

# Software Copyright and Software Patent

BY EUGENE DERÉNYI, STIKEMAN ELLIOTT LLP

As software patents have become increasingly common, the dichotomous legal protection afforded for software is the subject of lively debate. When does copyright protect software and when does patent law apply? Are the two forms of protection mutually exclusive? The current state of the law regarding legal protection of software in Canada and the United States is discussed below.

## COPYRIGHT VERSUS PATENT

There is a temptation to disregard the pursuit of patent protection at first blush, because the patent process is viewed as relatively slow and expensive compared with the often rapid pace of software version releases. Why apply for a patent on software when copyright arises automatically upon the creation of the work?

Even the terms of protection appear to favour the choice of copyright over patents. The term of protection for copyrights continues for 50 years beyond the death of the author, while patents expire 20 years from the date of filing.

The two alternatives, however, are not mutually exclusive and there are many distinctions between patents and copyrights that should be kept in mind when assessing protection options.

Copyright prevents the unauthorized copying of the *expression* of a work (such as copying the source code), while patents protect the *functionality* of elements in a work. The scope of patent protection, therefore, is much broader than the protection afforded by copyright, which does not protect a work from independent creation of a similar work or identical work. What patent protection lacks in longevity, it makes up for in scope of protection.

Indeed, the Ontario Court of Appeal decision in *Delrina v. Triolet Systems Inc.*<sup>1</sup> (“Delrina”) is a case in point. In *Delrina*, a programmer wrote software code for one employer, Delrina Corp., that assessed the efficiency of operation of Hewlett-Packard HP 3000 computers. A year later, that same programmer began designing a competing product with a different employer with similar functionality, keystroke commands, and display screens. The trial judge accepted that the programmer’s objective was to take away customers from his first employer and that his purpose was to design a product with identical functionality, but the judge ruled that these facts were insufficient to show copyright infringement. The trial judge noted that copyright does not provide the monopoly conferred by patents, and therefore, the

<sup>1</sup> (2002) 58 O.R. (3d) 339 (Ont. C.A.) [*Delrina*].

claim did not amount to copyright infringement. All of the alleged similarities were dictated by functional considerations or otherwise not protectable by copyright.

## SOFTWARE PATENTS IN CANADA:

A key issue is the degree to which software can be patented in Canada. In the United States, as will be discussed later, patent protection is available for a wide range of software. Patent authorities in Canada, however, have been more cautious in extending patent protection to pure software, as opposed to software coupled with computer hardware.

Software-related inventions have been recognized as patentable in Canada since the decision in *Schlumberger Canada Ltd. v. Commissioner of Patents* in 1981.<sup>2</sup> The decision was important because of the explicit holding that there is nothing in the *Patent Act* which excludes inventions involving computers [and was, until recently, the leading case on the issue?]. Since *Schlumberger*, the Canadian Patent Appeal Board has allowed the issuing of a number of system software patent applications.

The Board has also considered the patentability of software-related inventions in two decisions involving patent applications by Motorola (*Re Motorola Inc. Patent Application No. 2,085,228*<sup>3</sup> and *Re Motorola Inc. Patent Application No. 2,047,731*<sup>4</sup>).

In the first case, the applicant had discovered an algorithm for use in evaluating exponential functions, had converted this algorithm into a series of method steps and, finally, had developed a device to carry out this series of steps. The second case involved a similar invention except the algorithm discovered was a convergence algorithm for use with selected convergence rates which gives improved efficiency on mathematical computations of  $j^{\text{th}}$  roots.

The Board stated that it is not possible to obtain a patent containing claims to an algorithm *per se* or to a method which does nothing more than set out the steps needed to solve the algorithm. The applicant made it clear in argument before the Board that it was not seeking to prevent others from using the software it had developed, only the devices incorporating the software. The devices included read-only memory (ROM) coupled to the software.

The Board indicated that it was the presence of the ROM hardware which tipped the balance in favour of patentability. The software plus a hardware component yielded a patentable device. This on the surface is in keeping with *Schlumberger*, where it was held that software neither adds to nor takes away from the patentability of a computer related invention. Having said this, the Board at the same time appears to have been impressed by the applicant's discovery of the algorithm because the ROM on its own is not new and inventive. The result is that coupling new software to hardware, even a nominal hardware component such as ROM, can result in a patentable device. Although the software *per se* is not patentable, patenting the device yields useful protection because conventional computers use ROM to run software and any use of the software, even if independently created, on a computer could create an infringing device within the meaning of the claims.

Based on the Patent Appeal Board decisions, the patentability of software in Canada was affirmed. The key to such patenting was to recognize and highlight the functional nature of hardware, software and, in appropriate cases, data.

In 2005, the Canadian Intellectual Property Office (CIPO) revised the Utility and Subject Matter Chapter and the Computer-Implemented Inventions Chapter in its Manual of Patent

---

<sup>2</sup> (1981) 56 C.P.R. (2d) 204.

<sup>3</sup> (1998) 80 C.P.R. (3d) 71.

<sup>4</sup> (1998) 80 C.P.R. (3d) 76.

Office Practice (MOPOP). These amendments, although not legally binding, provide guidance on how patent applications will be processed.

The new chapters explain that a software invention or “computer implemented invention” is patentable when it passes the usual requirements for patentability (ie utility, non-obviousness, and novelty) and produces an essentially economic result in relation to trade, industry or commerce.

More specifically, for software to be patentable, the MOPOP explains that it must be “an act or series of acts performed by some physical agent upon some physical object and producing in such object some change either of character or condition” and “it must produce an essentially economic result in relation to trade, industry or commerce”.

