ASPEN PUBLISHERS

The COMPUTER & INTERNET Lawyer

Volume 23 ▲ Number 10 ▲ OCTOBER 2006

Arnold & Porter, Editor-in-Chief *

Source Code, Object Code, and *The Da Vinci Code*: The Debate on Intellectual Property Protection for Software Programs

By Murali Neelakantan and Alex Armstrong

For many years, the courts in England and the United States have tried to balance the protection of an author's skill and labor with the competing notion of a free market in which ideas are adapted in the search for newer and better products. It is sometimes argued that "what is worth copying is worth protecting." This statement is only a crude approximation of the central theme in a debate that remains as controversial now as it was 25 years ago—the suitability of a copyright or a patent-based regime for software programs.

For purposes of this article, we have restricted our analysis to the two markets that are likely to be of most interest to the reader—the United States and the United Kingdom. The debate has, however, taken on truly global proportions, with new and exciting markets (and competitive pressures) emanating from the Far East, India, and China most notably. The questions asked in this article are designed therefore to apply globally.

Murali Neelakantan is a dual qualified (Indian and English Law) partner and **Alex Armstrong** is an associate at the London office of Arnold & Porter LLP. Arnold & Porter acted for the defendants in the *Da Vinci Code* case.

Copyright in Software Programs

The ease with which copyright is granted sometimes betrays its limitations. Is copyright still "fit for purpose" as the global market for software continues its inexorable expansion?

In order to answer this question, this section will seek to:

- Examine the existing state of copyright law as it applies to software programs; and
- Determine whether current copyright law remains flexible enough to capture the dramatic changes to the methods used by developers to create software programs.

The English Law of Copyright

The English law of copyright is often described as drawing clear dividing lines between the idea (which is not protectable *per*

Wolters Kluwer
Law & Business

se) and the expression of an idea (which would be). This is a misleading simplification of the relevant provision of the Copyright, Designs and Patent Act 1988,[1] which requires that a work be recorded "in writing or otherwise"[2] before it can be afforded the protection of copyright. The law says that copyright is infringed if (a) there has been actual copying, and (b) a "substantial part" of the work has been taken. What amounts to a "substantial part" is a question of fact and degree and is the question that has exercised the courts most in the field of computer software.

In *Cantor Fitzgerald v. Tradition (UK),*[3] the court considered whether the developers of a rival bond-broking application had infringed the copyright in the claimant's program. The defendants were ex-employees of the claimant and had used an earlier version of the claimant's program as a reference for their own application. The court also found that the defendants had copied a small portion (3.3 percent) of the claimant's code into the defendants' own program. The judge held on the facts that there had been specific instances of copying by the defendants and that the copying had, in these instances, amounted to a substantial part of each module concerned.

The case seeks to shed some light on how the law of copyright can be applied to software programs. The judge in *Cantor Fitzgerald* made some important points:

- It was possible for the defendants to infringe the claimant's copyright at the "architecture" level, that is, its overall structure and how the program allocated certain functions to the various component modules.

- The definition of what constituted a "substantial part" of a software program required the court to consider each work as a whole, not the individual portions of code. He disagreed with the High Court of Australia in *Autodesk v. Dyason* (1992), which had found that any portion of code, no matter how small, would form a "substantial part" of the work since in its absence the application as a whole would fail to function correctly or at all. The Australian court's approach was technologically correct but legally inaccurate.

- In determining whether the infringing product copied a substantial part of the claimant's product, the court would assess the existence of copyright in the claimant's code as a function of the amount of skill and effort that had gone into designing and developing it.

The US Law of Copyright

In the United States, a work may be subject to copyright protection if it is both: (1) original and (2) "fixed in any tangible medium of expression."[4] Copyright protection is not available to any "idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in [the] work."[5]

In *Computer Associates v. Altai*,[6] Computer Associates alleged that Altai's "Oscar" program contained elements of Computer Associates' "Adapter" program. The court approached the problem by undertaking a series of detailed analyses into the claimant's program. It first analyzed the "level of abstraction," retracing the developer's steps back from the final object code through to the program's conception. The court then proceeded to "filter out" elements of the program that were (1) dictated by efficiency; (2) dictated by external factors; and (3) taken from the public domain.[7]

The court was able to group its findings into a "core of protectable expression," an approximate equivalent to the concept of "a substantial part" in English law. On the facts, the court found that the defendants had not appropriated this core of protectable expression.

It should be noted that the court considered the merger doctrine as one of its central themes in determining the core of protectable expression. The doctrine's underlying principle is that "[w]hen there is essentially only one way to express an idea, the idea and its expression are inseparable and copyright is no bar to copying that expression."[8] In *Computer Associates*, the court felt that the merger doctrine was "an effective way to eliminate non-protectable expression contained in computer programs" because it allowed the court to disregard those elements of a computer program that could only be expressed in a certain way.

