CopyrightWORLD

Source code, object code & the Da Vinci code

Arnold & Porter's Murali Neelakantan and Alex Armstrong examine the cross Atlantic debate on IPR protection for software programs

IN SUMMARY

– The suitability of a copyright or patent-based regime for software programs is as controversial a debate now as it was 25 years ago

– Both UK and US copyright regimes seek to go beyond the concept of pure textual copying and arrive at a broadly similar result, though differences remain

– This article discusses whether the *Da Vinci Code* case hints at a gradual shift towards the US position, whether patent protection is the answer and whether the market provides its own form of protection

AUTHOR

Murali Neelakantan (bottom left) is a dual qualified (Indian & English) lawyer and partner in the intellectual property group at the London office of Arnold & Porter. His practice is broad based and he has advised on a wide variety of matters including offshore IT, IP and business process outsourcing. He is also regularly quoted as an expert on Indian matters.

Alex Armstrong (bottom right) is an English solicitor and associate in the intellectual property group at the London office of Arnold & Porter. He advises clients on a broad range of information technology matters including outsourcing, hardware supply and support, software licensing and project agreements.

"*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. This article analyses the UK and US approaches to protecting the intellectual property rights ("IPR") of software programs and questions whether protection is required in such a dynamic market place.

Copyright in the UK

In the UK 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)*[1], the court considered whether the developers of a rival bond-broking application had infringed the claimant's copyright by using an earlier version of the claimant's program as a reference for their own application. A small portion, 3.3%, of the claimant's code was found to have been used. 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 established that it was possible for defendants to infringe a claimant's copyright at the *"architecture"* level (i.e. a software program's overall structure and how the program allocated certain functions to the various component modules once each work had been analysed as a whole and not as individual portions of code). To determine whether the infringing product copied a substantial part of the claimant's product, the court assessed the existence of copyright in the claimant's code as a function of the amount of skill and effort that has gone into designing and developing it.

Copyright in the US

In the US, copyright is granted to a work which is both: a) original, and b) "fixed in any tangible medium of expression."[2] Copyright is not afforded 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"[3].

In *Computer Associates v Altai (1992)*[4], Computer Associates alleged that Altai's *"Oscar"* program contained elements of Computer Associates' *"Adapter"* program. The Court analysed the claimant's program for the *"level of abstraction"*, retracing the developer's steps back from the final object code through to the program's conception, then proceeded to *"filter out"* elements of the program (those, dictated by efficiency, by external factors and taken from the public domain[5]) and finally grouped its findings into a *"core of protectable expression"*[6].

The US courts have identified the *"merger doctrine"* as one of its central themes in determining the *"core of protectable expression"*. The underlying principle of the doctrine 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" (Concrete Machinery Co. v. Classic Lawn Ornaments, Inc.[7])*. In the *Computer Associates* case, 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 which could only be expressed in a certain way.

Comparisons of the UK and US regimes

The exercise undertaken by the judge in *Cantor Fitzgerald* is similar to the abstraction exercise undertaken by 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 recognise the validity of the *"merger doctrine"*.

Can we therefore conclude that the English courts will value hard work rather than protect an original and good idea, and does the recent decision in *Baigent and Leigh v Random House (2006)* ("the *Da Vinci Code* case") hint at a gradual shift towards the American position?

The Da Vinci Code decision – placing a value on ideas?

As in the *Cantor Fitzgerald* case, the judge in the *Da Vinci Code* case looked for the expression of a combination of ideas, structure and content which taken together constituted a substantial part of the earlier work[8]. The claimants were required to show that there had been a *"putting together of facts, themes and ideas by them as a result of their efforts"* and that Dan Brown had copied these.

The English law of copyright has long established that this form of abstraction is a qualitative rather than a quantitative test[9]. The judge decided that while Mr Brown had used facts and some of the *"central themes"* which were contained in *The Holy Blood and Holy Grail*, facts were not copyrightable *per se* and the *"central themes"* reproduced in *The Da Vinci Code* were too abstract to constitute a *"substantial part"* of the earlier work. This conclusion, combined with a 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*, led him to find in their favour.

