INTEROPERABILITY DOWN UNDER: THE AUSTRALIAN COPYRIGHT LAW REVIEW COMMITTEE'S FINAL REPORT

by Jonathan Band and Masanobu Katoh[1]

On April 12, 1995, the Australian Copyright Law Review Committee (CLRC) concluded a nearly eight-year study of software copyright issues. The CLRC's 350 page final report culminated an open process of public hearings, several rounds of comments, technical demonstrations, and draft recommendations.[2] For each of the many issues it considers, the final report carefully discusses all perspectives, and then reaches a conclusion of the necessity of a statutory amendment.

Among the more contentious issues to emerge during the course of the CLRC's deliberations were the protectability of interface specifications and the permissibility of software reverse engineering. These same issues contemporaneously were the subject of debate in the European Union and the United States. The CLRC explained its approach to these issues as follows:

> [I]n the creation and protection of any property rights, an attempt must be made to strike the right balance between adequate protection and the need to provide the community with reasonable access to intellectual property and the benefits which it confers . . . . The striking of the balance is something which must be attempted in the public interest. The task has not been an easy one. [3]

This article discusses how the CLRC approached its difficult task and ultimately succeeded in striking the right balance.

A. The Copyright Law Review Committee

The Copyright Amendment Act of 1984 brought computer programs under the protection of the Australian copyright law. In October 1988, the Attorney General asked the CLRC, an officially convened group of jurists, intellectual property lawyers, and industry

---

[1] Jonathan Band is a partner in the Washington, D.C. office of Morrison & Foerster. Masanobu Katoh is the General Manager of the Washington, D.C. office of Fujitsu Limited. This article is based on the authors' recent book, Interfaces on Trial (Westview Press 1995).

[2] Copyright Law Review Committee, Computer Software Protection (1995) (CLRC Report).

[3] CLRC Report 2.01 at 4.

representatives, to consider whether Australian copyright law "adequately and appropriately protects computer programs . . . ."[4] In February 1989, the CLRC requested comments from interested parties on this question, and released in April 1990 an issues paper based on those comments. The paper recited the arguments for and against a reverse engineering exception. While the paper did not specifically address the protectability of interface specifications, it did discuss protection for program structure. The paper made no recommendations on the issues it raised, but invited further comments.

On July 26 and 27, 1990, the CLRC held a public hearing. Alcatel STC testified that reverse engineering was essential to the computer industry, and that copyright law should not be permitted to impede the practice. The U.S. Software Publishers Association, in contrast, testified that most copyright owners are vehemently opposed to permitting the reverse engineering of their products.

In September 1990, IBM submitted written comments to the CLRC stating that "no special category of 'fair use,' or similar exception, [should] be created which might sanction the incidental copying of computer programs where it is part of the process of decompilation."[5] In October 1990, Fujitsu Australia submitted detailed comments taking an opposing view. Fujitsu Australia argued that if Australian software vendors "cannot develop products that conform to de facto interfacing standards (almost always established outside Australia, most typically in the U.S.), programmers in Australia face extremely limited market opportunities (both in Australia and overseas)."[6] Accordingly, "the rules, formats, languages, protocols and similar information underlying a program, including its interfaces, should not themselves be

---

4 CLRC Issues Paper 1 (April 1990).

5 IBM Submission to the CLRC, Sept. 1990, at 25. Decompilation is a software reverse engineering technique that involves converting the machine-readable object code into a higher level, human readable form.