Comparisons Between the UK and US Regimes

The analysis of the judge in *Cantor Fitzgerald* is similar to the abstraction exercise of the US Court of Appeals in *Computer Associates*. Both regimes seek to go beyond the concept of pure textual copying and arrive at a broadly similar result, although the key difference remains the refusal by the English courts to recognize the validity of the merger doctrine. English courts will consider originality as a function of the skill and effort used to create the work, even if in some cases there is no real scope for alternative expression, and will refuse to determine originality through subjective judgment of the ideas that underpin the work. The genesis of this position lies in the numerous compilation cases, most notably *Ladbroke v. William Hill* (1964) and the position reflects a desire by the English courts to remove subjectivity from the assessment of what is and is not copyrightable and "compensate for a lack of a roving concept of unfair competition."[9]

Can we therefore conclude that the English courts will value hard work rather than protect an original and good idea?

The Da Vinci Code Decision: Placing a Value on Ideas?

Does the recent decision in *Baigent and Leigh v. Random House* (2006) (*Da Vinci Code*) hint at a gradual shift towards the US position?

As in the *Cantor Fitzgerald* case, the judge in the *Da Vinci Code* case was looking for the expression of a combination of ideas, structure, and content that, taken together, constituted a substantial part of the earlier work. The judge referred to another case of literary copying (*Ravenscroft v. Herbert* (1980)) in order to introduce the established rule that while facts, themes, and ideas cannot be protected *per se*, the way in which these facts, themes, and ideas are put together (the work's "architecture") could be. The judge went on to say that the claimants would need to "show that there is a putting together of facts, themes and ideas by them as a result of their efforts" and that Dan Brown had copied these. He then undertook a painstaking analysis of *The Holy Blood and Holy Grail* in order to draw a firm boundary between what was or was not protected.

The English law of copyright has long established that this form of abstraction is a qualitative test rather than a quantitative one.[10] It is a test that is similar in many ways to that of the court in *Cantor Fitzgerald*. The judge decided that, while Mr. Brown had used facts and some of the "central themes" that were contained in *The Holy Blood and Holy Grail*, facts were not copyrightable *per se* and that the "central themes" reproduced in *The Da Vinci Code* were too abstract to constitute a "substantial part" of the earlier work. Coupled with the finding that Mr. Brown and his wife had devoted significant skill and effort to the research and development of the content of *The Da Vinci Code*, it was no surprise that the judge found in favor of the defendants. The result was that Mr. Brown was entitled to express the themes he had picked from *The Holy Blood and Holy Grail* in a way and using methods that were his own.

The factual outcome of the case may have little or no relevance to software developers. Indeed the courts in England have always been uncomfortable about applying tests used to determine substantiality in purely literary cases to those involving software. It could be argued, however, that the analysis undertaken by the judge as part of his assessment of "substantiality" appeared to imply the need to consider the originality of an idea, at least as part of the overall test.[11]

The Problem

As it currently stands, copyright law (as it applies to software developers) can be summarized as follows:

- The developer's ideas cannot be protected to the extent they cannot be expressed on a medium.
- Therefore, object code, source code, program structures and program notes (taken together) constitute the expression of the developer's idea.
- The expression of the developer's ideas cannot be protected to the extent that the ideas can be expressed differently by someone else without reference to the developer's object code, source code, program structures, and program notes.

Does this emphasis on "perspiration" rather than "inspiration" capture the ways in which software developers create their products?

Software development has evolved significantly since the early days of programming. Programs have become more complex and the industry has grown in such a way that developers are increasingly reliant on shorter development cycles in order to preserve or enhance their competitive advantage. This has led the industry to adopt a natural way of making itself more efficient. Ever since the software industry evolved toward modular or object-orientated class-based programming (modules of code woven together to create a new piece of code), vast libraries of pre-packaged code have become available to many software developers who wish to use them under license in their own works. Software is now a complex mixture of *source code* (code created by the developer using a more human, high-level language) and *object code* (generally low-level machine-specific code, usually expressed as a collection of binary digits to call specific functions of a computer system).

Source code is usually compiled (translated into object code) and then linked to static and dynamically-linked object code libraries (some of which are off-the-shelf, others licensed from other developers), or increasingly linked to complex databases.

As a result of this push for standardization, it could be argued that the problem that software developers now face as a result of this shift in programming technique lies less in their ability to protect their code, but more in their ability to protect the algorithms (the ideas) that are expressed in their code. Supporters of the merger doctrine in the US recognize this issue,[12] which is the essence of the copyright versus patent debate as it applies to software programs.

