can be argued that article 10, paragraph 1 should be considered directly applicable to the software copyright issue. At least that article does not leave the matter to national law, as do many other Berne Convention provisions concerning copyright restrictions. As Ricketson points out, "this is a mandatory requirement of the Convention to which each Union member must give effect in relation to works claiming protection under the Convention."[43]

The most interesting part of article 10 states that quotations shall be allowed, "provided that their making is compatible with fair practice." It could be argued that, as a rule, it may be fair practice to copy an interface into a product which merely is intended to interoperate with the original product (the *Sega v. Accolade* situation), while it will not be fair practice if the copying aims at making a compatible product such as an IBM-compatible BIOS. Of course, the various national laws may contain yet other restrictions which might apply, at least in some situations.

V. Discussion

The debate concerning the Directive had reverse engineering and the compatibility aspects of computer software as special points of focus. This was by no means accidental. Rightholders are extremely apprehensive about the threat which they believe reverse engineering presents. Others feel that the need for software compatibility is as essential. In the debate, many arguments were presented, including argu-

41. One may quote from a protected work "in an announcement, review, discussion or scientific treatise."

42. "Les . . . courtes citations justifiées par le caractère critique, polémique, pédagogique, scientifique ou d'information de l'oeuvre à laquelle elles sont incorporées."

43. Sam Ricketson, The Berne Convention for the Protection of Literary and Artistic Works 477 (1987).

ments about the nature of copyright. At the same time, it is submitted that several aspects did not receive sufficient attention because they were not of primary concern to anyone. In order to reach a balanced view, one must first briefly review reverse engineering, the need for access to interfaces, other arguments for reverse engineering, and the copyright principles that have been invoked.

A. General Remarks About Reverse Engineering

When engaging in reverse engineering, one studies a product in order to find out how it works and what its specifications are. In the case of computer software, reverse engineering is understood as a technique whereby, starting from the program's object code, one tries to disassemble the program towards its source code, and from there further back to its underlying structure and algorithms. Many lawyers seem inclined to consider the reverse engineering of computer software as a more or less suspect procedure which should be admissible only in certain rather exceptional circumstances. Studying what someone else has done may perhaps be a good way of learning, yet by its very nature it carries a suggestion of imitation rather than creative activity. Such a method of study may be appropriate and welcome, provided it remains limited to merely studying ideas. But even the arduous student might be tempted to take more than ideas only, and to copy the whole structure of the program.

Such fears are understandable. Yet, in several respects these fears tend to lead to a distorted view. First of all, reverse engineering can be undertaken for a number of reasons: to learn, to correct errors, to prove infringement of one's own software by a third party, to establish compatibility, and perhaps also to imitate a program. The main reason why software is reverse engineered has nothing to do with imitating a competitor's products, trying to delve into his trade secrets, or even a desire just to educate oneself. Reverse engineering is primarily a technique used for the maintenance of proprietary software.[44] As often as not, complex computer programs are insufficiently documented to enable programmers to maintain them without reverse engineering. Parts of the code, even source code, turn out to be incomprehensible and the programmers who originally developed the program have left the company or if not, they have forgotten what the program was all about. The show must go on, however, and reverse engineering turns out to be the answer. Additionally, even well-developed and documented

44. G. Wets, *Reverse Engineering: State-of-the-Art*, 35 INFORMATIE 102 (1993); *see also* Kindermann, *supra* note 20, at 639; B. Lietz, *Technische Aspekte des Reverse Engineering: Motivation, Hilfsmittel, Vorgehensweise, Nachweisbarkeit*, 7 COMPUTER UND RECHT 564, 565 (1991).

software may become increasingly complex as time passes, changes are implemented, and new functionality is added. At a certain moment it may be necessary to rewrite the program, while sticking to its original structure and concepts. Again, it may be necessary to engage in reverse engineering to accomplish this result.

Given the tremendous demand for software maintenance, much energy has been devoted to attempts to automate the disassembling and reverse engineering process.[45] These attempts have been far from successful, and most software specialists doubt whether the automation of these processes will ever become feasible.[46] According to Lietz,[47] disassembly and decompilation lead to programs which may be read, not yet to programs which can be understood. It seems highly unlikely that anything of this kind will become possible within the next few years, if ever.