6 Fujitsu Australia Submission to the CLRC, Oct. 5, 1990, at 11.

copyrightable."[7]

Fujitsu Australia further argued that

> reverse analysis is an essential tool in the development of interoperable products. Unless discrete information can be discerned through machine analysis techniques, the development of compatible products can be frustrated; whether intentionally or otherwise. The absence of an ability to engage in reverse analysis would lead to de facto protection for product-to-product interfaces whenever a company failed to document that information.[8]

For this reason, Fujitsu Australia urged that reverse engineering "used to analyze or understand the uncopyrightable elements of a computer program, should be viewed as completely permissible under Australian law."[9]

On November 22, 1990, IBM conducted a reverse engineering demonstration before the CLRC. Through the demonstration IBM sought to show how decompilation facilitated disguised piracy. IBM also submitted a detailed paper on decompilation. IBM stated unambiguously that none of the reverse engineering techniques short of decompilation is "regarded as objectionable or fundamentally inconsistent with the principles of copyright law . . . ."[10] Nonetheless, "there is no justification for legitimizing" decompilation which "involves a flagrant infraction of the copyright owner's exclusive rights to control reproduction, especially where the intent of such process is to quickly develop a substitute program."[11]

IBM stressed that decompilation would facilitate undetectable piracy and that decompilation is unnecessary because "software suppliers publish ample information about their programs" and because less intrusive means of reverse engineering exist.[12] IBM argued that permitting decompilation would reduce a competitor's costs and the first developer's lead time. This, in turn, would reduce the incentives to create new programs. IBM, the world's largest

---

7 Id. at 3.
8 Id. at 18.
9 Id. at 3.
10 IBM Submission to the CLRC, Nov. 22, 1990, at 6.
11 Id. at 6-7.
12 Id. at 13.

computer vendor, was particularly solicitous of small developers: "An exception could impose particular hardship on small vendors who may have only a single successful innovative product."[13]

Several interoperable developers sought to attend IBM's reverse engineering demonstration, but IBM insisted that the CLRC exclude them. This galvanized the interoperable developers into action; they formed SISA, the Supporters of Interoperable Systems in Australia, which conducted its own reverse engineering demonstration before the CLRC on February 7, 1991.[14] SISA also provided the CLRC with presentations on the users' and the business perspective. The main thrust of SISA's advocacy was that the CLRC should (1) clearly exclude interface specifications from copyright protection and (2) permit access to interface specifications by reverse engineering so that (3) the Australian information technology industry could compete effectively in a global market.

B. Autodesk v. Dyason

While the CLRC considered the submissions of SISA and the proprietary vendors, the Australian High Court on February 12, 1992, handed down its decision in Autodesk v. Dyason.[15] This confused decision underscored the importance of the CLRC's work by demonstrating the need for clarification of the proper application of copyright law to computer programs.

Autodesk produced the computer assisted design program AutoCAD. Autodesk sold AutoCAD with a hardwired lock that had to be physically installed on the PC or terminal where AutoCAD was running. Because only one lock was sold with each AutoCAD program, the lock ensured that only as many copies of AutoCAD as had been purchased were in operation

[13] Id. at 14.

[14] SISA's members included several software companies based in Australia, as well as the Australian subsidiaries of Bull, ICL, Unisys, NCR, Sun Microsystems and Fujitsu.

[15] Autodesk Inc. v. Dyason, 173 CLR 330, F.C. 92/001, High Court of Australia (1992).

at any one time. The AutoCAD program issued a repeating cycle of challenges to the AutoCAD lock. Using a shift register, the lock transmitted a series of responses to the AutoCAD program. The AutoCAD program then used a look-up table to determine whether the responses matched the challenges. If it did not receive a proper response, the AutoCAD program (not the lock) would issue instructions to stop the program.

After reverse engineering the AutoCAD lock, the defendants developed an "Auto Key lock" which properly responded to the challenges issued by AutoCAD. This permitted copies of AutoCAD to be used on PCs without the AutoCAD lock. The Auto Key lock, therefore, allowed the user to circumvent the Autodesk lock and use unauthorized copies of Autodesk.

The Autodesk case presented several issues of first impression for Australian courts. First, the Trial Court had to determine whether the hardwired AutoCAD lock constituted a computer program at all. The AutoCAD lock did not issue instructions; rather, it issued a data stream in response to the AutoCAD program's data stream, which the program then evaluated. Second, the Court had to determine whether the defendant copied any of the plaintiff's protected expression.