The Expression of the Problem

> The main disadvantage of copyrights as a protection for algorithms is that copyrights do not protect the functionality or the technique of an algorithm. This disadvantage is insignificant if the algorithm is not the essence of the computer program. For instance, in video games, the meaningful part of the computer program is the interaction with the user, not the method used for solving a problem. However, when an algorithm is developed as a new method for solving a problem, the general idea and functionality of the algorithm—*i.e.,* the inventive leaps—[are] not protected.[13]

For the developer who seeks to protect a truly new algorithm, this creates two classes of commercial problems, which can be expressed as follows:

- **Problem 1:** Two or more developers discover the same algorithm separately and then go on to express the same algorithm using different program structures and different code. The algorithm is new and original. They are each equally deserving of the protection of copyright law, but in absolute commercial terms, they will share the potential market for the algorithm with the other.

- **Problem 2:** One developer discovers a new and original algorithm and then expresses this algorithm using a unique program structure and unique code. He has copyright protection in respect of the idea as expressed in the code, but in absolute commercial terms does not have sole control of the potential market for the algorithm; another developer could express the algorithm (the idea) differently using different code and market a rival product accordingly.

These problems share a common feature: The developer is not in absolute control of the market generated by his idea. If, in the commercial sphere, the goal of investing time and effort in creating new ideas and solutions is to achieve optimum commercial gain, then should not the intellectual property grant a monopoly right to the creator of these ideas or solutions?

Is Patent Protection the Answer?

In the United Kingdom, software is not patentable, but the debate over whether it should be is conjoined with the continuing debate in the European Parliament over the ambit of the proposed Directive on computer implemented inventions.

In the United States, software and business processes are patentable, but whether this system has a positive impact on the market is debatable.

Evolution

To obtain the benefit of patent protection, an invention must be (1) patent-eligible subject matter; (2) useful; (3) novel; and (4) nonobvious.[14]

The US courts have taken a gradual road toward granting software products the benefit of patent protection. Unlike the European Union, where the debate rages over the need to implement a community-wide software patent Directive, the US courts have adapted their position with regard to the patentability of software and business processes. Since the 1981 case *Diamond v. Diehr*[15] (in which the US Supreme Court ordered the USPTO to grant a patent in relation to an invention even though the substantial part of the invention consisted of a computer program), the USPTO has gradually extended patent protection to a wide variety of computer-based software products and business processes.

An algorithm (as applied to computer programs) can be thought of as a machine—a computer that is hard-wired to perform the algorithm. In *State Street Bank & Trust Co. v. Signature Financial Group, Inc.,*[16] the Federal Circuit held that an algorithm is capable of receiving patent protection if it is useful, concrete, and produces a tangible result.

Having overcome the conceptual difficulty of accepting algorithms as patentable *per se*, the courts brought further refinements to the requirements of usefulness, novelty, and non-obviousness. There is now a settled regime for the protection of algorithms, the foundations of any computer software program. The gradual introduction of this additional layer of protection for algorithms, however, while welcomed by some, has had unusual side effects on the US software industry.

Distortion?

Since the gradual extension of patent protection to software programs, the software industry has witnessed a significant growth in the number of patents being sought by large organizations. As of December 2003, "software and Internet-related patents account[ed] for more than 15 percent of all patents granted."[17] Large organizations are pursuing patents for two primary purposes:

1. **Revenue generation**: monopolizing ideas with a view to licensing them to developers who will transform or incorporate them into tangible products; and

2. **Deterrent value**: warding off the holders of and applicants for patents for similar ideas with the threat of protracted and necessarily expensive litigation.

The clearest manifestation of these side effects occurred recently in connection with the high-profile litigation between Canada's Research in Motion Limited (the makers of the Blackberry handheld device) and NTP Inc. (a US-based company). NTP's sole purpose, it appears, is to acquire and maintain a portfolio of patents with a view to bringing infringement proceedings against any person or organization that attempts to use any technology that is protected by those patents (a so-called patent troll). NTP sued Research in Motion alleging infringement of several of NTP's patents. The case went through the litigation process and ultimately ended with Research in Motion agreeing to pay the sum of US $613 million to NTP in settlement of all claims. The case demonstrates a fundamental feature of (or problem with) the modern US intellectual property protection system.

Research in Motion had spent a lot of time, effort, and money developing the system from which it has reaped stellar commercial rewards. The US district court was concerned only with the violation of the monopoly right granted in respect of NTP's idea, however, not its expression as a tangible product. The fact that NTP had tried (and failed) to market its invention 14 years prior was no bar to its preventing Research in Motion from marketing its own products using NTP's idea notwithstanding the value (both commercial and tangible) of the Research in Motion product.

