By Pamela Samuelson

legally speaking

Interface Specifications, Compatibility, and Intellectual Property Law

Perhaps you have read about the lawsuit currently pending between Atari and Nintendo in which Nintendo claims intellectual property rights in the interface between its video game console and cartridges designed for use in the Nintendo console as a basis for blocking Atari's right to develop and sell compatible video game cartridges for the Nintendo machines. The outcome of this lawsuit may have profound implications for competition in the software industry, as well as for the price you will pay for game cartridges for your child's Nintendo machine (which you may have noticed are not at all cheap).

Nintendo claims several intellectual property rights in its interface. It holds a patent for a method of verifying, by software installed in a processing unit in the video game hardware console and in processing units of the individual game cartridges, that the game cartridge has the interface necessary to operate in the Nintendo console. Nintendo also claims copyright protection for the software in both the console and the game cartridges. Nintendo claims Atari has infringed its copyrights because, although Atari's interface program is not identical to Nintendo's, Atari's program produces the same sequence of bits as the Nintendo program, and consequently Atari is said to be "singing a Nintendo-owned tune." Nintendo further claims that the detailed internal interface specifications embedded in the software are its trade secrets which Atari obtained unlawfully by making unauthorized copies of the interface programs in order to reverse engineer them, thereby infringing the program copyrights.

Atari claims that Nintendo's patent is invalid because of a prior patent on a "lockout" interface. Atari also claims that even if the Nintendo patent is valid, the patent is not infringed by Atari's Nintendo-compatible game cartridges because the Atari cartridges accomplish the same result as the Nintendo cartridges in a different and noninfringing way. Atari also insists that the reverse engineering it may have done to learn the details of Nintendo's software interface was a lawful way to acquire any trade secrets Nintendo might have had in the interface, and did not infringe the copyright. Atari further asserts that copyright protection is not available for interface information and that since there are functional reasons why the Atari and Nintendo programs produce identical sequences of bits, there can be no copyright infringement based on the "tune" theory. Atari also charges that Nintendo is guilty of unlawful monopolization of the home video game market, pointing to the firm's effort to exclude any but Nintendo-licensed firms from selling Nintendo-compatible game cartridges, as well as some other practices, to show that Nintendo has been artificially constricting the availability of games for the Nintendo console and thereby charging monopoly prices for them.

Nintendo responds to these charges by asserting that the control it is exercising over the development and sale of Nintendo-compatible game cartridges is necessary to keep the quality of games high so that the market for home video games does not collapse again as it did some years ago. Having essentially recreated the market for home video games by manufacturing and marketing a superior machine and a set of graphically pleasing, challenging, and desirable games, Nintendo thinks it deserves the benefit of the success it has created and regards Atari as a free-riding interloper. Nintendo also claims only to be exercising the intellectual property rights in its video game interface that it is entitled by law to assert.

Because the Atari/Nintendo case involves three different types of intellectual property rights—each of which addresses the issue of ownership rights of computer interfaces somewhat differently—this column will now discuss how trade secret law, copyright, and patent law would likely treat claims to ownership of computer interfaces. The column will also try to place this dispute in an historical context for the software industry. The expectations some firms in the software industry have about their right to assert property rights in interface information contained in software was derived from the era in which trade secret licensing was the predominant mode of protecting software—a set of expectations which some now think is inappropriate now that the software has become a mass-market commodity. Moreover, the competitive importance of compatibility has now been recognized by consumers and by the industry itself.

In the 1960s and 1970s, as the software industry began to develop, there was much uncertainty about whether either copyright or patent protection was available to protect computer programs. (Copyright has traditionally not been available to protect technological works; patent has traditionally not been available for processes that do not transform matter.) As a result, it became common practice in the computer industry to protect intellectual property rights in software through trade secret licensing agreements. Such agreements tended to define the rights of users in the software and any accompanying documentation in quite a restrictive manner.