The Trial Court found for Autodesk, ruling that the AutoCAD lock was a computer program and that the Auto Key lock infringed the AutoCAD copyright because it performed the same function. The Trial Court decision also implied that the defendant's study of the AutoCAD lock's output using an oscilloscope may be improper under copyright law.

The full Federal Court reversed the Trial Court, finding that the AutoCAD lock was not a computer program, and that similarity in function did not infringe copyright. The High Court then reversed the Federal Court and restored the Trial Court's finding of infringement, but for reasons different from those articulated by the Trial Court. The High Court agreed with the Federal Court that the AutoCAD lock was a piece of hardware and not a computer program, and thus was not covered by the Australian copyright law. The High Court nonetheless found

infringement because the defendant's Auto Key lock, even though a computer program, reproduced the protected expression of the AutoCAD look-up table, and this look-up table represented a "substantial part" of the AutoCAD program. Further, one of the concurring opinions suggested, as did the Trial Court, that the defendant's act of reverse engineering also infringed Autodesk's copyright.

SISA submitted a detailed criticism of the Autodesk decision to the CLRC. SISA stated that "[t]he danger of the High Court's Autodesk decision is that it will be portrayed as holding that all interface information by which two separate products interoperate is necessarily protectable 'expression' in Australia . . . ."[16] SISA first argued that the High Court erred in treating the look-up table as protected expression: "data which serves a purely functional purpose should not be viewed as expressive material." SISA next argued that even if the look-up table contained expressive elements, those elements merged with the "idea" of achieving interoperability with the program:

> [I]n order for Auto Key to work in place of the AutoCAD lock, the responses or return codes to signals sent to [the AutoCAD program] had to be identical. There was a very real functional imperative which limited the possible responses to stimuli sent from [the AutoCAD program] . . . . [I]f a discrete amount of data that is passed between computer programs must be identical in order for the programs to work together, that very real absence of choice on the part of the developer creating a new product intended to work with or substitute for an existing product must be considered in determining whether, as to that bit series, "idea" and "expression" have merged.[17]

SISA further contended that Autodesk should not be interpreted as prohibiting reverse engineering. Finally, SISA argued that Autodesk was an instance of bad facts generating bad law. The High Court understandably had little sympathy for the defendant, who sought to circumvent Autodesk's copy protection device. Dyason's product was not intended to increase

---

[16] SISA's Views on the High Court's Autodesk Decision 9-10 (1992) (SISA Autodesk Comments).

[17] Id. at 16.

consumer choice, but to facilitate unauthorized copying. SISA concluded that "[i]n light of the High Court's Autodesk decision SISA believes more than ever in the importance of the Amendment to Australian Copyright law"[18]excluding interface specifications from protection and explicitly permitting reverse engineering.

C. The CLRC Draft Report

In July 1993, the CLRC issued a 350-page draft report. The draft contained a lengthy, but somewhat confusing, discussion of U.S. case law on the scope of protection for non-literal program elements. The draft recommended against any amendment to the Australian Copyright Act specifically dealing with the scope of protection. Thus, the draft did not support SISA's request for a provision specifically excluding interface specifications from copyright protection.

The draft also contained a lengthy discussion of reverse engineering. It acknowledged that because programs typically are distributed in an object code form that is not understandable by humans, certain exceptions to the copyright owner's exclusive rights are required to ensure that the public has access to the unprotected elements of the program. The draft recommended the adoption of a provision similar to Article 6 of the EU Software Directive, permitting decompilation for purposes of achieving interoperability.[19] Indeed, the draft report

---

[18] Id. at 18.

[19] The Draft Report recommended the following language:

[D]ecompilation of a computer program should be allowed where it is necessary to achieve the interoperability of an independently created computer program with other programs provided:

(a) decompilation is performed by the owner of a lawfully acquired copy of the program or another person having a right to use the copy or on their behalf by a person authorized to do so;

(b) the information necessary to achieve interoperability has not previously been readily available; and

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

The following limitations should apply:

(i) the decompilation should only be used to achieve interoperability; and

improved on the language of the Software Directive by eliminating Article 6's confusing reference to the Berne Convention.[20] The draft also endorsed permitting decompilation for error correction. Finally, the draft recommended that the "fair dealing" provision of the Australian Act govern the permissibility of decompilation "to understand techniques."

