Singapore Journal of Legal Studies
[2005] 234–245

REVERSE ENGINEERING THE NEW REVERSE ENGINEERING PROVISIONS IN THE COPYRIGHT (AMENDMENT) ACT 2004

DANIEL **SENG***

I. INTRODUCTION

Largely pursuant to the copyright obligations contained in the *United States-Singapore Free Trade Agreement* ('*USSFTA*'), the Singapore *Copyright Act*[1] was substantively revised twice last year, firstly through the *Intellectual Property* (*Miscellaneous Amendments*) *Act 2004*[2] in July 2004, and again through the *Copyright* (*Amendment*) *Act 2004*[3] in January 2005. This process of revising the *Copyright Act* would appear to be a continuing one, as witnessed by the request for public feedback regarding the *Copyright* (*Amendment*) *Bill 2005*.[4]

It is however curious that the *Copyright* (*Amendment*) *Act 2004* introduced new provisions relating to the reverse engineering of computer programs even though these formed no part of Singapore's obligations under the *USSFTA*.[5] Nor do the explanatory statements to the *Copyright* (*Amendment*) *Bill 2004*[6] shed much light. Perhaps these new provisions—sections 39A, 39B and 39C—were inserted in an attempt to "restate the law on certain matters".[7] But as this note seeks to demonstrate, these three provisions neither restate the law nor provide any solace for reverse engineers seeking to take advantage of what the legislators claim to be the creation of an "enhanced fair use" regime that is "necessary in light of new technologies and enhanced rights being conferred on the copyright owners."[8]

---

* LL.B., (NUS); B.C.L. (Oxon); Advocate & Solicitor (Supreme Court of Singapore); Associate Professor, Faculty of Law, National University of Singapore. All the views expressed in this article are mine, and I am also responsible for all errors and omissions.

1 Cap. 63, 1999 Rev. Ed. Sing.

2 *Intellectual Property* (*Miscellaneous Amendments*) *Act 2004* (No. 21 of 2004, Sing.), which came into force on 1 Jul 2004.

3 *Copyright (Amendment) Act 2004* (No. 52 of 2004, Sing.), which came into force on 1 Jan 2005.

4 Online: Intellectual Property Office of Singapore <http://www.newiplaws.org.sg/Copyright%20Bill.pdf>

5 The only reference to reverse engineering is with respect to circumvention of technological measures, with a view to "achieving interoperability of an independently created computer program with other programs." See *USSFTA*, Article 16.4.7(e)(i). This is implemented in the *Copyright Act*, s. 261D(1)(d).

6 *Copyright (Amendment) Bill 2004* (Bill No. 48 of 2004, Sing.).

7 *Ibid.* at Explanatory Statement.

8 Intellectual Property Office of Singapore, *Introduction to the Copyright (Amendment) Bill 2004*, at 10-11, online: Intellectual Property Office of Singapore <http://www.newiplaws.org.sg>. Aside from

This however should not be regarded as a denial of the importance of reverse engineering in the software industry. If at all, these new provisions, setting out reverse engineering as "acts not constituting infringements of copyright works",[9] must be carefully examined. In addition, these new reverse engineering defences should also be considered together with the recently revised section 35 of the *Copyright Act*, which had previously been considered by the Court of Appeal in *Creative Technology Ltd v. Aztech Systems Pte. Ltd.*[10] in 1996, and which was the subject of legislative intervention in 1998 pursuant to the *Copyright (Amendment) Act 1998*.[11]

II. Section 39A—Decompilation

Section 39A is derived from section 50B of U.K.'s *Copyright, Designs and Patents Act 1988*[12], which is in turn derived from Article 6 of the *EC Council Directive on the Legal Protection of Computer Programs*[13]. Our section 39A is largely identical to section 50B, albeit with just only one substantive and one minor difference, which will be discussed below.