Disappointing as this sorry state of automated reverse engineering may be, at least it has the advantage that, as a rule, competitors are not unduly tempted to use the technique for imitating third party software. As Johnson-Laird has very convincingly pointed out, reverse engineering is not only far from easy but also extremely boring.[48] It is probably easier and almost certainly is much more attractive for competitors to develop completely new software, rather than to reverse engineer existing code.[49]

It is not for this author to say whether Johnson-Laird's remarks cover all situations. For instance, reverse engineering of smaller programs or modules might be less difficult and more rewarding. Still, the

45. Incidentally, the fact that reverse engineering will often be undertaken with respect to proprietary software implies that the development and possession of tools that are designed to assist the reverse engineering process cannot as such be considered an unpermissible or even undesirable practice.

46. V. Ilzhöfer, *Reverse-Engineering von Software und Urheberrecht: Eine Betrachtung aus technischer Sicht*, 6 Computer und Recht 578, 579 (1990); Lietz, *supra* note 44, at 566.

47. Lietz, *supra* note 44, at 567.

48. Anthony Johnson-Laird, *Reverse Engineering of Software: Separating Legal Mythology from Actual Technology*, 5 Software L.J. 331, 340 (1992).

49. *Cf.* Decision of the Technical Board of Appeal of the European Patent Office (EPO), April 17, 1991, Official Journal EPO 1993, at 295. In this case, a device which had been disclosed to the public contained a microchip on which a program written in machine language was stored and which realized a control procedure. It was held that that procedure had not become state of the art by the disclosure of the device. The Board stated :

> In theory, it is possible to reconstruct the contents of a program stored on a microchip, for example by using a 'disassembler' program or by so-called reverse engineering. However, these procedures require an expenditure of effort on a scale which can only be reckoned in man-years . . . . [T]he usefulness of the knowledge to be gained by investigating the microchip would therefore have been entirely disproportionate to the economic damage caused by the time spent on such an investigation.

Id. at 308.

mere thought that at some time in the future automated software reverse engineering might become a reality, thus enabling competitors to disassemble programs to find an easy way around existing protection has been enough to scare the wits out of software owners. It is submitted that the fear of such an eventuality has profoundly influenced the reverse engineering debate in the European Parliament. In a way, the reverse engineering provisions in the Directive can probably be characterized as a rare instance of "legislation ahead of its time." It must be doubted whether that time will ever come.

B. *The Need for Interface Information*

It can be argued that programmers who want to make their products interface with other software products must simply ask for the details of the interface. Quite often the rightholder will be only too pleased to give the necessary information, as the availability of applications helps him in gaining market acceptance.

True as this may often be, it certainly does not guarantee that the information will indeed be made available in all situations. To cite one example from The Netherlands: Some years ago, Tulip Computer, the largest PC-compatible manufacturer in the country and one of the largest in Europe, was one of the first to announce a 386, 16 Mhz PC. Towards the end of the development, the company was confronted with the problem that Lotus 1-2-3 would not access on this particular model; instead a screen message accused the user of using a non-authorized copy. Clearly this was a major problem, since a DOS PC on which Lotus 1-2-3 will not run simply cannot be marketed. Tulip suspected the Lotus 1-2-3 copy protection system to be responsible for the problem. According to Tulip, however, for the very reason that its copy protection system was involved, Lotus refused to provide information and merely suggested that the error was due to some mistake on Tulip's side. Eventually, after decompiling the relevant piece of software, Tulip found that the copy protection system indeed caused the problem, as it was based on a carefully timed question-and-response procedure which did not function correctly because the system ran faster than the software anticipated. The problem was cured by an adaptation of Tulip's own microcode, without any change in Lotus software.[50] Although this one example cannot justify the entire decompilation provision, at least it shows that reverse engineering may be undertaken for quite reasonable, necessary, and it is submitted, harmless reasons as far as the rightholder's interests are concerned.

50. Documentation of Tulip Computers for the Working Session of The Netherlands' Association of Informatics and Law, on 30 March 1990, case 1.

C. *Reverse Engineering and Copyright Principles*

Much of the debate concerning the Directive in general, and the reverse engineering and compatibility issues in particular, has focused on copyright principles. This Article has already examined such notions as originality and reproduction. Other aspects of copyright principles concern the proper scope of copyright and the question of whether copyright should offer access to ideas.

1. The Proper Scope of Copyright

Cohler and Pearson have argued that article 6, by allowing decompilation in order to obtain indispensable information for achieving compatibility, is contrary to intellectual property principles.[51] Article 6 "provides access to interface information despite contentions concerning the proper scope of intellectual property protection." As a general statement, this seems to go too far. There are situations where interface information is indispensable and the desire to obtain it is justified, yet the information can only be obtained by way of decompilation.