Thus, the CLRC proposed decompilation rights somewhat broader than those under the Software Directive. The CLRC envisioned an unambiguous right to decompile for purposes of interoperability and error correction, and a flexible case-by-case, fair use approach to determining the lawfulness of decompilation to understand elements of the target program unrelated to interoperability. The draft report, however, without comment omitted a provision similar to Article 5(3) of the Directive permitting black box reverse engineering.[21]

SISA filed detailed comments on the draft report. SISA stated that it believes that the CLRC's Draft Report strikes a fair balance between the interests of copyright holders and the public at large. SISA also believes that the Draft Report is consistent with and in furtherance of the emerging international consensus of protecting computer programs as literary works, on the one hand, while at the same time avoiding excess protection by creating express exceptions to the copyright holder's exclusive rights.[22]

SISA did, however, recommend several revisions. First, it renewed its request for a specific exclusion for interface specifications. Second, it advocated the adoption of a black box reverse engineering provision similar to Article 5(3) of the Software Directive. Third, it recommended several changes to the language of the Article 6-equivalent, most notably broadening it to cover explicitly decompilation for purposes of achieving interoperability between hardware and software.[23]

(ii) the information obtained should only be given to others when necessary for the interoperability of the independently created program.

[20] See Band & Katoh, Interfaces on Trial, at 254-55.

[21] Black box reverse engineering includes research methods other than decompilation, such as line traces and input-output tests.

[22] Comments on the Draft Report of the Copyright Law Review Committee on Computer Software Protection submitted by the Supporters of Interoperable Systems in Australia 1 (1993).

[23] This had been a contentious, and somewhat unresolved, issue during the legislative battle

The U.S. Computer and Business Equipment Manufacturers Association (CBEMA) also filed comments on the draft report on behalf of proprietary vendors.[24] It offered several reasons why Australian copyright law did not need to permit decompilation for purposes of achieving interoperability. First, the policy objective of fostering interoperability

> is already being satisfactorily advanced by a combination of market forces and liberal cross licensing policies of software developers . . . . [T]here is no evidence at all that a 'crisis' has developed in the industry due to an absence of special rules aimed at fostering interoperability.[25]

Second, CBEMA hauled out the disguised piracy rationale: "the protected expressions obtained through decompilation of computer program, often in an easily disguised form, can be used for a number of illegitimate purposes, which may cause substantial harm to the right holder."[26]

Third, citing the reliance of the U.S. Court of Appeals for the Ninth Circuit on the fair use doctrine to excuse decompilation in Sega v. Accolade,[27]CBEMA argued that fair dealing, the Australian analog to fair use, provided the means for "an Australian court to balance interests in the area . . . ."[28] Further, "there is little evidence to suggest that Australian courts, as their U.S. counterparts, will have any difficulty in reaching fair results in specific factual situations through the application of the doctrine of fair dealing . . . ."[29] The irony of this argument requires emphasis. CBEMA in two amicus briefs to the Ninth Circuit vigorously opposed excusing decompilation under the fair use doctrine. Yet in Australia, CBEMA employed the Ninth Circuit's fair use finding as a justification for not adopting a specific decompilation exception.

Perhaps in response to SISA's renewed request for a specific exclusion for

---

leading up to the adoption of the Software Directive. See Band & Katoh, Interfaces on Trial, at 248.

[24] CBEMA's members included IBM, Apple and DEC. CBEMA recently changed its name to Information Technology Industry Council (ITI).

[25] CBEMA Comments on CLRC Draft Report 6 (Oct. 1993).

[26] Id. at 7.

[27] 977 F.2d 1510 (9th Cir. 1992).

