CASE NOTE[*]

DATA ACCESS CORPORATION v POWERFLEX SERVICES PTY LTD AND OTHERS

The domestic software development industry is understandably enraged at the decision in *Data Access Corporation v Powerflex Services Pty Limited.*[1] The case held that a computer language is capable of being protected by copyright. In the context of software generally, however, the decision means that developers can own copyright in what are commonly known as 'interface standards'.[2] Ownership of interface standards inhibits the ability of rival developers to develop substitutes. While this is welcomed by the multinational software companies which own most of the existing interface standards, it means that any smaller developer is competing with one arm tied behind its back.

---

* Simon Cant, BCom (Information Systems) LLB (UNSW), Solicitor, Allen Allen & Hemsley.

1 (1996) 33 IPR 194.

2 In the context of computer software, interfaces are those parts of a computer program which interact with other programs or with users. Interfaces can be divided into program interfaces and user interfaces.
A program interface can usually be described as a set of rules for the passing of information between programs. Those rules might include how to send information to a program (that is, what name or memory address to use) and any limits on the quality (that is, value, length etc) of the information which may validly be sent.
The user interface of a program is the behaviour of the program that is apparent to the user. It includes all the rules governing output, such as how graphics or text may be displayed on the computer screen, and all the rules governing input such as how the keyboard or mouse may validly be used.

I. THE POWER OF THE INTERFACE

Many types of products cannot be used in isolation, but must be used in combination with additional products. An increasingly well understood computer software example is the relationship between operating systems such as Windows and application programs such as a word processor. An application program is of no value without the operating system with which it was designed to operate. Similarly, an operating system is of little use without applications that are compatible with it. The substitutability of rival operating systems, and of rival applications, depends upon the rival products having program interfaces which are similar, if not identical, to those of the originals.

The ability to develop substitute products can also depend on the user interface. For example, most modern day word processors, such as Word or WordPerfect, require users to invest substantial amounts of time and energy in order to become proficient. Users' ability to switch to a competing product depends upon the similarity of its user interface to that of the original.

In the United States, a body of case law has developed limiting the extent of copyright protection for interfaces.[3] In Europe there has been legislative reform to achieve this end.[4] However, in Australia the principal authority, the High Court's decision in *Autodesk v Dyason*,[5] suggests that copyright protection does extend to interfaces.

The device which had its interface reproduced in *Autodesk* was a *program lock.* The sole purpose of a program lock is to prevent people from using pirated versions of certain software, in that case computer aided design software.[6] Accordingly, the sole purpose of reproducing the interface of the program lock is to enable, if not promote, software piracy. It is widely believed, therefore, that that case should be limited to its facts.[7]

However, the decision is binding and was followed in *Data Access* with predictably unfortunate results for domestic software developers. *Data Access* also raises other issues in relation to the application of the *Copyright Act* 1968 (Cth) to computer programs. Each of these issues is considered separately below.

3 *Computer Associates v Altai* (1992) 982 F 2d 693, which has been followed in numerous cases including *Sega Enterprises Limited v Accolade Inc* (1992) 977 F2d 1510 (9th Cir) at 1525; *Atari Games Corp v Nintendo of America Inc* (1992) 975 F2d 832 at 839.

4 The *Directive on the Legal Protection of Computer Programs*, Art 6, provides that reproduction and adaptation is permitted in certain circumstances where it is necessary to create or operate interoperable programs, although they may not be used to create a substantially similar program.

5 (1992) 173 CLR 330.

6 The program lock must be plugged into the back of a computer to enable the software to be operated. Thus while unlimited copies of the software may be made, only those who have a program lock will be able to use it.

7 G Greenleaf, "Information Technology and the Law: Autodesk (No2) (continued) ... more functions for 'function'" (1993) 67 *ALJ* 541; P Prescott, "Was AutoCAD Wrongly Decided" [1992] 6 *EIPR* 191; PM Conrick "Autodesk - a response" (1992) 5 *IPLB* 12; K Manwaring, "Autodesk Inc v Dyason" (1992) *Sydney Law Review* 518; D Hunter, "Reverse Engineering Computer Software - Australia Parts Company with the World" (1993) 9 *Computer Law and Practice* 122; K Davey, "Reverse Enginnering of Computer Programs" (1993) 4 *Australian Intellectual Property Journal* 59.