The object of section 39A is to enable "decompilation" of a computer program, which is legislatively defined as the process of converting a computer program expressed in a low level language into a version expressed in a higher level language, and incidentally copying the program in the course of so converting it.[14] It should be noted that Article 6 of the *Software Directive* does not define "decompilation" as such: it only refers to such activities that will entail "reproduction of the code and translation of its form".[15]

To a software engineer, the definition of decompilation in section 39A is technically inaccurate. Decompilation does not necessarily and always entail producing a "higher level language" version of the program code. Occasionally, information about code behaviour or pseudo-code from code fragments from the decompilation process that is at a higher level of abstraction than the object code may be all that is necessary. It will be difficult to characterise this abstracted information as "code in

---

the increasing use of technological measures to encrypt and protect electronic works, including computer programs, there would appear to be no technological developments that would trigger a review of the law of reverse engineering in Singapore. The matter of reverse engineering in relation to technological measures is separately legislated in the *Copyright Act*, s. 261D(1)(d) (referring to the use of reverse engineering to circumvent technological measures).

9 *Copyright Act*, Part III, Division 3.

10 *Creative Technology Ltd. v. Aztech Systems Pte. Ltd.* [1997] 1 S.L.R. 621, [1996] S.G.C.A. 71 [*Creative Technology* (*C.A.*)].

11 *Copyright (Amendment) Act 1998* (No. 6 of 1998 Sing.).

12 *Copyright, Designs and Patents Act 1988* (U.K.), 1988, c. 48 [*CDPA*], s. 50B. A similar provision is found in the Australian *Copyright Act 1968,* Cth. s. 47D. This, and other similar provisions in the Australian *Copyright Act 1968*, were enacted pursuant to the recommendations of the Australian Copyright Law Review Committee Report. *See generally* Copyright Law Review Committee, *Computer Software Protection* (Australian Attorney-General's Department, 1995).

13 EC, *Council Directive 1991/250/EEC of 14 May 1991 on the Legal Protection of Computer Programs*, [1999] O.J. L. 122/42, Article 6 (decompilation), at 42-46 [*Software Directive*].

14 *Copyright Act*, s. 39A(6); *CDPA*, s. 50B(1).

15 *Software Directive*, *supra* note 13 at Article 6(1).

a higher level language", because there is no commonly-known programming language in this form. It would have been more accurate to describe such an activity as "disassembly".[16]

In addition, section 39A may be too restrictive because decompilation is not permitted of computer programs that are *not* expressed in low level languages, which is typically understood as object or binary code.[17] This is the case with modern programming platforms that use pseudo-code.[18] By this definition of decompilation, no legally permissible decompilation is possible with pseudo-code unless a purposive interpretation is given to the expression "low level language". In comparison, the language used in Article 6 of the *Software Directive*—"translation of [the] form [of the code]"—is less technology specific and thus much more appropriate.

Furthermore, section 39A does not actually permit "decompilation" *per se*. It only allows decompilation of the source computer program "if … it is necessary [to do so] to create an independent [new] computer program which can [interoperate] with the [decompiled computer program] or with [a third computer program]." This is identified in section 39A as the "permitted objective",[19] although it is more apt to refer to it as the *only* permitted objective for decompilation.

This is unfortunate, because as an engineering activity, decompilation is often the means to an undefined end in reverse engineering. Engineers do not necessarily decompile programs to achieve interoperability. Decompilation assists in a study of a computer program, an activity that is often supplemented with observing and testing the program. As Jacob J. puts it in *Mars U.K. Ltd. v. Teknowledge Ltd.*,[20] when describing reverse engineering as an activity:

> A reverse-engineer of any sort (whether one who intends just to copy or one who intends to learn how to make improvements) must start by examining the article, and if necessary, taking it apart to find out how it is made and works. This is true whether the article is mechanical or electrical. In the case of computer programs or chips with stored information the process may not involve physical de-construction: examination by electronic testing may do. But it comes to the same thing.[21]

So the erroneous assumption is made, that the engineer could only decompile the program to achieve interoperability, when the engineer often decompiles a program,

---

16 It is noteworthy that the Australian *Copyright Act 1968*, s. 47D, assiduously avoids the language of "decompilation" and "disassembly", preferring instead to describe such an activity as "reproduction or adaptation [of the original computer program] for the purpose of obtaining information necessary [for interoperability purposes]."

17 See Webopedia Computer Dictionary, *What is low-level language?*, online: Webopedia Computer Dictionary <http://www.webopedia.com/TERM/L/low_level_language.html> and Webopedia Computer Dictionary, What is machine language?, online: Webopedia Computer Dictionary <http://www.webopedia.com/TERM/M/machine_language.html>.

