How copyright and patent laws protect computer software

IP column published in *The Lawyers Weekly* May 27, 1994

Edward Hore
Hazzard & Hore
141 Adelaide Street West, Suite 1002
Toronto, ON M5H 3L5
(416) 868-1340
edhore@ hazzardandhore.com

The purpose of this article is to try and explain, briefly, the present state of the law as it relates to software. I am going to try to explain it (a) adequately, and (b) without boring you.

The following is the gist of it.

The first thing to understand is that our intellectual property laws were conceived long before the invention of computers or software. Software protection is a kind of square peg hammered into a round hole, or, to be more precise, into two round holes: copyright and patents. (Contracts and confidential information may sometimes be relevant, but the principles that apply do not need any real modification to deal with software, so I will skip them).

There is also a lack of useful caselaw in Canada, so you generally have to resort to US cases.

Let’s start with copyright.

Copyright

In 1988, the Copyright Act was amended to define “literary works” in the Act to include computer programs. (This was not a big deal, however, because a number of Canadian decisions had already said that they were literary works anyway.)

There can be copyright in either an operating system or an application program. (What the hell are those?, you ask. Look it up in *Computers For Dummies*!). It seems there can also be copyright in the user interface i.e. what you see on the screen.

If someone is copying outright, then there is obviously infringment. Assuming your client is the author or has an assignment from the author, and that the software is original, then he or she has all the usual remedies: damages, an injunction and so on.

So far, so good. But thing get complicated when software is similar, but not actually identical.

It is fundamental to copyright that copyright protects the *expression* of an idea, but not the *idea* itself. In the pre-software world, this was a reasonably simple concept to apply in real life; it is reasonably straightforward that the words in a book are expression but the theory that the book is about is an idea, and can be used by anyone. But drawing the distinction between an idea and an expression when you are dealing with software is difficult. Computer programs are generally series of commands which achieve a result. The expression of an idea in programming language arguably *is* the idea.

The US cases take two different approaches, one now somewhat out of fashion.

The out-of-fashion approach was taken in a 1986 case called *Whelan* (797 F.2d 1222), which involved computer software for dental offices. Basically, the court said that everything in the program not absolutely necessary to the central, single idea of the program as a whole was *expression,* and was protected.   In other words, what was protected by copyright was anything that could possibly be done in any other way to achieve the same result.

A 1992 US case called *Altai* (23 U.S.P.Q. 2d 1241) has poured scorn on this, on the grounds that there is no single "idea" in a computer program; the program is made up off lots of little bits and pieces or "abstractions" all of which should be looked at individually. Abstractions that are necessary to the function or are in the public domain should be "filtered"out by the court, leaving only those portions, if any, which are original and therefore subject to copyright. The net effect of all this is that it is probably somewhat more difficult to assert a successful claim.

What this all boils down to is that the courts have to some extent moved away from the "look and feel" concept in *Whelan* ("if it looks kind of the same, and feels kind of the same, then there must be infringement"). Instead, the courts have adopted an approach that recognises that many computer programms are made by putting together odds and ends that are really in the public domain, or are ideas, and are therefore not original expression protectable by copyright.

There is a Canadian case called *Delrina* (1993) 47 C.P.R..(3d) 1, in which the defendant's monitoring program was found not to infringe the plaintiff's copyright, even though the programs had many similarities. The court broke down the program into its component parts, and concluded that the similarities arose mostly because the defendant's programming "bag of tricks" consisted of steps which were in the public domain.

Now that copyright as it relates to software is crystal clear in your mind, let's move on to patents.

Patents

You get copyright just by creating an original work. To get patent protection, on the other hand, you have to actually do something. You have to apply to the government for a patent. If you get one, you get twenty years of protection from the date of your application for whatever is claimed in the patent. This means you can sue the pants off anyone who infringes your patent, *if* (and this may be a big if) your patent is valid.

The problem is that it is not easy to get a patent for software. In fact it's impossible, in theory, because the Canadian Patent Office said in a policy released last summer that "computer programs *per se* are not patentable."

On the other hand, you can get patent protection for a "computing apparatus programmed in a novel manner, where the patentable advance is in the apparatus itself". In other words, if your software is part of some gizmo that is patentable, then you can, in effect, get a patent for it. The trick is to draft the patent application in traditional patent language so that the software sounds like a patentable article or process, and not like software. This is no easy trick.

Not long ago, software producers rarely thought about or tried to get patent protection. But this seems to be changing, particularly in the US, where there are now over 16,000 software related issued patents.