II. FACT

The products at issue in *Data Access* were *application development systems*.[8] The two systems were highly compatible in the sense that applications developed on one could generally also operate and be modified on the other.

The original DataFlex system was created by Data Access Corporation ("Data Access") and first released in 1981. Dr Bennett, co-founder of Powerflex Services Pty Limited ("Powerflex Services"), having used the DataFlex system for some time, set about developing his own application development system based on the DataFlex language. In 1989 Powerflex Services began selling this system, first under the name Powerflex and later under the name PFXplus.

The two systems had the following similarities:

1. The PFXplus language had 192 of the 225 non-graphics-related commands in common with the DataFlex language. Use of the corresponding words in each language caused the same function. However, there was no similarity between the portions of the source code of the two systems which carried out those corresponding functions except that the words, which were in effect labels for those functions, were stored as data in both systems.
2. Both systems had identically named function keys.
3. In relation to three commands known as macro commands, the code which carried out their functions was in an intermediate language in DataFlex known as I-code. In PFXplus the functions were encoded in source code. Expert evidence was given that if the I-code was translated to source code there was substantial similarity between that translation and the source code for the functions in the DataFlex system.
4. Both systems had the same data compression table. This table was used by the program to reduce the space on a disk or other storage device required for storing data. It did this by providing a conversion between the conventional digital representation of a character (or other type of information) and a shorter representation of that character. The shortest representations were reserved for the most commonly occurring characters.
5. Both systems created and were only able to access files with the same specific structure, although the corresponding portions of program code of the two systems which created and accessed files in accordance with that structure were not similar.
6. The two systems had similar but not identical error tables, such that when a certain error occurred a similar message appeared.

Similarities 1, 4 and 5 were essential to enable programs developed using one system to operate on the other. Similarities 2 and 6 were merely for the

8 An application development system is the collective name for a set of programs that enable the writing of applications in some language. Applications are programs developed for specific commercial or other purposes other than the maintenance and operation of computers. Applications are distinguished from operating systems, for example, which have no direct use except in conjunction with applications.

convenience of users in that users accustomed to one system could easily use the other. Similarity 3 was arguably for the convenience of Powerflex in creating PFXplus.

III. ISSUES

The primary allegation was that in creating a system compatible with the DataFlex language, Powerflex Services had infringed copyright in the DataFlex system.

In relation to similarities 1 and 2, Jenkinson J found that the label corresponding to each command and function key amounted to a computer program in and of itself. Further, he rejected the suggestion that the function or idea of each command was inseparable from its expression as a label so as to lose protection. Accordingly, he found that copying each of the labels in respect of 192 commands of the DataFlex language and the 16 function keys amounted to an infringing reproduction.

In the alternative, if it were the case that each label was not a computer program, he held that the copying of the words of the DataFlex language amounted to the making of a substantial adaptation of the DataFlex system.

He also found that there had been substantial adaptation in relation to similarities 3 and 5: similarity 3 because of the resemblance between the PFXplus source code for the macro commands and a source code translation of the DataFlex I-code for those commands; and similarity 5 because of the correspondence between the function of the relevant parts of each system in creating and accessing files of the same structure.

In relation to similarity 4, Jenkinson J rejected the argument that the table was inseparable from the programs idea of function so as to be unprotectable. Accordingly, he held that the copying of the table infringed Data Access's copyright.

Only in relation to similarity 6 did Jenkinson J find that the similarity between the expression contained in the systems was dictated by the idea which it was intended to convey. The expression being thus inseparable from the idea, the relevant portions of the error table were not protected by copyright and, therefore, not infringed by their reproduction.

From the foregoing, it is apparent that the case primarily concerned three issues in relation to computer program copyright:

1. the definition of a computer program
2. substantial adaptations; and
3. merger of an idea with its expression so as to render it unprotectable by copyright.

These issues are examined in turn.

A. The Meaning of Computer Program in the Copyright Act

On the face of it, the most striking finding in the case is that each word of the DataFlex language per se[9] is a computer program, and therefore each is protected as a separate work under the Copyright Act 1968 (Cth). However, Jenkinson J acknowledged that this finding may be incorrect and held in the alternative that even if the term computer program were given its natural meaning, the words of the DataFlex language constituted "a substantial part of each of the computer programs comprising the DataFlex application development system".[10]

Section 10(1) of the *Copyright Act* defines a computer program as "an expression of a set of instructions intended to cause a device having digital information processing capabilities to perform a particular function". There are two aspects to Justice Jenkinson's finding:

(a) each set of instructions associated with a single command constitute "a set of instructions intended to cause a [computer] to perform a particular function"; and

(b) each word (that is, the string of letters) corresponding to a command, was an "expression", albeit at the highest level of abstraction, of a set of instructions.