18 See Webopedia Computer Dictionary, *What is pseudocode? at* http://www.webopedia.com/TERM/p/pseudocode.html. A good example is Java programming source code, which is converted into pseudo-code (bytecode), which is in turn interpreted by the Java Virtual Machine to produce the operative object code on different platforms. See *e.g.* Tim Lindholm & Frank Yellin, *The Java Virtual Machine Specification*, 2d. ed., c. 7, online: Sun Microsystems <http://java.sun.com/docs/books/vmspec/2nd-edition/html/Compiling.doc.html>.

19 *CDPA*, s. 50B(2)(a).

20 [2000] F.S.R. 138 (H.C.)

21 *Ibid.* at para. 29.

in addition to using black box testing[22] and other reverse engineering techniques, to establish what the different parts of the program do in the first place.

A. *Decompilation Conditions*

In addition to the requirement for decompilation to be only for the purposes of interoperability, section 39A prescribes five additional conditions for lawful decompilation, one of which is that the person decompiling the software must be a "lawful user".[23] This is quite readily satisfied, since most reverse engineers will be conducting reverse engineering on legitimate copies of computer programs licensed to them personally or their employers. It is clear from section 39A that a "lawful user" need not be a licensee: his right to use the computer program may accrue outside the licence[24] from *e.g.* an implied *Betts v. Wilmott* right,[25] from authorization granted by a licensed user, or from some other right in the law of copyright such as section 39.[26]

But it may be difficult for a reverse engineer to comply with the other four conditions set out in section 39A(2). These are: firstly, that the lawful user *does not* have readily available to him the information necessary to achieve interoperability, secondly, that the lawful user *must* confine the decompilation "to such acts as are necessary to achieve [interoperability]", thirdly, that the lawful user *must not* supply the information obtained by decompilation "to whom it is not necessary to supply the information in order to achieve [interoperability]" and fourthly, that the lawful user *does not* use the decompiled information "to create a computer program which is substantially similar in its expression to the computer program decompiled; or to do any act restricted by copyright."[27]

In relation to the first condition, where the engineer has "readily available to him the information necessary to achieve [interoperability]" with a computer program, it is presumed in law that there is no necessity to decompile that computer program. This presumes too much.

For instance, operating systems often use a mixture of publicly disclosed Application Program Interfaces ('APIs') and private or internal APIs. Section 39A creates a practical hurdle for the engineer. He will not be permitted to decompile those portions of the operating system whose APIs have been publicly disclosed; he may supposedly decompile only those parts of the operating system for which there are no public APIs. Thus an engineer will have to try to obtain documentation on the APIs for a computer program that he wishes to study. If such documentation is not in the public domain, but is only available upon request from the vendor for the program, would such documentation be considered "readily available" so as to debar the engineer from decompiling the program in question? But in most instances, it

---

22 The term "black box testing" is explained below, in the discussion on s. 39B.

23 *Copyright Act*, s. 39A(1), (5).

24 *Copyright Act*, s. 39A(5).

25 (1871) 6 Ch. App. 239 (C.A.).

26 For instance, a user may have to decompile a computer program protected by technological measures to understand (and possibly even disable or remove its technological measures) in order to make a backup copy of his computer program pursuant to s. 39. (While there is no express exception to this effect in the newly enacted s. 261D, Part XIIA of the *Copyright Act* is expressly made subject to the defences in ss. 39 and 39A.)

27 *Copyright Act*, s. 39A(2).

will not be in the commercial interest of the engineer's employer to put his competitor on notice that he wishes to decompile a competitor's program. The competitor may also decline.[28] Additionally, concerns have been raised as to whether such information will be "readily available" if the software vendor is prepared to make such information available to the engineer, but only at a price or on licensing terms that the engineer considers unreasonable or exorbitant.[29] This condition thus puts a practical check on legitimate decompilation activities, even though a software vendor may not prohibit decompilation in the software licences, as these are void and unenforceable.[30]

One suggestion has been to give this condition a practical interpretation. Thus, information must be said to be readily available if the engineer can find out the required information by "making obvious and straightforward enquiries".[31] This interpretation has its own problems. For instance, notwithstanding any available documentation, decompilation may still be necessary, because of incomplete, poor or even erroneous documentation! In a case like this, an engineer who conducts decompilation risks having his interpretation of the documentation called into question *ex post facto*, by experts and counter-experts, on the issue of whether the ambiguous or unclear documentation has made it necessary to decompile the software in question, because the documentation does actually disclose the relevant information and makes it 'readily available'.