Three possible types of claims are listed for computer implemented inventions: art or process (method) claims, machine (apparatus and system) claims, and manufacture (products or computer media, embodying code or data structures) claims.<sup>5</sup>

Method claims will define the series of operations which take place in the computer when the computer program is run. The claims must describe the steps that are carried out by, or on, the inventive combination of hardware and/or software. In regards to “machine” claims, CIPO considers that a computer which has been configured with a novel computer program is a different machine from the same computer when programmed in another way.

Manufacture claims describe computer readable memory as it is used to store software statements and instructions. These instructions must direct a computer system to function in a particular manner.

In addition to enumerating these three categories of valid claims, the MOPOP also provides examples of software inventions that exhibit the principles delineated in the amendments.

Although not legally binding, the practical effect of the revised chapters of the MOPOP, in combination with the Patent Appeal Board decisions, is to affirm that patent in addition to copyright should be considered when protection is sought for software.

## **SOFTWARE PATENTS IN THE UNITED STATES:**

The patenting of software is a more liberal process in the United States. This may come as no surprise considering the United States Supreme Court once stated that patentable subject matter includes “anything under the sun that is made by man”.<sup>6</sup> The history leading to the legal condition today has, however, been conflicted.

The first case on the issue of software patents considered by the United States Supreme Court is *Gottschalk v. Benson* in 1972.<sup>7</sup> In considering the patentability of a computer program that converted binary coded decimal into a pure binary number, the Court ruled that the program was not patentable subject matter for the reason that granting a patent would in practical effect grant “a patent on the algorithm itself.”<sup>8</sup>

---

<sup>5</sup> CIPO however released a Practice Notice on Aug. 14, 2007, “Office Practice Regarding Signals”, overturning the patentability of signals: “electromagnetic and acoustic signals are forms of energy and do not contain matter even though the signal may be transmitted through a physical medium. As a result, claims to electromagnetic and acoustic signals do not constitute statutory subject matter within the meaning of the definition of invention in section 2 of the Patent Act.”  
Online: <[http://strategis.ic.gc.ca/sc\\_mrksv/cipo/patents/notice\\_aug14\\_07-e.html](http://strategis.ic.gc.ca/sc_mrksv/cipo/patents/notice_aug14_07-e.html)>.

<sup>6</sup> *Diamond v. Chakrabarty*, 447 U.S. at pp. 303, 309 (1980).

<sup>7</sup> 93 S. Ct. 253 (1972).

<sup>8</sup> *Id.* at p. 257.

Later, in *Diamond v. Diehr*,<sup>9</sup> the Supreme Court again considered the issue but came to the opposite conclusion. The Court held that a process for moulding uncured synthetic rubber involving a programmed computer was patentable subject matter.

The issue of mathematical algorithms was not forgotten, however, and the “Freeman-Walter-Abele” test was developed in three decisions from 1978-1982 in this regard.<sup>10</sup> The two-pronged test asked first whether a claim recited a mathematical algorithm,<sup>11</sup> and second, if it did, whether the algorithm applied to “physical elements or process steps.”<sup>12</sup> Thus, otherwise unpatentable software could be saved if the algorithms directed, in some manner, physical elements or processes.

In 1998, the treatment of software patents by the courts was changed again. Although there was an absence of any physical transformation, the Federal Circuit in *State Street Bank Trust Co. v. Signature Financial Group, Inc.*<sup>13</sup> concluded that a computer program for administering mutual funds was patentable. The Court held that it was a practical application of a mathematical algorithm and thus patentable, since it was more than simply an abstract idea alone. In doing so, the decision overruled the “mental steps” doctrine and “business method” exception, which had previously prevented software patents for being too abstract and effectively mathematical algorithms.

In the recent Federal Circuit decision *In re Stephen W. Comiskey*<sup>14</sup>, however, the Court appears to have resurrected the exception at least in regards to business methods that only involve mental processes. The claim in question involved methods and systems for performing mandatory arbitration resolution. The Court stated that such a claim will be patentable only if it involves another category of patentable subject matter such as a computer.

Shortly after *State Street*, *AT&T Corp. v. Excel Communications, Inc.* was decided.<sup>15</sup> One of the issues in the case pertained to the patentability of a computer program as a process claim. The issue had not been considered in *State Street*, since that case involved a machine claim. The Court held that regardless of the form of the claim, machine or process, the same analysis applied. Here, as there was a “useful, concrete, tangible result”, the program was considered patentable. The approach to software patents in *State Street* and *AT&T* stands: a software related claim will be patentable where it is an application of an abstract idea or implements a useful, concrete, tangible result.

For further information, please contact your Stikeman Elliott representative or the author, Eugene Derényi at [ederenyi@stikeman.com](mailto:ederenyi@stikeman.com)

**Eugene Derényi** heads the Patent Section of Stikeman Elliott's Intellectual Property Group and is co-head of the Patent Litigation section. He wishes to thank Martin Lapner, Student-at-Law for his assistance in preparing this paper.

This publication provides general commentary only and is not intended as legal advice.

© Stikeman Elliott LLP

---

<sup>9</sup> 450 U.S. 175, 177 (1981).

<sup>10</sup> *In re Walter*, 618 F.2d 758 (C.C.P.A. 1980); see also *infra* notes 10, 11.

<sup>11</sup> *In re Freeman*, 573 F.2d at pp. 1237, 1245 (C.C.P.A. 1978).

<sup>12</sup> *In re Abele*, 684 F.2d at pp. 902, 906 (C.C.P.A. 1982).

<sup>13</sup> 149 F.3d at p. 1375 (1998) [*State Street*].

<sup>14</sup> No. 2006-1286 (Fed. Cir. 2007).

<sup>15</sup> 172 F.3d 1352 (1999).