Accordingly, his Honour held each word *per se* was a computer program within the terms of the *Copyright Act.*

(i) Delineating a Program From a Set of Instructions

The first aspect of the finding touches upon the issue of how a single computer program should be delineated in a larger set of instructions. Few computer users would regard this as an issue. A single computer program is that program which is operated when a file name is typed into a keyboard or when an icon is clicked. However, what appears to be a single program to users is in fact often the combined operation of programs contained in multiple files (not the least important being those files comprising the operating system). Moreover, many largely autonomous programs may be contained in a single file.

Justice Jenkinson's approach of identifying as a computer program the instructions associated with a single word in the DataFlex language is consistent with the definition of "computer program" in the Copyright Act. That definition refers to instructions which relate to "a particular function". Secondly, as the Copyright Law Review Committee noted when the issue of how to delineate a single computer program from a larger set of instructions was raised before it, courts have to deal with the same issue in relation to many other types of works.[11] Thirdly, compilations of computer programs which might generally be understood to be a single computer program are still protected, since the definition of a literary work in the Copyright Act includes "a compilation of computer programs."

9 As opposed to the program code which carried out the functions invoked by each of those words.

10 Note 1 *supra* at 205.

11 Copyright Law Review Committee, *Computer Software Protection Office of Legal Information and Publishing*, 1995 at p 61.

(ii) A Single Word of a Language as the 'Expression of a Set of Instructions'

The second aspect of the finding is more controversial. The words which Jenkinson J held to be computer programs are not instructions but are labels, which allow DataFlex to find the appropriate instructions to carry out a command when it comes across that command in a program written in the DataFlex language. It could therefore be argued that as labels they are not instructions, but data.

In this light, the finding is not consistent with the approach of the High Court in *Autodesk*. There, Dawson J, with whom the majority agreed, held that data (which contained the code for a program lock) "does not of itself constitute a computer program within the meaning of the definition - it does not by itself amount to a set of instructions".[12] The same could be said of the labels or data representing words of the DataFlex language. Therefore, arguably the words *per se* should not be copyright protected.

Justice Jenkinson's primary finding also sits uneasily with *Exxon Corporation v Exxon Insurance Consultants International Limited.*[13] In that case, it was held that the original word "Exxon" was not protected by copyright because it "has no meaning and suggests nothing in itself. To give it substance and meaning, it must be accompanied by other words or used in a particular context or juxtaposition."[14]

Similarly, in relation to the words comprising the DataFlex language, if each word is to be protected as a work in its own right, rather than as a substantial part of the DataFlex system, they should be considered without reference to the DataFlex system. As a result, it could be argued that those words which are original are meaningless and therefore not protected by copyright. Those words which are simply normal English words, on the other hand, are not protected because they are not original.

(iii) The Alternative Finding is More Supportable

The High Court's reasoning in *Autodesk* does, however, support Justice Jenkinson's alternative finding that PFXplus infringed the copyright in DataFlex, because the words were a substantial part of the larger sets of instructions comprising programs in the DataFlex system.[15] The High Court in *Autodesk,*[16] held that while not of itself a computer program, the data in that case did constitute a "substantial part" of the computer program within which it was embedded, and was accordingly protected by copyright, although there was some disagreement on this issue in *Autodesk v Dyason (No2).*[17]

12 Note 5 *supra* at 346.

13 [1981] 2 All ER 495.

14 *Ibid* at 503.

15 Under s 14 of the Copyright Act, acts performed in relation to a substantial part of a work are deemed to be acts in relation to the work as a whole. Whether a part of a work is substantial is always held by courts to be determined by a qualitative rather than a quantitative assessment.

16 Note 5 *supra* at 346, per Dawson J.

17 (1993) 176 CLR 300.
This was a decision on an application for rehearing argument in relation to the first Autodesk case. The application was rejected by the majority. However, various of the judges made comments on the arguments

Justice Jenkinson's alternative finding may also avoid the problem that in isolation the words of the DataFlex language have no meaning. As a substantial part of the programs comprising the DataFlex system, the words are given meaning by their context.