Likewise, the second condition requiring the engineer to confine his decompilation to such acts as are necessary to achieve interoperability will create practical difficulties.[32] It is often not possible, given the unavailability of source code, to always limit the decompilation to those "parts of the original program which are necessary to achieve interoperability", as the language of Article 6 of the *Software Directive* requires.[33] As a reverse engineering activity, an investigating engineer may inadvertently decompile some computer programs, or parts of a program, that are not necessary for interoperability purposes. In fact, the engineer will only be able to isolate those parts of the program that are *not* relevant to his investigations *after* the decompilation and analysis of the information extracted.

The third condition requires the engineer to ensure that the decompiled interoperability information is not supplied "to whom it is not necessary to supply the

---

28 See *e.g. Sony Computer Entertainment, Inc. v. Connectix Corp.* 203 F.3d 596, 601 (9th Cir. 2000) [*Connectix*].

29 S. Ricketson, *1 The Law of Intellectual Property: Copyright, Designs & Confidential Information* (New South Wales: Lawbook Co., 2002), at para. 11.185.

30 *Copyright Act*, s. 39A(3).

31 Hugh Laddie *et al.*, *The Modern Law of Copyright and Designs*, 3rd. ed. (London: Butterworths, 2000), at 1632-1633 para. 34.49.

32 The authors of *The Modern Law of Copyright and Designs*, *ibid.* at 1633 para. 34.49 call this condition "technical nonsense".

33 Thankfully, this artificial restriction and limitation on reverse engineering as prescribed in *the Software Directive*, art. 6(1)(c), is not replicated in U.K.'s *CDPA*, s. 50B, nor in the Singapore *Copyright Act*, s. 39A. Or as the authors of *Copinger and Skone James on Copyright* put it, it may be difficult to limit the decompilation to those parts of the software necessary to achieve interoperability "unless one has established what the different parts of the program do." See Kevin M. Garnett & Gillian Davies, *Copinger and Skone James on Copyright: First Supplement to the Fourteenth Edition* (London: Sweet & Maxwell, 2002) at 342. And often, one of the most efficient and unequivocal ways to do so is by decompiling all the different parts of the program.

information in order to achieve the permitted objective". This imposes on the engineer, who may be working with a team, an obligation to manage any disclosure of such information to his colleagues (who may have no responsibilities for the interoperability), his manager or supervisor (who need not actually know the details of his interoperability work) or even his director or Chief Executive Officer. This condition would appear to require the engineer to treat the decompiled interoperability information as if it were a trade secret, but subject him in some regards to even more onerous obligations, since he would be forbidden by this condition to communicate such information to his fellow employees or even his superiors.

The last condition would appear to be both redundant and over-inclusive. What if the decompilation engineer develops a computer program (new program) that infringes the copyright in the decompiled program, because it is "substantially similar in its expression to the computer program decompiled"? A copyright work can be both an original work as well as an infringing work of copyright.[34] The fact of decompilation and its subsequent use to create an original computer program must be seen as two separate activities, since not all decompilation will give rise to the creation of a new program. Even if it does, the fact that separate copyright may vest in the new program justifies the fact that the act of decompilation is *ex facie* non-infringing. There is no reason to make an act of legitimate decompilation illegitimate by the *ex post* use of the decompilation information to create a new program, especially where the copyright owner in the decompiled program has separate redress against the developer of the new program for infringement.[35] The result is that the last condition adds nothing, and helps neither the developer nor the copyright owner of the decompiled program.[36]

B. *Section 35—Reverse Engineering as Fair Dealing?*

Under U.K.'s *CDPA*, decompilation can never constitute fair dealing,[37] unlike section 39A of our *Copyright Act*, which expressly preserves section 35,[38] and thus the more generous U.S. position where reverse engineering may be fair use under the U.S. *Copyright Act*.[39] The U.S. cases have been unequivocal in sanctioning decompilation as a species of permissible "fair use"[40] when tested against the four factors of purpose and character of use, nature of work, amount and substantiality of portion of work

---

[34] *MacMillan Publishers Ltd. v. Thomas Reed Publications Ltd.* [1993] F.S.R. 455 (H.C.); *ZYX Music GmbH v. Chris King* [1995] F.S.R. 566 (H.C.); *Virtual Map (Singapore) Pte. Ltd. v. Suncool International Pte. Ltd.* [2005] 2 S.L.R. 157, [2005] SGHC 19.