The point is that a full-fledged copyright system requires some checks and balances. Since the beginning of copyright law in its modern form, with broadly worded reproduction and public performance rights, it has been widely accepted that there must be restrictions to the general rules. For example, certain uses of copyright materials in view of the public interest or other needs must be balanced against the copyright owner's interests. As Ricketson remarks, "Even in those countries where there is the most vigorous commitment to the advancement of author's rights, it is recognised that there is a need for restrictions or limitations upon these rights in particular cases."[52] Consequently, all copyright laws provide such restrictions and most continental European copyright statutes contain many of the restrictions. There is no reason why software copyright should not be equally subject to either the same or specific restrictions.

2. Access to Ideas

Copyright only protects expression, and only protects expression from reproduction and other acts of exploitation. It leaves others free to study the expression and even to copy the ideas, a principle which is openly accepted in article 1 of the Directive. A much debated issue is whether this implies that copyright also gives a right of access to such

51. C.B. Cohler & H.E. Pearson, *Software Interfaces, Intellectual Property, and Competition Policy*, 9 COMPUTER L. AND SEC. REP. 160 (1992).

52. RICKETSON, *supra* note 43, at 489.

ideas, a view which is advocated by several authors.[53] Lehmann[54] takes an especially fundamental line. In his view, once a work has been published, copyright should not be allowed to protect the ideas it contains in whatever way, either directly or indirectly. To allow protection of ideas under copyright law where software is concerned would amount to introducing an element of unfair competition into copyright law that is comparable with patent law, but without the same safeguard measures for the public interest.

Others, such as Vivant and Le Stanc,[55] Ilzhöfer,[56] and Huydecoper[57] take the opposite view, as does this author. First, as Huydecoper rightly points out, the fact that copyright only protects expression, not ideas, is "beside the point." The fact that ideas which have been made publicly accessible will thereby fall into the public domain does not give any guidance as to what should happen to ideas which have not thus become accessible. Second, it must be pointed out that to refuse access (through decompilation) to such ideas that are the basis of a computer program is quite another thing than patent protection. It does not in any way form a bar to the working of one's own creativity. A patent will prevent a person who independently develops the same invention from using it in a competing or even in a non-competing program. The harm which the inventor may suffer by disclosing his invention is, therefore, limited. Copyright does not offer any comparable protection to ideas, whether they are novel or not.

It cannot be said to follow from the rationale of copyright that authors must always give up their ideas. Even if one considers copyright, *inter alia*, as a stimulus for the creation and a reward for the publication of new works,[58] it is not clear why this should imply a forfeiture of the right to keep to oneself — whenever possible — such ideas as are at the bottom of the work. After all, copyright does not give anything in return, as it does not protect those ideas. Incidentally, the absence of protection for ideas may at times form a bar to publication because potential authors prefer to keep their knowledge to them-

53. *See, e.g.*, de Cock Buning, *supra* note 25, at 135.

54. Lehmann, *supra* note 14, at 1060.

55. Vivant & Le Stanc, *supra* note 23, at 523.

56. Ilzhöfer, *supra* note 46, at 581; *see also* Kindermann, *supra* note 20, at 640.

57. J.R.L.A. Huydecoper, *"Reverse engineering" en computerprogramma's*, COMPUTERRECHT 58, 59 (1991).

58. In this regard, it should be remembered that U.S. copyright doctrine is more inclined to stress the utilitarian aspects of copyright than is its continental European counterpart. Especially in the French and German copyright doctrines, there is a strong tendency to first consider copyright as a natural right, with the utilitarian side at best coming second. *See also* Jane C. Ginsburg, *A Tale of Two Copyrights: Literary Property in Revolutionary France and America*, 64 TUL. L. REV. 991 (1990).

selves rather than to lose control of these ideas through publication. Taken together, in this author's view the ban on decompilation is not contrary to copyright law.

VI. The Special Nature of Software Copyright

The debate over reverse engineering and compatibility is representative of a discussion about conflicting interests. The discussion tends to get obscured rather than clarified by the fact that copyright is involved. Some people merely stress the fundamentals of copyright law, while others simply insist on the special needs of software. Such one-sided approaches are not much help in understanding the nature of the conflict.

Originally software needed legal protection and copyright could offer such protection. Software could be copied more easily than any other product. Copyright, as its very name demonstrates, deals with copying. Software needed protection at once, internationally and without red tape. Copyright was available almost worldwide and with little or (in most countries) no formalities.