From the above analysis it could be argued that when assessing *"substantiality"* there is a need to consider the originality of an idea, at least as part of the overall test,[10] although the English courts have

been wary to apply literary cases to those involving software.

Is copyright a suitable form of protection for software developers?

Problems arise for software developers when their 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 they can be expressed differently by someone else without reference to the developer's object code, source code, program structures and program notes.

"The expression of the developer's ideas cannot be protected to the extent they can be expressed differently by someone else without reference to the developer's object code, source code, program structures and program notes"

Is patent protection the answer?

While software is not patentable in the UK, software and business processes are patentable in the US, although it is arguable whether this has had a positive impact on the market.

In order to obtain the benefit of patent protection in the US, an invention must satisfy four criteria: (1) patent-eligible subject matter; (2) useful; (3) novel; and (4) non-obvious[11].

The US courts have taken a gradual road towards granting software products the benefit of patent protection. Since the 1981 case of *Diamond v. Diehr* (where 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. In the case of *State Street Bank & Trust Co. v. Signature Financial Group, Inc*[12], the Court held that an algorithm is capable of receiving patent protection if it is useful, concrete, and produces a tangible result. These cases aided the establishment of the now settled regime for the protection of algorithms – the foundations of any computer software program. This additional layer of protection for algorithms, while welcomed by some, has had some unusual side-effects on the software industry in the US.

With 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 organisations (i.e. to generate revenue and to ward off the holders of applications for patents for similar ideas). The high-profile litigation between Canada's Research in Motion Limited ("RIM") the makers of the Blackberry handheld device and NTP Inc.(a US-based company), being a case in point.

RIM had spent a lot of time, effort and money developing the Blackberry system. However, the US District Court was only concerned with the violation of the monopoly rights granted in respect of NTP's idea, not their expression as a tangible product. The fact that NTP had tried (and failed) to market its invention 14 years ago was no bar to it preventing RIM from marketing its own products using NTP's idea notwithstanding the value (both commercial and tangible) of the RIM product.

The case is significant because it polarises the debate on the suitability of patent protection for software programs and highlights potential problems software developers face when trying to rely on copyright protection for their programs.

Does the market provide its own form of protection?

The debate over the respective merits of copyright protection and patent protection are all 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. Software developers are currently caught between the inadequacies of the copyright regimes in the UK and US and too much protection from the US patent regime. Are either of these suited to the developer who creates a software product based on a new idea?

Discussions surrounding the suitability of copyright or patent protection as the most effective means of protecting a developer's research and development often ignore 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 generate an original and unique idea that can be made into software ahead of the competition and then marketed efficiently ahead of the competition. The developers enjoy a *de facto* monopoly while their competitors try to catch up.

Even with the comparatively weaker protections offered by copyright, a rival cannot develop a competing product quickly from scratch. A competitor cannot avoid the time penalties for which the market will penalise him by simply reverse-engineering a product and adapting only superficial aspects of it to disguise the infringement. He would (a) fall foul of the law and (b) would not fool the market. In addition, if a developer released a product to the market (having kept the idea secret) then regardless of how the law protects the idea, he can release a newer and improved version of the end product by the time a rival has caught up. In other words, competitive advantage is maintained for as long as the developer continues to develop and improve an idea.

Notes

1 [2000] RPC 95
2 *17 U.S.C. at 102*(a).
3 Ibid. 102(b).
4 982 F.2d 693, 23 USPQ2d 1241
5 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."
6 An approximate equivalent to the concept of *"a substantial part"* in English law
7 43 F.2d 600, 606 [6 USPQ 1357] (1st Cir. 1988).
8 Relying on the literary works case of *Ravenscroft v Herbert* [1980] to establish 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.
9 See *Ladbroke v William Hill* [1964] 1 W.L.R 273.
10 *Baigent & Leight v Random House* [2006] EWHC 719, paragraphs 268 and 270
11 *35 U.S.C.* 101-103.
12 *149 F.3d 1368, 1373 (Fed. Cir. 1998)*.