[35] *Cf. Campbell v. Acuff-Rose Music, Inc.* 510 U.S. 569 (holding that a transformative use of the original subject matter supports a finding of fair use).

[36] It should be noted that in the Australian *Copyright Act 1968*, the same condition is stated as a *positive condition*, in that "to the extent that the new program reproduces or adapts the original program, it does so only to the extent necessary to enable the new program to connect to and be used together with, or otherwise to interoperate with, the original program or the other program." *See* Australian *Copyright Act 1968*, s. 47D(1)(d).

[37] *CDPA*, s. 29(4).

[38] *Copyright Act*, s. 39A(4). *See* also *Copyright Act*, s. 39B(3).

[39] Our s. 35 was derived from the U.S. *Copyright Act*, s. 107 (17 U.S.C. 107), even though the rest of the provisions in our *Copyright Act* were adapted from the Australian *Copyright Act 1968*. *See* Daniel Seng, "Reviewing the Defence of Fair Dealing for Research or Private Study" [1996] Sing. J.L.S. 136.

[40] *See Lewis Galoob v. Nintendo* 964 F.2d 965 (9th Cir. 1992), *Sega Enterprises v. Accolade* 977 F.2d 1510 (9th Cir. 1992) [*Sega Enterprises*] and *Connectix, supra* note 28.

used and effect of use upon the potential market for the work. Likewise, in *Aztech Systems Pte. Ltd. v. Creative Technology Ltd.*, the Singapore High Court found for the plaintiff, who had admittedly decompiled the defendant's software, on the basis that the decompilation constituted fair dealing under section 35, when tested against the same four factors.[41]

At first sight, the most recent amendments to the *Copyright Act* seem to have altered this understanding. The *Copyright (Amendment) Act 2004* has inserted a new fifth factor in section 35(2)(e), which requires consideration of "the possibility of obtaining the work or adaptation within a reasonable time at an ordinary commercial price".[42]

In the context of reverse engineering, this would appear to now require the engineer to find out if information regarding that part of the program that is being decompiled (or subsequently incorporated in a new program) may be obtained from the right holder within a reasonable time and at an ordinary commercial price. This would mirror the condition in section 39A, namely, that the decompilation information be not readily available. If so, this interpretation will seriously hinder the reliance on section 35 as an alternative to section 39A.

It has been explained above why this is an impractical condition for reverse engineers. It is thus submitted that in the context of reverse engineering, this additional fifth factor should not be construed adversely against the engineer. This is so since to adopt the same interpretation will render otiose the express preservation of section 35 as a defence to decompilation. Additionally, this fifth factor is but one of the non-exhaustive factors[43] to be considered in section 35, and it should not be given undue weight.

III. Section 39B—Observing, Studying and Testing of Computer Programs

Section 39B is derived from section 50BA of U.K.'s *CDPA*.[44] It was a late addition to the *CDPA*, and it post-dated section 50B by more than 10 years. It is an implementation of Article 5(3) of the *Software Directive*.[45] As the recital explains:

> [A] person having a right to use a computer program should not be prevented from performing acts necessary to observe, study or test the functioning of the program, provided that these acts do not infringe the copyright in the program;[46]

---

41 *Aztech Systems Pte. Ltd. v. Creative Technology Ltd.* [1996] 1 S.L.R. 683, [1995] S.G.H.C. 294 [*Creative Technology (H.C.)*]; subsequently reversed by the Court of Appeal in *Creative Technology (C.A.)*, *supra* note 10, but on the basis that the then s. 35(5) prohibited all forms of "commercial" research and private study, and that included commercial reverse engineering. S. 35(5) was subsequently abolished by Parliament pursuant to the *Copyright (Amendment) Act 1998*, thereby restoring the reasoning of the High Court in *Creative Technology (H.C.)*.

