Functionality and Expression in Computer Programs

Pamela Samuelson

It would be convenient if we could assign patent law the role of protecting program functionality and copyright law the role of protecting program expression. At a high level of generality, this seems an intuitively sound approach, and in some instances, it may be relatively workable. However, separating functionality and expression in programs is a daunting challenge, as the *Oracle v. Google* case illustrates.

At issue in *Oracle v. Google* is whether the structure, sequence and organization (SSO) of the 37 Java application program interface (API) packages that Google incorporated into its Android platform software are protectable by copyright law. Android software now resides on more than 250 million smart phones worldwide. A trial judge ruled that these API elements constituted a system or method of operation for which copyright protection was unavailable and alternatively that functionality and expression in the APIs had merged. The judge noted that Oracle and Sun (Oracle's predecessor-in-interest in intellectual property rights in Java technologies) had sought patent protection for some aspects of the Java API. The judge perceived this to be an indication that these API elements were more appropriate for patent than for copyright protection.

The Court of Appeals for the Federal Circuit (CAFC) disagreed and reversed the trial court's ruling in Google's favor. It rejected as unsound Google's system/method of operation and merger defenses. It treated Google's patents-for-APIs defense as an attack on copyright protection for computer programs. The CAFC felt bound by precedents and the statute to reject this defense. It found untroubling the notion that patents and copyrights might both be available to protect the Java APIs.

One cornerstone of Google's petition for Supreme Court review is its contention that Oracle is trying through its copyright claim to do an "end run" around the patent system—that is, to get a much longer duration of protection from copyright than patent would provide without satisfying the novelty, nonobviousness, application, and examination requirements for obtaining patent protection. Google relies on the Supreme Court's venerable 1880 *Baker v. Selden* decision in support of this claim.

Baker held that Selden's copyright extended only to its author's explanation of his novel bookkeeping system, not to the system itself and to the forms in the book that were implementations of the system.

The Court regarded the bookkeeping system as a useful art that could be legally protected, if at all, only by a patent. Yet, 115 years later when the Court took the *Lotus v. Borland* case which involved a spreadsheet program (which was a kind of electronic literary work to facilitate bookkeeping), it split 4-4 on whether a command hierarchy for the Lotus 1-2-3 program was protectable expression or an unprotectable system or method of operation. Borland had sought to bolster the First Circuit's ruling by citing to patents for command

hierarchies as evidence they were patent, not copyright, subject matter.

This article offers a pragmatic approach to conceptualizing the roles of different types of intellectual property protections for computer programs and innovative elements they embody. It begins in Part I by explaining why computer software has proven to be a difficult misfit with conventional doctrines of patent and copyright laws. It discusses what is settled and unsettled in the case law as to the roles of patent and copyright in protecting functionality and expression in programs.

Part II concentrates on program APIs and command hierarchies as examples of elements of computer program as to which it is extremely difficult to distinguish functionality and expression or to assign workable roles for patent and copyright protections. It focuses mainly on the *Lotus v. Borland* and *Oracle v. Google* cases to illustrate different ways that courts have grappled with this deep problem.