Because it was often not discernible merely by using a program, interface information was among the set of internal details of programs that software developers wanted to protect as trade secrets. Separate licensing arrangements would have to be made if a firm wanted to use another firm's secret interface information to create a compatible product. In time, some firms found it more beneficial to publish interface information (or license it on a low royalty basis) to those who might develop software for their computer system. The more software available for a particular firm's computer, the more desirable, in general, was that firm's computer system. And because of this, firms often had incentives to reveal interface information, rather than keeping it a trade secret.

In the late 1970s, as software was on the verge of becoming a mass-market commodity, it became apparent that trade secret protection alone would not provide adequate intellectual property protection for programs. It was not realistic to think one could maintain meaningful secrecy or negotiate meaningful licensing restrictions for millions of copies of software products with millions of customers. Indeed, some would question whether trade secrets could be said to exist in millions of copies of a commodity distributed to customers over the counter, even if with a "shrink wrap license." (Imagine if an automobile manufacturer made its cars available on a "license only" basis, and the car came with a proprietary legend on the hood saying that the manufacturer claimed valuable trade secrets in the engine which you would be considered to have misappropriated if you opened the hood and looked at or tinkered with the engine.) Congress' Commission on New Technological Uses of Copyrighted Works (CONTU) studied the software protection issue in the late 1970s and recommended use of copyright to provide intellectual property protection for computer programs, noting that copyright was inherently much better suited than trade secret law for the protection of mass-marketed software products. In 1980, Congress passed an amendment to the copyright law to make explicit that copyright protection was available to computer programs.

Neither the CONTU Report nor the legislative history of the 1980 amendments directly addresses the issue of whether it would infringe a copyright to copy an interface in order to make compatible software. Nor does either address the issue of whether it would be an infringement or a noninfringing fair use of copyright to make a copy of a program in order to reverse engineer it to find out how another program might need to be structured to communicate effectively with a computer system or other software. There is some indication that the Commission thought that copyright presented no threat to competitive software development because in CONTU's view there would almost always be another way to write code (besides using the expression adopted by a first program author) to perform specific functions. In those rare instances when there might be only a very limited number of ways to express some function, CONTU claimed that copyright would provide no protection because "idea" and "expression" would have "merged" and copyright would not protect the expression in order not to protect the idea.

Also left unclear was the extent to which trade secrets and copyright would coexist for software and the extent to which licensing agreements purporting to deprive those who lawfully acquired software of rights they would otherwise have in the software under copyright law would be enforceable. CONTU seemed to assume that because copyright was more suited to mass marketing, reliance on trade secrecy would subside. CONTU also thought that licensees of software should have the same rights to make copies of it in order to use it in the computer, make back-up copies, and make certain modifications to the software that owners of copies of software had. Congress, however, seems to have decided to limit the availability of these rights in software only to owners of copies, and not to licensees.

Enterprising lawyers have developed theories which would make it an infringement of copyright to make a copy of a program for reverse engineering purposes, and a theft of trade secrets to decipher the contents of a program.

What has happened in the past decade is that software firms have been claiming both copyright and trade secret rights in software and making the software available only in machine-executable form—and then only on a licensing basis, restricting even the rights to make copies to use the software, to make back-up copies, and to make modifications. Such licenses also often attempt to prohibit making copies of the software for reverse engineering purposes. Enterprising lawyers have developed theories which would make it an infringement of copyright to make a copy of a program for reverse engineering purposes, and a theft of trade secrets to decipher the contents of a program. Although it is generally lawful to reverse engineer a product to discern the trade secrets it contains, the theory went that the copying of the program to reverse engineer it made this kind of reverse engineering into the use of improper means (namely, copyright infringement) to acquire the trade secrets. In addition, of course, one who reverse engineered the software would also have breached the licensing contract. And so not just one but three legal violations might arise at the same time. The fact that it might have been necessary to try to reverse engineer the software in order to find out how its interface was structured so that one could develop a compatible system would, in this proprotectionistic view, be irrelevant.