42 This appears to have been derived from the Australian *Copyright Act 1968*, s. 40(2)(c). The present s. 40 and the five factors for assessing fair dealing were derived from the recommendations of the Australian Copyright Law Committee on Reprographic Reproduction, Report (also known as the Franki Committee) (1976) at 29.

43 S. 35(2) describes the five factors as inclusive and thus, non-exhaustive, factors. See also *Creative Technology (H.C.) supra* note 41 at 702.

44 A similar provision is found in the Australian *Copyright Act 1968*, s. 47B.

45 Pursuant to the U.K. *Copyright and Related Rights Regulations 2003* (S.I. 2003 No. 2498), reg. 15.

46 *Software Directive*, Recital 19.

Reverse engineers will call this 'black box' testing of computer programs. Black box testing, which does not involve any decompilation or disassembly of the object code of the program, has been sanctioned by our courts as a legitimate manner of conducting reverse engineering.[47] In fact, the view of the U.K. government, until the 2003 legislative amendments, was that any legitimate user of a computer program had a right to observe, study or test a computer program as part of fair use.[48] Section 50BA was inserted only in 2003 because the fair use defence was changed in U.K. to exclude commercial research.[49] Thus, section 39B is unlikely to create any significant substantive change to the law, since our fair use defence has not been changed substantively in the same way,[50] although it helpfully states that any term or condition that prohibits or restricts any acts of observing, studying or testing of computer programs is void in law.[51]

There has been a suggestion that U.K.'s *CDPA*, section 50BA, may be used to sanction the creation of intermediate copies of software made in the course of decompilation, where the acts of decompilation extended beyond those parts of the program necessary to achieve interoperability.[52] It is respectfully observed that this interpretation of the U.K. equivalent of section 39B will erode section 39A because all decompilation activities involve the observing, studying and testing of computer programs. Section 39B sanctions only the observing, studying and testing of computer programs, "while performing any of the acts of loading, displaying, running, transmitting or storing the computer program which he is entitled to do" (black box testing). It does not further sanction the conversion or adaptation of the program from one form to another (decompilation) as part of the process of 'observing, studying and testing'. Thus section 39B may sanction the creation of intermediate copies of the software made as part of black box testing, while section 39A may sanction the creation of intermediate copies as an incidental part of the decompilation process,[53] but only if the decompilation did not extend beyond those parts of the program necessary to achieve interoperability in the first place.

IV. Section 39C—Other Acts Permitted to Lawful Users

Section 39C is derived from section 50C of U.K.'s *CDPA*.[54] It sanctions any copying or adaptation of a computer program, where such copying or adaptation is necessary

---

47 See *Creative Technology (H.C.)*, *supra* note 41; subsequently reversed by the Court of Appeal in *Creative Technology (C.A.)*, *supra* note 10, but on a different point of law.

48 British Copyright Society, *Copyright and the Information Society: Response to Consultation Paper on the Implementation of EC Directive 2001/29/EC*, online: British Copyright Society <http://www.britishcopyright.org/pdfs/policy/2002_016.pdf> at 4. See also Kevin Garnett, Gillian Davies, Gwilym Harbottle, *Copinger and Skone James on Copyright*, 15th ed. (London: Sweet & Maxwell, 2005) at 490.

49 *Ibid*.

50 See *e.g*. *CDPA*, s. 29(4A).

51 *Copyright Act*, s. 39B(2).

52 *Copinger and Skone James on Copyright: First Supplement to the Fourteenth Edition*, at 342.

53 *Copyright Act*, s. 39A(6)(b).

54 A similar provision is found in the Australian *Copyright Act 1968*, s. 47E. See also s. 47F (correcting a security flaw in a computer program).

for its lawful use. The Preamble to the *Software Directive* explains this provision as follows:

> [T]he exclusive rights of the author to prevent the unauthorized reproduction of his work have to be subject to a limited exception in the case of a computer program to allow the reproduction technically necessary for the use of that program by the lawful acquirer;
>
> [T]his means that the acts of loading and running necessary for the use of a copy of a program which has been lawfully acquired, and the act of correction of its errors, *may not be prohibited by contract*; whereas, *in the absence of specific contractual provisions*, including when a copy of the program has been sold, any other act necessary for the use of the copy of a program may be performed in accordance with its intended purpose by a lawful acquirer of that copy;[55]