B. The Role of Function in Determining Substantiality and Adaptation

Jenkinson J found three instances in which he regarded Powerflex as having adapted parts of the DataFlex system. In making these findings, he was required to make qualitative assessments of the relevant portions of the two systems and relied heavily upon the function of those portions. While the High Court has indicated that reliance on function is not permissible, this case highlights the difficulty of finding any other relevant quality of program text upon which judgments about substantiality or adaptations may be based.

(i) Translations of Programs into Different Languages are Adaptations

The meaning of "adaptation" as it relates to computer programs is defined in s 10(1) of the *Copyright Act* as "a version of the work (whether or not in the language, code or notation in which the work was originally expressed) not being a reproduction of the work". As Jenkinson J notes, according to the *Explanatory Memorandum,* Copyright Amendment Bill 1984 (Cth), this aspect of the definition of the term "adaptation" is intended to cover:

> translation either way between the various so-called "high-level programming languages" in which the programs may be written by humans (often called "source code") and languages, codes or notations which actually control computer operations (often called machine code or object code). Thus "adaptation" is intended, for example, to cover the compilation of a FORTRAN program to produce machine code which will directly control the operation of a computer. Languages, etc of intermediate level would also be covered.[18]

Jenkinson found that if the I-code expression of the three macros[19] in the DataFlex system were translated into source code, there was a strong "objective similarity" between that source code and the source code for the three macros in the PFXplus system. Having made that finding of fact, it was a natural application

---

put. In particular, in relation to the argument that the table ought not be protected because it did not form part of a set of instructions, Mason CJ and Gaudron and Brennan JJ each made comments.

Mason CJ acknowledged that it was arguable that the table was merely data and not "instructions" for the purposes of the definition of computer program. As to whether it would still be protected by reason of the words "(whether with or without related information)" in the definition, he did not express a view. Both Gaudron and Brennan JJ, however, took the view that commands and data in their entirety constituted instructions for the purposes of the definition of "computer program" in the *Copyright Act.*

18 Explanatory Memorandum to the *Copyright Amendment Act* 1984 (Cth) cited in *Data Access*, note 1 *supra* at 204.

19 When software packages are purchased, users usually only receive object code, which is code that can only be understood by the computer. The source code from which the object code is derived is readable by humans. Usually this source code is kept secret by the vendor of a software package and consequently only the vendor can modify the software.

In order to allow additional functions to be added to software packages, vendors developed the concept of macros. These are programs which function as part of the software but which can be modified by the user. They are usually written in some minimalist language often described as a macro language. The macros in the DataFlex system were written in a language called I-code.

of the concept of adaptation as articulated in the *Explanatory Memorandum*, to find that the expression of the macros in the PFXplus system was a substantial adaptation of the corresponding expression in the DataFlex system.

(ii) Functionally Equivalent Programs Held to be Adaptations of Each Other

Jenkinson J relied on more than the similarity in expression. He also referred to the fact that the expression of the macros in each system "compile to cause the computer to perform the same complex function".[20]

The function of the source code is given even greater emphasis in relation to his Honour's finding in relation to the file structure.[21] Jenkinson J specifically noted that the relevant sections of the source code versions of two systems were not objectively similar. His Honour also noted that there was no evidence of similarity at a machine code level. In relation to this aspect of the systems, the only thing in common between the two systems was their function. Nevertheless, Jenkinson J relied on this similarity in function to find that there had been an adaptation of the instructions which determine file structure.

This form of reasoning was expressly rejected by the High Court in *Autodesk*. In that case, the judge at first instance, Northrop J, found that because the two program locks in question performed identical functions, the substitute was a reproduction of the original. Dawson J held that:

> the significance placed by Northrop J upon the function of the two locks would appear to be in disregard of the traditional dichotomy in the law of copyright between an idea and the expression of an idea.[22]

(iii) Functionally Important Instructions Held to be Substantial

Dawson J also noted that the distinction between idea and expression had been criticised and was often difficult to apply. The difficulty in applying the distinction is indeed manifest later in Justice Dawson's judgment where he holds that the table which had been copied and which represented the program lock's code was a substantial part of the program in which it was embedded, because it was "essential". It is difficult to imagine how such a table might be essential except in its role in the functioning of the program lock. There is a suggestion later in Justice Dawson's judgment that this is precisely the basis upon which he finds the lock to be essential. He notes "it is *essential* that it should be so", meaning that one table be a copy of the other, "for the lock to operate as intended" (emphasis added).[23]

In a similar vein, there is a suggestion that Jenkinson J relied on the identical function of the corresponding commands in the two systems when he made his alternative finding that in so far as the words had been copied there had been a

20 Note 1 *supra* at 201.

21 This is the structure imposed on data files used by the systems. Both DataFlex and PFXplus used data files to store object code which had been compiled for users' programs written in the DataFlex or PFXplus languages. Note that the word 'structure' in this context does not refer in any way to the structure of the programs comprising the DataFlex or PFXplus systems.

