Chapter XXV
Legal and Economic Justification for Software Protection

Bruno de Vuyst
Vrije Universiteit Brussel, Belgium

Alea Fairchild
Vrije Universiteit Brussel, Belgium

ABSTRACT

This chapter discusses legal and economic rationale in regards to open source software protection. Software programs are, under TRIPS[1], protected by copyright (reference is made to the Berne Convention[2]). The issue with this protection is that, due to the dichotomy idea/expression that is typical for copyright protection, reverse engineering of software is not excluded, and copyright is hence found to be an insufficient protection. Hence, in the U.S., software makers have increasingly turned to patent protection. In Europe, there is an exclusion of computer programs in Article 52 (2) c) EPC (EPO, 1973), but this exclusion is increasingly narrowed and some call for abandoning the exclusion altogether. A proposal by the European Commission, made in 2002, called for a directive to allow national patent authorities to patent software in a broader way, so as to ensure further against reverse engineering; this proposal, however, was shelved in 2005 over active opposition within and outside the European parliament. In summary, open source software does not fit in any proprietary model; rather, it creates a freedom to operate. Ultimately, there is a need to rethink approaches to property law so as to allow for viable software packaging in both models.

INTRODUCTION

Copyright Protection of Software

A software program is foremost a sequence of orders and mathematical algorithms emerging from the mind of the innovator, hence creating a link with copyright law as a prime source of intellectual property protection.

According to Article 10 TRIPS, computer programs, whether in source or object code, shall be protected as literary works under the Berne

Convention provided that they are (1) original and (2) tangible. In light of Article 9 TRIPS, which states that copyright protection shall extend to expressions, but not to ideas, procedures, methods of operation or mathematical concepts as such, copyright protects the actual code of the computer program itself, and the way the instructions have been drawn up, but not the underlying idea thereof (Overdijk, 1999).

Hence, an author can protect his original work against unauthorized copying. Consequently, an independent creation from another person would not automatically be seen as a copyright infringement (Kirsch, 2000a; Leijnse, 2003). With respect to software programs this could have as consequence that a person disassembles and decompiles an existing software program to determine the underlying idea and uses this idea to build his own program (reverse engineering). As he only uses the idea, which is not copyrightable, no infringement will result.

BACKGROUND

Patent Law Protection of Software

Software is a novel form in the technology world, and may make a claim to patent protection from that angle. The conditions to be met to enjoy patent protection are more stringent than those to enjoy copyright protection. In Europe[3], for example, an invention will enjoy protection from patent law provided that the invention (1) is new (i.e., never been produced before), (2) is based on inventor activity (i.e., not have been before part of prior art), and (3) makes a technical contribution (i.e., contribute to the state of the art). In the U.S., the patent requirements to be met are (1) novelty, (2) non-obviousness, and (3) the innovations must fall within the statutory class of patentable inventions.

Pursuant to patent law, a patent holder can invoke the protection of his patent to exclude others from making, using or selling the patented invention. As opposed to copyright protection, the inventor's patent is protected regardless whether the software code of the patented program was copied or not.

The Evolution of the Legal Protection of Software

Prior to the 1980s, U.S. courts unanimously held that software was not patentable and that its only protection could be found in copyright. Indeed, the U.S. Supreme Court ruled in two landmark decisions, Gottschalk vs. Benson (1972) and Parker vs. Flook (1978), that software was similar to mathematics and laws of nature (both excluded from being patented) and, therefore, was unpatentable.

In Diamond vs. Diehr (1981), however, the court reversed course, deciding that an invention was not necessarily unpatentable simply because it utilized software. Since this decision, U.S. courts as well as the US Patent Office gradually broadened the scope of protection available for software-related inventions (Kirsch, 2000). The situation evolved to the current status in which it is expected to obtain a patent for software-related inventions. Since the State Street Bank and Trust Co. vs. Signature Financial Group Inc. (1996) case even mathematical algorithms and business methods have been found to be patentable (see also the Amazon One-click case IPXL Holding, plc vs. Amazon.com, Inc., 2005; Bakels , 2003). As from this decision, the U.S. focus, for patentability, is "utility based," which is defined as "the essential characteristics of the subject matter" and the key to patentability is the production of a "useful, concrete and tangible result" (Hart, Holmes, & Reid, 1999). The evolution resulted in a rush of patent applications for software-related inventions and business methodologies.

Contrary to the U.S., Europe has been unwilling to grant patents for ideas, business processes and software programs. The most important rea-