On another level, however, the situation proved to be less ideal. Software needs protection against outright copying, but it also needs other protection. Software developers do not merely want to protect literal source or object code, they want to keep others from following the program structure, user interfaces, and various other creative aspects of the program. In this respect, the match between software and copyright law turned out to be less than perfect. Essentially, copyright protects expression, not function, while software function is certainly more important than form. Even if the functionality of at least larger computer programs involves selection as well as other forms of personal choice, copyright is hardly able to adequately protect the core of the program, and thereby it does not protect the considerable time and money often invested in development of a software program.

This could have been the end of the software copyright story and the beginning of the search for other ways of protection. Indeed, other protection methods such as patents, trade secrets, and contracts all play an important role, but they cannot at present take the place of copyright. Instead, the program core turned out to be copyright protectable after all, be it indirectly, because of two rather remarkable factors. First, a computer program's function can, and in the case of an object program, will indeed be hidden in a tremendous amount of almost incomprehensible expression. Second, since even the merest use of a computer program will involve its being copied wholly or in part into the computer's memory, the broad interpretation of the copyright notion of "reproduction" provided rightholders with a fairly effective shield against attempts to make this incomprehensible code more understandable.[59]

Together, this is exactly the system of the Directive. In fact, the Directive not only protects software against copying, it also treats software as a black box. Provided the inside is original in the sense that it is the author's personal creation, the black box will enjoy copyright protection under article 1. Every glance into the box is inevitably preceded by copying in the copyright sense, and therefore is forbidden—except, of course, with the copyright owner's authorization—which is the essence of article 4. On the other hand, the legitimate user remains free to use the black box, and from the results of that use, to make educated guesses as to what takes place inside. This rule is articulated in article 5. But no rule can be without exception. In two instances one may have to look into the black box: for correcting errors (article 5, paragraph 1) or for achieving interoperability (article 6).

One might say that this system, and perhaps even software copyright in general, protects the software shell, not its core. Since the system offers considerable protection, one is even tempted to say the system works nicely. But even if it does, it does so more by luck than by judgment. In practice this system does not always work. Although it works differently, the system is based on an application of traditional copyright rules. As a result, use of the system will lead to different results than would occur in the case of "ordinary" protected works. In fact, the whole copyright system and most of its rules were developed with regard to those "ordinary" works and were specifically adapted to such works. The system, therefore, may easily produce unpredictable results when applied to software, if consideration is not given to its special nature. This is especially so if existing concepts, such as "reproduction" or existing restrictions to copyright, continue to be interpreted in the same way they have been in the past.

For two reasons this author is inclined to think this risk may be greater in Europe than in the United States. First, the impression is that the "fair use" concept offers greater flexibility and, therefore, more room for a balanced interpretation than the restrictions in the continental European copyright systems, and perhaps even more than the U.K. notion of "fair dealing." Second, although the Directive has struck a reasonable, or at least acceptable, balance between the interests of software owners on the one hand, and the need for compatibility

59. *See* Vivant & Le Stanc, *supra* note 23, at 523. "C'est, en effet, un point remarquable que l'application du droit d'auteur au logiciel permet, pour la première fois, dans l'histoire des propriétés intellectuelles de conjuguer - de cumuler - propriété et secret." *Id.*

on the other, in other respects it may well form too narrow a system. While it permits decompilation for error correction and compatibility purposes, it does not leave any room for decompilation for other reasons which, *prima facie*, might at times also be justified. These justified reasons include, perhaps to a limited extent and subject to further conditions: educational purposes, maintenance of software which is no longer supported by the rightholder,[60] and the wish to prove through decompilation that certain third party software infringes the rights in one's own software.

VII. Conclusion

The software reverse engineering and compatibility issues form a challenge to traditional copyright. While the EC Directive can be said to have struck a fair balance between the main interests at stake, it may at the same time leave too little room for other interests, as well as for new developments. Implementation of the EC Directive at the national level may lead to unpredictable and perhaps nonuniform results. The U.S. system probably leaves more room for adequate answers to the many new challenges that will certainly manifest themselves in the rapidly changing area of computer software protection.

Additonal References Not Cited

J. Berger, *Italy: Implementation of the Software Directive*, 8 Computer L. & Prac. 177 (1992).

J.M.A. Berkvens & G.O.M. Alkemade, *Software Protection: Life After the Directive*, 12 Eur. Intell. Prop. Rev. 476 (1991).

Ivan Cherpillod, *Protection des logiciels et des bases de donnees. La revision du droit d'auteur en Suisse*, 1993 Expertises des Systèmes d'Information 217 (1993).