The case is significant because it polarizes the debate on the suitability of patent protection for software programs. It gives further ammunition to those who believe that ideas should be prized above endeavor and commercial skill, and it reinforces the argument from others that extending patent protection for software programs creates unnatural distortions in a market that has thrived against the backdrop of existing copyright laws.

Despite the obvious passions expressed by those on opposing sides, it is possible to view the debate in pragmatic, commercial terms.

The Market Forces Argument

The debate over the respective merits of copyright protection and patent protection is driven by one central theme: appropriate commercial rewards for the creators of innovative software programs through control of the markets for which their products were created.

A Commercial Question

The current differences in the regimes that exist in England and in the United States show that developers are caught between two conflicting pressures: (1) inadequate protection (the current risk with existing English-style copyright-only regimes) and (2) too much protection (the stifling of healthy competition and the creation of unnatural distortions in the market, as seen in US-style patent regimes).

Which regime is best suited to the developer who creates a software product based on a new idea?

The Commercial Answer: Theory

The debate over the suitability of copyright or patent protection as the most effective means of protecting a developer's research and development often ignores one crucial point: The market ultimately decides whether a product succeeds or fails. A developer can make a commercial success out of a new idea if he is able to follow three simple rules:

1. His idea is original and unique.

2. He can translate that idea into software ahead of the competition.

3. He can bring the resulting product to market efficiently ahead of the competition.

These three rules work because the market recognizes the value of marketing new and original ideas ahead of the competition. It allows the developer to enjoy a *de facto* monopoly while the others try to catch up.

The Commercial Answer: Applied

Even with the comparatively weaker protections offered by copyright, we know that a rival cannot develop a competing product quickly from scratch. He cannot avoid the time penalties for which the market will penalize him by simply reverse engineering a product and adapting only superficial aspects of it to disguise the infringement. He would (1) fall foul of the law and (2) would not fool the market. In addition, if one released a product to the market (having kept the idea secret), then regardless of how the law protects the idea, one can release a newer and improved version of the end product by the time a rival has caught up. In other words, an inventor can maintain a competitive advantage for as long as he develops and improves an idea.

As all developers know, turning a good idea into a successful product is hard work. The idea is only the beginning. The *Da Vinci Code* case confirms the view that the English courts will consider (but ultimately subordinate) ideas and themes to the skill and effort used to express them. Perhaps this is no accident. Genius, after all, is 1 percent inspiration and 9 percent perspiration.

Notes

1. Sections 3(2), 178 (for definition of "writing").
2. *Id.*

3. Cantor Fitzgerald v Tradition (UK), RPC 95 (2000).
4. 17 U.S.C. § 102(a).
5. *Id*. § 102(b).
6. Computer Associates v Altai, 982 F.2d 693 (1992), 23 U.S.P.Q. 2d 1241.
7. The court explained the general concept of the abstraction process as follows: "In ascertaining substantial similarity under this approach, a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material."
8. Concrete Machinery Co. v. Classic Lawn Ornaments, Inc., 843 F.2d 600, 606 (1st Cir. 1988).
9. Cornish & Llewelyn, Intellectual Property: Patents, Copyright, Trade Marks and Allied Rights (5th ed.) at 391.
10. *See* Ladbroke v. William Hill, 1 W.L.R. 273 (1964).
11. Baigent & Leight v. Random House, EWHC 719 (2006), ¶¶ 268 and 270.
12. *See* Melville B. Nimmer & David Nimmer, Nimmer on Copyright, § 13.01, at 13-65 and 13-66-71: "[I]n many instances it is virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques. [. . .] This is a result of the fact that a programmer's freedom of design choice is often circumscribed by extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designated to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry."
13. Allen Clark Zoracki, "Comment: When Is An Algorithm Invented? The Need For A New Paradigm For Evaluating An Algorithm For Intellectual Property," 15 *Alb. L.J. Sci. & Tech*. 579 (2005).
14. 35 U.S.C. §§101-103.
15. Diamond v. Diehr, 450 U.S. 175 (1981).
16. State Street Bank & Trust Co. v. Signature Financial Group, Inc., 149 F.3d 1368, 1373 (Fed. Cir. 1998).
17. Jonathan Krim, "Patenting Air or Protecting Property? Information Age Invents a New Problem," *Wash. Post*, Dec. 11, 2003.

Reprinted from *The Computer & Internet Lawyer*, October 2006, Volume 23, Number 10, pages 1 to 5, with permission from Aspen Publishers, Inc., a Wolters Kluwer business, New York, NY, 1-800-638-8437, *www.aspenpublishers.com*.