22 Note 1 *supra* at 344.

23 *Ibid* at 346.

substantial adaptation of the programs comprising the DataFlex system. He stated it was substantial "because what has been copied informs virtually all of the expression of the instructions".[24]

Once again, it is difficult to imagine in what sense the words of the DataFlex language inform the instructions comprising the DataFlex system, except that they dictate the functions which those instructions must carry out.

(iv) The Difficulty of Determining Substantiality or Adaptation without Having Regard to Function

In *Autodesk v Dyason (No 2)* Mason CJ was alert to the danger of relying upon the function of instructions as a basis for finding that they were *essential* and therefore a substantial part of the program within which they were embedded. He stated that "such an argument ... misconceives the true nature of the inquiry and seeks to reintroduce by another avenue an emphasis upon the copyright work's function".[25]

His Honour suggests instead that in the context of copyright which protects original expression, substantiality should be judged by reference to the originality of the relevant expression. However, even this raises difficulties since originality in copyright requires nothing more than that the expression was not copied. There are not degrees of originality. Therefore, it provides very little basis for making a qualitative determination of whether a portion of expression is substantial.

In fact, if courts are to ignore program function (so as to not inadvertently protect it) it is difficult to imagine what other basis they have to apply various qualitative assessments required by copyright such as substantiality and adaptation. Unlike other texts, program instructions have only one relevant meaning - their function.

C. The Role of Function in the Doctrine of Merger

This raises arguably the most critical issue in relation to the application of copyright to computer software; that is the application of the doctrine of merger. The doctrine of merger provides that where the expression of an idea is inseparable from the idea it is not protected by copyright. This doctrine having only been recently accepted by the High Court in *Autodesk v Dyason,* Australian law now teeters precariously between two different approaches to the idea-expression dichotomy.

(i) The Roots of the Doctrine of Merger

The doctrine of merger as it applies to utilitarian works has its roots in the United States case of *Baker v Selden.*[26] The case concerned copyright in a set of accounting forms consisting of ruled lines and headings. The court stated as the basic principle that:

24 *Ibid* at 378.

25 Note 17 *supra* at 305.

26 (1978) 101 US 99.

> Where the truths of a science or the methods of an art are the common property of the whole world, any author has the right to express the one, or explain and use the other, in his own way.[27]

Applying that principle, the court held:

> [W]hilst no one has a right to print or publish [the plaintiff's] book, or any material part thereof, as a book intended to convey instruction in the art, any person may practise and use the art itself which he has described and illustrated therein. ... The copyright of a book on book-keeping cannot secure the exclusive right to make, sell, and use account-books prepared upon the plan set forth in such book.[28]

The doctrine was approved by the High Court in *Autodesk*. It restated the principle thus: "[when] the expression of an idea is inseparable from its function, it forms part of the idea and is not entitled to the protection of copyright."[29]

(ii) Merger Construed Narrowly in Data Access

Powerflex sought to rely on merger to justify three similarities between the DataFlex and PFXplus systems: the use of identical words for the commands comprising the DataFlex and PFXplus languages; the use of identical data compression tables; and the similar expression of the error messages in each system. The first two of these similarities were essential to ensure system compatibility and consequently were, it argued, inseparable from the system's function.

The issue turned on whether the merger doctrine denied protection to the detailed aspects of a program's function (that is, its "method of operation"), or whether it merely denied protection to some more abstract idea of a program's function such as its general purpose or principle. These two alternative approaches were drawn from two United States cases both concerning the spreadsheet package Lotus 123.