M. Colombe & C. Meyer, *Seeking Interoperability: An Industry Response*. 12 Eur. Intell. Prop. Rev. 79 (1990).

M. Colombe & C. Meyer, *Interoperability Still Threatened by EC Software Directive: A Status Report*, 12 Eur. Intell. Prop. Rev. 325 (1990).

W.R. Cornish, *Inter-operable Systems and Copyright*, 11 Eur. Intell. Prop. Rev. 391 (1989).

E.J. Dommering, *Reverse engineering: a Software Puzzle, reprinted in*, Amongst Friends in Computers and Law. A Collection of Es-

60. The question of whether decompilation for this purpose should also be permissible, and how it should be distinguished from decompilation for interoperability purposes has led to debate in the WIPO Commission of Experts with respect to a possible protocol to the Berne Convention. *Cf.* T. Hoeren, *Revidierte Berner Uebereinkunft und Softwareschutz. Zu den Ueberlegungen der WIPO für ein ergänzendes Protokoll zur RBUe*, 7 Computer und Recht 243, 245 (1992).

SAYS IN REMEMBRANCE OF GUY VANDENBERGHE 33 (H.W.K. Kaspersen and A. Oskamp eds., 1990).

A.A. van Eck, *Interfaces in de richtlijn*, 1991 Computerrecht 55 (1991).

Th. de Galard, *Le reverse engineering est-il légal?* 1993 EXPERTISES DES SYSTÈMES D'INFORMATION 215 (1993).

T. Hoeren, *Revidierte Berner Uebereinkunft und Softwareschutz. Zu den Ueberlegungen der WIPO für ein ergänzendes Protokoll zur RBUe*, 8 COMPUTER UND RECHT 243 (1992).

J. Huet, *Le reverse engineering, ou ingênierie inverse*, 44 RECEUIL DALLOZ SIREY 99 (1990).

S.J. Karageorgiou, *Implementation of the European Software Directive in Greece*. 9 COMPUTER L. & PRAC. 114 (1993).

M. Kindermann, *Copyright Protection of Computer Programs in Germany: Nixdorf v. Nixdorf*, 13 EUR. INTELL. PROP. REV. 296 (1991).

W.T. Lake, ET AL., *Seeking Compatibility or Avoiding Development Costs? A Reply on Software Copyright in the EC*. 11 EUR. INTELL. PROP. REV. 431 (1989).

M. Lehmann, *Freie Schnittstellen ("interfaces") und freier Zugang zu den Ideen ("reverse engineering"). Schranken des Urheberrechtsschutzes von Software*, 5 COMPUTER UND RECHT 1057 (1989).

Stéphane Lemarchand, *Le tout numérique. Vers une codification internationale des oevres informatiques*. Expertises 1993, p. 305.

K.H. Pilny, *Schnittstellen in Computerprogrammen. Zum Rechtsschutz in Deutschland, den USA und Japan*, 1990 GEWERBLICHER RECHTSSCHUTZ UND URHEBERRECHT INTERNATIONALER TEIL 431 (1990).

C. Reed, *Softwareverträge und das neue englische Urheberrechtsgesetz* 6 COMPUTER UND RECHT 697 (1990).

C. Reed, *Reverse Engineering Computer Programs without Infringing Copyright*, 13 EUR. INTELL. PROP. REV. 47 (1991).

Pamela Samuelson, *Computer Programs, User Interfaces, And Section 102(b) of the Copyright Act of 1976: a Critique of Lotus v. Paperback*, 6 HIGH TECH. L. J. 209 (1991).

A.D. Schuz, *An overview of the Berne Convention - generally and in relation to computer programs and semiconductor chips*, 9 COMPUTER L. & PRAC. 115 (1993).

R. Tuckett, *Access to Public Standards: Interoperability Revisited*, 14 EUR. INTELL. PROP. REV. 423 (1992).

E. Ullmann, *Urheberrechtlicher und patentrechtlicher Schutz von Computerprogrammen. Aufgaben der Rechtsprechung*, 8 COMPUTER UND RECHT 641 (1992).

G. Vandenberghe, *Copyright Protection of Computer Programs: An unsatisfactory Proposal for a Directive*, 11 EUR. INTELL. PROP. REV. 409 (1989).
A. Wiebe, *"User Interfaces" und Immaterialgüterrecht. Der Schutz von Benutzungsoberflächen in den USA und in der Bundesrepubliek Deutschland*, 1990 GEWERBLICHER RECHTSSCHUTZ UND URHEBERRECHT INTERNATIONALER TEIL 21 (1990).
A. Wiebe, *Reverse Engineering und Geheimnisschutz von Computerprogrammen*, 8 COMPUTER UND RECHT 134 (1992).