Copyright and Application Programming Interfaces in Canada

September 30, 2015

By Victor Krichker, Catherine Lovrics, Paul Horbal, and Andrew Ngo

The legal battle between Google and Oracle concerning copyright in the Java programming language framework will not be going to the United States Supreme Court, for now. The US Supreme Court recently refused to hear Google's appeal of a Federal Circuit decision, which found that copyright can exist in parts of the Java Application Programming Interface (API), and confirmed that Google copied portions of Oracle's Java APIs when creating its Android operating system. The case will now make its way back to the lower courts, for consideration of whether Google can avoid infringement by relying on the "fair use" defence under U.S. copyright law.

How might Oracle and Google have fared in Canadian courts? What legal protections can the creator of an API expect in Canada? Only a few Canadian cases have explored the scope of copyright protection for, and the appropriate approach to infringement of computer programs. It is unclear what balance would be struck in Canada between protecting APIs, on the one hand, and leaving ideas and elements from the public domain free for all to draw upon to create APIs in the future.

Are APIs likely to be protected by copyright in Canada?

If Canadian courts had to consider the copyrightability of the Java APIs at issue in the U.S. litigation between Oracle and Google, our courts would probably grant protection to the APIs. The APIs are likely to be considered "computer programs", created using skill and judgment, and therefore original works.

The Canadian *Copyright Act* explicitly extends protection to "computer programs", a subset of literary works, which are defined as "a set of instructions or statements, expressed, fixed, embodied or stored in any manner, that is to be used directly or indirectly in a computer in order to bring about a specific result". While Canadian courts have not dealt with the threshold issue, it would appear that most APIs would meet the definition of "computer program", being a set of instructions or statements which are to be used when one computer program interfaces with (or "speaks with") another.

An interesting question is whether or not a Canadian court would refuse to grant copyright protection in certain APIs on the basis that the monopoly would be contrary to the fundamental distinction between ideas and expression. For example, in *Delrina Corp. v. Triolet Systems Inc.*, the Ontario Court of Appeal found that, if there is only one or a very limited number of ways to achieve a particular result in a computer program, to hold that such way or ways are protected by copyright could grant a monopoly on the idea or function itself. In that case, the Court accepted that the "merger" doctrine prevalent in US law is a natural corollary of the idea/expression distinction fundamental to Canadian law (although it did not formally adopt the doctrine). The "merger" doctrine holds that the expression of an idea loses copyright eligibility if it is inextricably linked with its underlying idea.

On the other hand, in 2004, in *CCH Canadian Ltd. v. Law Society of Upper Canada*, the Supreme Court stated that a work is protected by copyright if it is original, meaning it originates from the author's exercise of skill and judgment in the expression of an idea, requiring "intellectual effort". Creativity is not required to make a computer program original, but the program must be more than a mere copy of another work, or a mechanical exercise. It would appear that after CCH, the subsistence of copyright depends only on whether or not the work is original, and the expression involved the exercise of skill and judgment.

Oracle v. Google also raised another question: whether the elements of a copyrighted work can *become* unprotectable by

becoming commonplace – a concept analogous to the “genericization” of trademarks. The U.S. Court did not accept this argument. It is likely that Canadian courts would agree, since generic elements, ideas, and elements from the public domain may be the subject of original expression that results from the exercise of an author’s skill and judgment.

What approach might a Canadian court take to determine if copyright in an API had been infringed?

Canadian copyright law has evolved considerably over the past decade, and today it is unclear what approach a Canadian court would take to copyright infringement involving an API. Canadian courts may apply an abstraction-filtration-comparison type test, similar to that prevalent in the US. Alternately, the Court may adopt a holistic and qualitative approach.

It appears probable that, regardless of the approach, a Canadian court would find that exact (or literal) copying of headings and function names amounts to infringement on the basis they are a substantial part of an API. To quote the U.S. Federal Circuit, the author is not “selecting among preordained names and phrases to create its packages" (*Oracle v. Google*). Headings and function names are the very parts of an API that programmers need and learn, and so that portion of the API would likely be considered a substantial part. What is less clear is whether or not copying the overall architecture of an API (or its structure, sequence, and organization) would amount to infringement. The latter appears more likely to depend on the approach taken to infringement.