Data Access argued that the narrow view of *Baker v Selden* which was applied in the first Lotus case, *Lotus Development Corporation v Paperback Software International*,[30] should also be applied. In that case the court held that the hierarchical menu of commands was not excluded from protection either by the exclusion for methods of operation, s 102(b),[31] or the doctrine of merger as articulated in *Baker v Selden*. It held that something would only be excluded if:

> the subject matter would be appropriated by permitting the copyrighting of its expression. ... [I]f there are numerous other ways of expressing the non-copyrightable idea, then those elements of expression ... are copyrightable.[32]

Powerflex, arguing that the merger doctrine excluded methods of operation from protection, drew support from the second Lotus case, *Lotus Development Corporation v Borland International Inc.*[33] In that case the court held that a hierarchical menu of commands was a "method of operation" and was therefore

27 *Ibid.*
28 *Ibid* at 104.
29 Note 5 *supra* at 345.
30 (1990) 18 IPR 1 at 25.
31 Copyright Act (US).
32 Note 30 *supra*.
33 (1995) 49 F3d 807.

not protected by copyright relying on both *Baker v Selden* and the express exclusion of "methods of operation" in s 102(b) of the *Copyright Act* (United States). While acknowledging that Australia does not have an equivalent exclusion under its *Copyright Act*, Powerflex argued that case was nevertheless persuasive because it purported to be an application of *Baker v Selden* which was approved by the High Court in *Autodesk v Dyason*. Applying the "methods of operation" test to this case they argued that: "In the hands of those using DataFlex programs to develop applications, the words of the language are tools by which the computer is operated."[34]

Jenkinson J, understandably, felt bound to adopt the narrower view of *Baker v Selden* applied in the first Lotus case, since it was that case which had been cited by the High Court in *Autodesk v Dyason* (the second Lotus case had not been decided at that point). Accordingly, he held that each word of the DataFlex language, rather than being an integral part of that command's function was merely one possible expression of it. Given that for most of the DataFlex commands there were other words which would aptly describe their general purpose, he found that:

> most of the words - ie the concatenations of letters - of the DataFlex language which have been used as words of the PFXplus language in my opinion go beyond the functional elements of the ideas they express ... and [are] therefore copyrightable.[35]

Similarly, in relation to the data compression table, while not disputing that it was necessary to copy the compression table in the PFXplus system in order to achieve compatibility with the DataFlex system, he noted that the merger test is applied before consideration of any question of infringing reproduction.[36] Clearly, his Honour's view that merger only excludes abstract ideas from protection is implicit in this finding.

The only point on which his Honour was prepared to accept that merger doctrine applied was in relation to the similar error messages. Ironically, this was the only similarity which was not necessary to achieve technical compatibility.

(iii) Merger in Australia is Neither Fish nor Flesh nor Good Red Herring

Justice Jenkinson's approach, while consistent with *Autodesk*, is indicative of the fact that Australian law on the idea-expression dichotomy, while having departed from its United Kingdom roots, is now based on an approach which, whilst born in the United States, has since been dismissed by that country's courts as unworkable.

Whilst the High Court did not acknowledge the fact in *Autodesk*, the merger doctrine appears to have been new to Anglo-Australian law. This view is articulated in the judgment of Jacob J in *Ibcos Computers Limited v Barclays Mercantile Highland Finance Limited*,[37] where he suggested that:

> There is, I think, danger in the proposition that, 'If there is only one way of expressing an idea that way is not the subject of copyright.' As Lord Hailsham observed in *LB Plastics v Swish* [1979] RPC 551:

34 Note 5 *supra* at 199.
35 *Ibid* at 201.
36 *Ibid* at 203.
37 *Ibcos Computers Limited v Barclays Mercantile Highland Finance Limited* (1994) 29 IPR 25.

> But of course as the late Professor Joad used to observe it all depends on what you mean by 'ideas'. What the respondents in fact copied from the appellants was no mere general idea.
>
> ... The true position is that where an 'idea' is sufficiently general, then even if an original work embodies it, the mere taking of that idea will not infringe. But if the 'idea' is detailed, then there may be infringement. It is a question of degree. The same applies whether the work is functional or not, and whether visual or literary.[38]

Jacob J went on to note that United States law differed in this respect from English law:

> The Americans (many would say sensibly) never developed copyright so that functional things like exhaust pipes could not be copied. This is partly due to their statute, which is different from our Act [and as noted above different from the Australian Act which like the United Kingdom statute has no exclusion for methods of operation]... Moreover United States case law has, ever since *Baker v Selden*, been extremely careful to keep copyright out of the functional field ... In *Baker v Selden* a design of ledger sheets which had a particular function was refused copyright. I doubt that would have happened here."[39]

The approach of the Australian courts in *Autodesk* and *Data Access* has been largely dismissed in the United States. In the second Lotus case the First Circuit held that: "If specific words are essential to operating something, then they are part of a 'method of operation' and, as such, are unprotectable."[40] That case, being a decision of the First Circuit Court of Appeals, overrules the reasoning in the first Lotus case, which was a decision of a single judge in the United States District Court. The First Circuit decision was appealed to the Supreme Court, but due to an even split on the case, the decision of the First Circuit stands.