The object of section 39C is to ensure that the owner may not deny the lawful user of a computer program those rights that are necessary to enable the user to use the program. But while section 39C is unhelpful, in that it merely describes these rights as any "copying or adapting [of a computer program] necessary for [its] lawful use", and for 'correcting errors', the relevant provisions of the *Software Directive* describe these rights as including:

> [(i)] the permanent or temporary reproduction of a computer program by any means and in any form, in part or in whole …
>
> [(ii)] [the] loading, displaying, running, transmission or storage of the computer program [which] necessitate such reproduction …
>
> [(iii)] the translation, adaptation, arrangement and any other alteration of a computer program and the reproduction of the results thereof …
>
> [that] are necessary for the use of the computer program by the lawful acquirer in accordance with its intended purpose, including for error correction.[56]

But these rights are, in the language of the *Software Directive* and section 50C, subject to "specific contractual provisions"[57] or "not prohibited under any term or condition of an agreement regulating the circumstances in which his use is lawful".[58] Although our section 39C is silent in this regard, it is submitted that the same interpretation will be reached because the expression "lawful user" as defined in section 39A(5) includes "a licence to do any act restricted by copyright in the computer program".

Given the myriad ways in which computer programs may be deployed or used, it would be inapposite to create a blanket right for any user to engage in *any activity* of *copying or adapting* a computer program. Any such activity must be circumscribed as being *necessary* for his *lawful use*.[59] But once that particular use of the software

---

[55] *Software Directive*, Recitals 17 and 18 [emphasis added]. These recitals are implemented as Article 5(1) of the Software Directive.

[56] *Software Directive*, Article 5(1) (read with Article 4(a) and (b)).

[57] *Software Directive*, Article 5(1).

[58] *CDPA*, s. 50C(1)(b).

[59] For instance, the vendor of the program may require that the user acquire a special licence to copy his program onto a network, where it may be used by more than one user. Or the vendor may require that a

is licenced, all copying and adaptation necessary to effect that use is sanctioned by section 39C, which thus operates as in effect a statutory implied licence of use. This would include the acts of storing, loading and running copies or adaptations of the computer program in the random access memory, hard disk or cache memory of computer systems or networks, pursuant to its lawful use.

So interpreted, the same object can be found in section 39(3) of the *Copyright Act*, which authorises the making by the owner of a copy of a computer program of "another copy of that computer program … provided that such a new copy or adaptation is created as an essential step in the utilisation of the computer program … in conjunction with a machine and that it is used in no other manner." It remains to be seen if a more generous interpretation will be given to section 39C, especially since the two provisions seem to overlap in their statutory object.

However, the real innovation in section 39C lies in the creation of a statutory right of repair ("correcting errors in the computer program"). Considering that a similar (but qualified) implied right of repair in patent law exists,[60] the statutory right of repair of programs in copyright is a highly valuable right. But its precise ambit is unclear. Does a right of correcting errors encompass making improvements to the program?[61] For instance, is one correcting errors or improving a computer program if the program, which accepts inputs A to C, but not input D, is revised to accept input D? Is one restoring the program to good and proper condition,[62] or introducing a new function by doing so? Is the intent of the software developer as expressed in the software design relevant in determining what was the "good and proper condition"?[63] And if so, are the developers' intents always captured in the software specifications?[64] Unlike the law of patents, where the patent product claims demarcate where repair stops and shades into making of the product,[65] there is no similar requirement[66] to publish or even document the

---

user, who adapts a one-user program for use by multiple users as part of an Application Service Provider business, to acquire a special rental licence for this purpose. Or the vendor may licence software to be used by the user on a per-CPU or per-system basis.

60 *United Wire Ltd. v. Screen Repair Services (Scotland) Ltd.* [2000] 4 All E.R. 353 (H.L.) [*United Wire*] (the right of repair as "a residual right, forming part of the right to do whatever does not amount to making the product [in the law of patents]"), qualifying the right as an implied licence to repair in *Solar Thomson Engineering Co. Ltd. v. Barton* [1977] R.P.C. 537 (C.A.).