In *Delrina*, the Ontario Court of Appeal discussed the “abstraction-filtration-comparison” method used in the US to determine the protectable expression in the course of determining infringement. The Court agreed that such a general “weeding-out” approach to remove from copyright protection portions that cannot be protected by copyright was “not wrong” (but did not formally adopt the doctrine). The facts in *Delrina* involved a computer program that the defendant's company had hired the plaintiff's ex-employee to write. This individual had previously written a program for the plaintiff that performed essentially the same functions. The Court found that, to the extent there were similarities between the two programs, such similarities were not eligible for copyright protection because they were necessitated by the functions the program was required to perform.

However, a more recent Supreme Court decision, *Cinar Corporation v. Robinson*, affirmed that generally Canadian courts should adopt a holistic and qualitative approach to assessing infringement. The Court found that many works do not lend themselves to a “reductive” analysis, and rejected the “abstraction-filtration-comparison” method in relation to the development of a children’s show that took inspiration from Daniel Defoe’s *Robinson Crusoe*. The Court found that a piecemeal approach was inappropriate, and non-protectable elements should not be excluded at the outset of the analysis. Elements that were commonplace, generic and in the public domain were not “weeded out” before determining whether or not a substantial part had been copied. The Court found that all relevant aspects – patent, latent, perceptible, intelligible – should be considered. The Court also affirmed that infringement includes colourable imitation, and covers both literal and non-literal copying. Consequently, the alteration of copied features or their integration into a work that is notably different from the original work does not necessarily preclude a claim that a substantial part of a work has been copied. The Court shifted the focus to considering whether the *“cumulative effect”* of the features copied from the work amount to a substantial part of the author's skill and judgment expressed in his work as a whole. The focus on “effect” of the copied features is new to Canadian law, and it remains to be seen how this test will be applied to the infringement analysis going forward, including in relation to software APIs.

The *Cinar* decision did not involve computer software, and notably, the Court left the door open for the abstraction-filtration-comparison method being used with respect to software. Further, the Court acknowledged that it may be necessary to go beyond the perspective of the audience (for example, software user), to call upon an expert to put the judge in the shoes of “someone reasonably versed in the relevant technology”. Consequently, the approach in *Delrina* may still be followed in future cases.

If the abstraction-filtration-comparison method is applied, the "structure, sequence, and organization" of the API may be “weeded out” from copyright protection because, for example, the elements could only be expressed, or programmed, in one or a very limited number of ways. A finding of infringement under this test may also depend largely on whether the structure, sequence, and organization of an allegedly infringing API amounts to a “substantial part” of the plaintiff’s skill and judgment in creating the infringed computer program. Unfortunately, there is nothing inherent about APIs that suggests that this question can be answered consistently one way or the other. In a simple API, there may be relatively few ways to organize a list of functions, meaning that the API structure would not form a substantial part of the author’s use of skill and judgment in creating the API. An alleged infringer might point out known design patterns in the copyright owner’s API

specification (e.g. “create, read, update and delete”) to argue that the author’s skill and judgment was mostly expended elsewhere (e.g. in the writing of the implementation). On the other hand, the author of a more complicated API specification might argue that writing the implementation – once the organization of the API specification had been settled – was a largely mechanical exercise. Therefore, a substantial part of the author’s skill and judgment must have gone into devising the structure, sequence, and organization of that API.

Conversely, if a holistic and qualitative approach is applied, a court may find infringement based on the “overall architecture” of the API having been copied. The Court may find that copying of the overall architecture is not a reproduction of an abstract idea, but rather copying of the detailed manner in which the ideas are expressed. Applying a holistic test, infringement could be found based on non-literal copying of a combination of features of the API. The court may regard such combination of features as a substantial part of the API, even if they do not form a discrete part of the API, but are abstracted from the API.

Conclusion

As ever, it is difficult to predict how courts will rule on a novel issue. Judgments from foreign courts can have some predictive value, but are not determinative given varying legal frameworks and traditions. In Canada, it is possible that copying original function naming could form the basis for a finding of copyright infringement. However, whether infringement would be found for copying the structure, sequence and organization, or “overall architecture”, of an API will likely depend on the specific facts of each case.