More importantly, the first Lotus case is not consistent with *Computer Associates v Altai*,[41] in which it was held that: "aspects of computer software cannot be subject to copyright if they are dictated by the hardware or software with which the software is designed to interact." This case, whilst also not specifically approved by the Supreme Court, has been followed by almost every other circuit of the Federal Court.[42]

VI. ECONOMIC IMPLICATIONS OF THE MERGER DOCTRINE

It is submitted that unless and until a world wide consensus is reached on a more tailored means of determining the level of protection to be granted to software interfaces, Australia does its industry a great disservice by granting more protection to such interfaces than is granted in other countries, particularly the United States. The reasons for this are economic, and therefore should be addressed at a legislative level if the courts fail to follow the current United States approach.

---

38 *Ibid* at 38-9.
39 Note 29 at 292.
40 *Note* 33 *supra* at 816.
41 Note 3 *supra*.
42 *Ibid.*

A. The Role of Legally Created Monopolies

The law has long recognised the danger of monopolies.[43] By denying copyright protection to abstract ideas, the law prevents authors from obtaining monopolies over functional objects unless they meet the more rigorous requirements of patent law such as novelty and inventiveness. In the United States, the courts have gone further, holding that in the context of computer software and certain other functional works, merger should apply to interfaces in order to prevent authors from obtaining monopolies over interface standards. However, the monopolies which may be created by granting copyright in interfaces are qualitatively different from those which would be created by allowing abstract ideas to be copyrighted.

The value of a monopoly over an abstract idea is always to some extent, and often entirely, dependent upon the quality of that idea. It is intrinsic to that idea. Therefore, the value of a monopoly over a drug depends upon the usefulness of that drug (although this may vary over time).

The value of a monopoly over an interface, on the other hand, depends upon the quantity and quality of *other* items which that interface allows the work to operate with. Thus, the value of a monopoly over an operating system interface depends upon the number and quality of applications that will operate with it. The value of a monopoly over a user interface depends upon the number of people trained in that user interface. These values are not necessarily dependent upon the quality of the interfaces and are therefore extrinsic to the interfaces.

B. The Economics of Product Networks

The economics of interfaces are known to economists as network effects. The classic example of network effects is the telephone system. The value of a connection depends upon the number of other people who have connections. While large networks create value for consumers, they also tend to reduce the variety of interface standards available and therefore inhibit the normal function of a market in fostering the emergence of better interfaces.

(i) The Tendency of Product Networks to Expand

On one view, network effects will create externalities[44] where networks are not owned by a monopolist. For example, where one firm owns the interface of an

43 Where there is competition in a market firms will tend to sell products at cost because if they sell above cost consumers will turn to other sellers who are selling at cost. In such markets, firms are called price takers because a single firm cannot influence the price set by the market as a whole. Monopolists, on the other hand, are price setters because consumers have no alternative seller from whom to buy. The only factor limiting prices which monopolists may charge is the value of a product to consumers. As they raise prices, more people will consider the product not worth buying, and less product will be sold. Therefore monopolists will stop raising prices when the additional income they receive from raising prices across the market is outweighed by the drop in income from lost sales. Economists consider monopolies inefficient in that too little is produced and sold at too high a price. The law has long recognised this problem.

44 Most people are probably familiar with negative externalities. An oft cited example of a network externality is pollution. The harm resulting from the pollution produced by any one individual will be spread over the whole community. Therefore, given that each individual will only bear a fraction of harm they will be causing, there is little incentive not to engage in pollution creating activity. Anybody who refrains from

operating system, it should be able to extract from users the value to them of a large network of compatible applications because it can charge above cost prices. However, where no firm owns an operating system interface and there is competition amongst vendors of rival operating systems which use that interface, no vendor will be able to recoup the value to users of encouraging a large network of compatible applications because none will be able to charge above cost prices. Therefore, there will be sub-optimal investment in expansion of the network.