61 For instance, Freedom 3 of the Free Software movement defines this freedom as the freedom to improve the program. See Joshua Gay, ed., *Free Software—Free Society: Selected Essays of Richard M Stallman* (Boston, MA: Free Software Foundation, 2002), c.3 at 41. See also Ricketson, *supra* note 29 at para. 11.190, who noted that the equivalent provision in the Australian *Copyright Act 1968*, s. 47E, was declared to be applicable to enable licensees of programs to correct them to ensure year 2000 compliance, but however doubted if s. 47E could be used for this purpose.

62 The Oxford English Dictionary defines repair as "To restore (a composite thing, structure, etc.) to good condition by renewal or replacement of decayed or damaged parts, or by refixing what has given way; to mend" or "To renew, renovate (some thing or part); to restore to a fresh or sound condition by making up in some way for previous loss, waste, decay, or exhaustion".

63 See *e.g*. *Australian Copyright Act 1968*, s. 47E(1)(b)(i).

64 See *e.g*. *Australian Copyright Act 1968*, s. 47E(1)(b)(ii).

65 See *United Wire*, *supra* note 60.

66 It is a legal requirement to support every application for a patent with a specification containing a description of the invention and any claims as regards the invention. See *Patents Act* (Cap. 221, 2002 Rev. Ed. Sing.), s. 25(3)(b).

functional specifications of software protected by copyright.[67] So even if correcting errors can be conceptually distinguished from making functional improvements to software, there may still be serious evidential impediments in applying this distinction.

V. Conclusion

Computer programs have always been odd siblings to the traditional forms of non-industrial works of aesthetics. But in the law of copyright, they are equated with literary works. However, unlike a traditional literary work where its publication entails making the work and its contents accessible to the public, the 'publication' of a piece of computer software can be achieved by making available its object code without its source code. The result is that to a mere user of a computer program, the computer program is a "black box". There is no avenue for the user, skilled or otherwise, to learn, study, examine, change, adapt, revise, correct and build on the software that has been released into the market, unlike other types of works. The ability of the software vendor to publish software without disclosing its source code has created a serious schism in one of the most fundamental cardinals in copyright—where users of works can build on existing works to create new works of authorship. It is this exact shortcoming which has lead to the recent growth and popularity of the open-source movement, in its emphasis on the 'freedom' of the user of any software—the freedom to study how the program works, to adapt it to one's needs, and the freedom to improve the program and release one's improvements to the public, to benefit the whole community.[68] And for this freedom to be meaningful, the free software movement demands that there be access to the source code of the program.[69]

While copyright law has not made it a condition for software vendors to release their source codes, the law should be cognizant of this "information" imbalance between the software vendor and the user of computer software. The legislators are to be applauded for righting this imbalance by introducing new users' rights for observing, studying, testing, using and correcting errors in software. But the law has come up short in the area of disassembly. Disassembly is very often the only way to gain access to the ideas and functional elements embodied in a computer program.[70] But as provided in the recent amendments to our *Copyright Act*, the right of "decompilation" is really too narrow, as it is strictured with too many pre-conditions. In addition, many of these pre-conditions are unclear or ambiguous, and it is to be seriously doubted if a reverse engineer can fully comply with them in good faith. Where decompilation is not with a view to interoperability, or where the pre-conditions cannot be satisfied, reliance must be had to section 35 of the *Copyright*

---

67 There can be very difficult technical issues to resolve in this regard as well. For instance, if software X can read only version 1 files, but not version 2 files, is one correcting software X to make it read version 2 files, or is it an 'improvement'? Does it depend on whether software X was by design not made to read version 2 files? If so, are the details of software design always found in the software specifications, if available?

68 Stallman, *supra* note 61 at 41.

69 *Ibid.*

70 *Sega Enterprises, supra* note 40 at 1528.

Act, as a form of fair dealing. Thus, it must be doubted if the new reverse engineering defences will have much of an impact in introducing certainty and predictability in the law of copyright, let alone in creating an 'enhanced fair use' regime for reverse engineers. In fact, the opposite is true—that a patent lack of clarity and utility in the new reverse engineering exceptions may discourage reverse engineering, and instead encourage software research and development firms to take their activities to countries which have laws that are more conducive to their work.[71]

[71] Laddie, *supra* note 31, at 1640 para. 34.57, commenting on the equivalent provisions in U.K.'s *CDPA*.