[28] CBEMA Comments at 6.

[29] Id.

interface specifications, CBEMA raised several arguments against the adoption of such an exception. CBEMA first argued that including specific terms of art in statutes "narrows the scope of the law, and, thus reduces its applicability in the full range of factual situations which may arise over time."[30] CBEMA next argued that "the copyrightability of specific elements of a program should be determined by the same rules as the copyrightability of specific elements of any other work."[31] CBEMA finally argued that an express exception for interface specifications for the purpose of promoting standardization and interoperability was unnecessary because over the past decade the software market has become increasingly standardized on its own: "[C]ompatibility (or interoperability) has evolved without specific rules diminishing protection."[32]

The U.S. Government also commented on the CLRC draft report. It approved the decompilation provision because it "appear[s] to be generally consistent with the provisions of Article 6 of the EC Software Directive and appear[s] to be directed to achieving the goal of the creation of interoperable programs while protecting the copyright owner against abuse."[33] The U.S., however, suggested that the decompilation provision explicitly state, as does Article 6 of the Directive, that the "end result of the process must be the creation of a program that is itself original."[34] Moreover, the U.S. expressed concern about permitting decompilation for error correction.

D. The CLRC Final Report

The CLRC issued its final report in April, 1995. The final report made several significant changes favoring interoperability.

---

30 Id. at 3.

31 Id.

32 Id. at 4.

33 United States Government Comments on the Copyright Law Review Committee's Draft Report on Computer Software Protection, United States Government Cable to United States Embassy, Australia (1993).

34 Id.

The final report adopted the distinction made by MIT Professor Randall Davis between computer programs as "text" and computer programs as "behavior."[35] The CLRC concluded that behavior should not receive copyright protection, thereby permitting the development of functionally equivalent programs with different texts.[36]

The final report did not accept SISA's invitation to specifically exclude interface specifications from copyright protection. The CLRC did, however, explicitly endorse the Computer Associates v. Altai decision of the U.S. Court of Appeals for the Second Circuit.[37] The report described the Second Circuit's three step test, noting that the unprotectable elements to be removed in the filtration step included "elements dictated by external factors, such as . . . compatibility requirements with other programs . . . ." The CLRC then stated that

> [i]t regards the test set out by the court in that case as a very practical and useful guide for determining infringement if computer programs and supports the approach it adopted.[38]

The report proceeded to discuss the favorable reception Computer Associates had received in Canada[39]and the U.K.,[40]and to reject the District Court decision in Lotus v. Paperback.[41] (The appeal in Lotus v. Borland was still pending when the CLRC drafted its final report.) The report also discussed "look and feel," concluding that "the need for standardization and the need for efficient user interfaces to be used and developed outweighs the need to grant authors express copyright protection in the 'look and feel' of their programs' behaviours."[42] The final report observed that "[w]hile industrial efficiency may not be a consideration in determining protection of other categories of works, that it is in the case of computer program serves to mark them out

---

35 CLRC Report 9.09 at 102.
36 See id. at 9.38 at 112-13.
37 982 F.2d 693 (2nd Cir. 1992).
38 CLRC Report 9.27 at 109.
39 Delrina Corp. v. Triolet Systems, 9 B.L.R.2d 140 (Ont. Ct. of Justice 1993).
40 John Richardson Computers Ltd. v. Flanders and Chemtec Ltd., 1993 FSR 497.
41 740 F. Supp. 37 (D. Mass. 1990).
42 CLRC Report 9.42 at 114.

on account of their functional nature."[43] These statements, taken together, suggest that the CLRC opposed copyright protection for interface specifications.