On another view the same size networks may be achievable even with unowned interfaces. Competitive producers of compatible products, being unable to compete on price, will be forced to compete by developing applications for the operating system interface that will provide them with the largest market. Thus there will always be a tendency for large networks to develop.

(ii) Expanding Product Networks Reduces the Variety of Interfaces and Therefore Inhibit the Emergence of Better Interface Standards

The real disadvantage of product networks appears to be that their tendency to converge on a single standard interface inhibits the emergence of optimal interfaces. Success breeds success in product networks so that, for example, once an operating system interface has become more popular, developers of applications will tend more and more to produce their applications for that interface. The same market dynamic will operate, although to a lesser extent, even where interfaces are owned. Assuming owners of competing interfaces have the same ability to discount, the extra value to users of a superior interface could easily be outweighed by the extra value of a wider range of compatible products available for an inferior interface. Therefore, even where interfaces are owned there is a tendency to converge on a single network. The dominant network could well be owned by the most financially able firm rather than the firm with the best product.

(iii) Ownership of Standards May Encourage Alternative Interfaces to be Developed and Therefore Foster Innovation

Competition amongst interfaces may be partly addressed by patents providing a monopoly to new and inventive interfaces. Provided the existing standard is not owned, such a monopolist will be able to extract from consumers the value to them of a new interface. However, in this context, patents are a blunt instrument. Ideally firms should compete on the basis of a variety of consumer preferences.

creating pollution will either be doing it for altruistic reasons or because they believe that it is part of a cooperative effort, and that if they refrain from causing pollution others will also refrain.

Positive externalities, by contrast, exist when by buying a product or engaging in activity, the benefit will be spread over society. The classic example is vaccination. By being vaccinated a person not only protects himself or herself against disease but will benefit society by not communicating diseases to those with whom they come into contact. The benefit to them from being vaccinated will therefore be only a fraction of the benefit which society as a whole will gain from their activity. Accordingly, the incentive to be vaccinated is likely to be sub-optimal in terms of society's welfare as a whole. It is this problem which leads public authorities to regularly warn the public of the long term dangers to society of individuals failing to have themselves and their children vaccinated.

However, interfaces which might be preferred by consumers will not necessarily be new and inventive.

Possibly the ideal position from a social welfare point of view is that interfaces should be protected until they become standards. As noted earlier, it is arguable that product networks will tend to expand, regardless of whether interfaces are owned or not. Therefore, there is little value in allowing interfaces which have become standards to be protected. On the other hand while interfaces are not dominant, ownership promotes competition between rival interfaces. Moreover, where a superior standard is developed and owned, it will have a far greater likelihood of penetrating the market against an unowned standard than against owned standard.

(iv) Interface Protection Should be Governed by Competition Law

The issue then is to find the appropriate level at which an interface should lose its protection. This will involve weighing the advantages of standards innovation against the advantages of large networks. This approach is akin to that taken by some courts in the United States. However, given that the considerations are purely economic, it is a matter which should be governed by competition law, in particular access regimes, rather than copyright law.

Nevertheless, while there is no appropriate access regime in place, there is a strong case that Australia does its industry a great disservice by granting more protection through copyright law than the United States. Australian companies are unlikely to have the capital to be unseating worldwide interface standards. Therefore, most Australian companies must compete on the basis of established standards. While those standards are protected by copyright in Australia, our only real avenue for competition is significantly inhibited.

V. CONCLUSION

Data Access is largely consistent with *Autodesk.* However, it highlights the fact that, following *Autodesk,* Australian copyright law on the idea-expression dichotomy has departed from its United Kingdom roots only to follow a line of reasoning borne in the United States but no longer followed there.

In June this year, Powerflex was granted a stay of execution by the Full Bench of the Federal Court in relation to the orders made by Jenkinson J pending an appeal of his decision to the Full Bench.

Given the economic implications of protecting software interfaces, the most desirable outcome of the appeal in this case would be a finding that interfaces are not protected. If not, the Australian Government should act to alleviate the disadvantages to the domestic software industry of an overly protective copyright regime. In the long run legislatures around the world may develop legislation which is better tailored to address the peculiar economic characteristics of product networks. If so they should be mindful of how such legislation may interact with copyright to give appropriate protection to functional works.