With respect to reverse engineering, the CLRC adopted SISA's recommendation of a black box reverse engineering exception similar to Article 5(3) of the EC Software Directive.[44] The CLRC also considered, and rejected, CBEMA's opposition to a decompilation exception. The report concluded that the existing fair dealing provision in Australian copyright law was narrower than the fair use doctrine in U.S. copyright law, and probably would not permit decompilation in a commercial context.[45]

The CLRC also rejected IBM's contention that decompilation facilitated disguised piracy. It stated that it found IBM's reverse engineering demonstration unconvincing:

> At that presentation, only a simple form of decompilation was demonstrated, namely the disassembly of a relatively small program. No evidence of generalised decompilation to high level computer languages was provided.[46]

The CLRC noted that "any new program that is produced by reverse engineering an existing program and which is a copy or adaptation of the latter program is no less an infringement."[47] The CLRC thus suggested that the focus of the copyright analysis should be the finished product brought to market, and not the intermediate development steps. The CLRC acknowledged IBM's argument that many operating system interfaces were published, but responded that many other products were not publicly documented. The CLRC further observed that reverse engineering is time consuming, costly, and rarely leads back to a complete version of original source code. For this reason, it is likely to be performed only by interoperable developers with no practicable alternative.

---

43 CLRC Report 9.39 at 113.
44 CLRC Report 10.95-96 at 175-76.
45 CLRC Report 10.27-31 at 147-49.
46 CLRC Report 10.38 at 153.
47 CLRC Report 10.39 at 153.

The CLRC then considered SISA's specific recommendations for amending the decompilation language proposed in the draft report. SISA had opposed the provision limiting decompilation only to those parts of the program necessary for interoperability because it was impossible to know in advance what parts of the program needed to be decompiled. The CLRC agreed and replaced the problematic language. The CLRC also agreed with SISA's suggestion that decompilation be permitted for purposes of achieving interoperability between hardware and software, as well as between two programs.[48]

Further, the CLRC endorsed SISA's suggestion that contractual restrictions on reverse engineering not be enforceable.

The CLRC reviewed the criticisms of its provision permitting decompilation for purposes of error correction, particularly those submitted by the U.S. government. The CLRC rejected the criticisms and retained the provision. The CLRC also retained its approval of decompilation to uncover ideas unrelated to interoperability pursuant to Australia's fair dealing exception.

Finally, the CLRC considered the status of reverse engineering in other jurisdictions. It correctly concluded that its recommendations were consistent with the Software

---

[48] The final report worded the decompilation exception as follows:

[D]ecompilation of a computer program should be allowed where it is necessary to achieve the interoperability of an independently created computer program or hardware device with other programs or hardware devices provided . . . .

(a) decompilation is performed by the owner of lawfully acquired copy of the program or another person having a right to use the copy or on their behalf by a person authorized to do so; and

(b) the information necessary to achieve interoperablity has not previously been readily available; and

(c) the acts are confined to those necessary to achieve interoperability.

The following limitations should apply:

(i) the decompilation should only be used to achieve interoperability; and

(ii) the information obtained should only be given to others when necessary for the interoperability of the independently created computer program or hardware device.

Directive in the EU and the Sega decision in the U.S., and complied with the provisions of GATT-TRIPs and the Berne Convention.[49]

E. Conclusion

It is now up to the Ministry of Justice to determine whether to submit the CLRC's recommendations to the Australian Parliament. Even if the recommendations are not adopted into law, the report's careful analysis of reverse engineering will prove helpful to courts and legislatures throughout the world when they consider these issues. Because of the parallels between the CLRC's recommendations and the EU Software Directive, the report will be particularly useful in interpreting and applying the Directive's provisions. Specifically, the report resolves three potential ambiguities in Article 6 of the Directive: it eliminates the confusing reference to the Berne Convention; it permits decompilation to achieve interoperability between software and hardware; and it removes the technologically infeasible limitation of decompilation to only those parts of the program necessary for interoperability. The CLRC's thoughtful resolution of these potential ambiguities should be followed in Europe.

[49] The final report states "that in Japan the law, if literally interpreted, would prohibit the reproduction and adaptation of computer programs for the purpose of reverse engineering." CLRC Report 10.100 at 177. A close examination of Japanese copyright law, however, reveals that it in fact permits reverse engineering. See Band & Katoh at 294-97.