The first few software copyright cases in which compatibility defenses did not sympathetically receive these defenses. Two firms who offered Apple-compatible computer raised compatibility defenses when Apple Computer sued them for copyright infringement based on their copying of Apple's operating system programs. In both cases,the defendants' programs were virtually identical, byte-by-byte, to the Apple programs. Neither defendant seemed to have made a genuine effort to independently develop compatible operating system software. (Given that there is still no Apple-compatible computer on the market, despite significant market incentives to develop such computers, perhaps these defendants in the early Apple clone cases were right that the idea and expression in the Apple operating system programs had merged such that the only way to make an Apple-compatible machine was to copy Apple's operating system.) Not surprisingly, the courts found copyright infringement in both cases. In one case, a federal appellate court cast doubt on whether a compatibility defense might ever be available in a software case. In another case decided at about the same time as the Apple-compatible cases, a judge decided that making a copy of a program in order to try to reverse engineer it infringed the copyright.

The NEC *v.* Intel decision, which came down in February 1989, was the first software copyright case to give more receptive treatment to a compatibility defense. NEC had hired a team to reverse engineer the microcode programs found on Intel microprocessor chips to study the functions the microcode performed and to identify the functional specifications needed to make compatible chips for NEC's customers. This team provided another team of NEC programmers with the functional specifications for the microcode, and the second team of programmers wrote a new set of microcode programs for NEC's chips. The court found no copyright infringement because the similarities in the NEC and Intel microcodes were due to their functionalities and the need to make a compatible chip.

A somewhat different outcome resulted from the arbitration of another major dispute between an American software developer and a Japanese firm that had reverse engineered the American firm's software in order to discern interface information to develop compatible software. The IBM–Fujitsu dispute was resolved by an arbitration order in November 1988. As in the NEC/Intel litigation, reverse engineering of programs, the appropriation of interface information, and use of that information to develop programs that were similar on account of their interfaces were central to the IBM/Fujitsu dispute. As in the NEC/Intel case, the defense argued that non-infringement should be found because of the need to develop compatible systems, for which interface information was critical. The arbitration order which resolved this dispute left IBM better off than Intel, however. The arbitrators relied on the principle of payments by Fujitsu in exchange for access to IBM programming material, but even then only in a secure facility and only for the purpose of extracting interface information. This information could, however, be used by Fujitsu in developing compatible systems.

The first court decision to rule directly on claims of trade secret and copyright protection in a software interface protocol involved a facsimile machine's "handshake" protocol for transmission of classified data. In Secure Services Technology, Inc. (SST) *v.* Time and Space Processing, Inc. (TSP), decided by a federal judge in Virginia in September 1989, the court ruled that the digital interface protocol was not a trade secret because SST had sold fax machines with the protocol to the U.S. government without having put proprietary legends on them. The court also ruled that the interface protocol was not protectable by copyright because there was insufficient original authorship in the protocol, the expression being constrained by functional considerations related to interoperability of the machines. Thus, TSP's competitive facsimile machine utilizing the same interface protocol was not infringing of the SST copyright.

As you can see, if the judge in the Atari/Nintendo dispute followed the rationale of NEC *v.* Intel or Secure Services Technology case, he or she might well throw out the copyright infringement and trade secret misappropriation claims against Atari. However, if the judge followed the reasoning of the Apple-compatible cases or the IBM–Fujitsu arbitration, the judge might uphold Nintendo's copyright and trade secret claims.

Because the Atari *v.* Nintendo dispute involves a patent on a computer interface technology, and Nintendo's power to block Atari's sale of Nintendo-compatible games turns substantially on the validity of the

While patents on software interfaces are not that common, they may become so in the future, especially if firms are unable to use trade secrets or copyright to protect interfaces in widely marketed software products.

patent, some attention should be paid to claims of patent rights in interfaces. The question about the validity of the Nintendo interface patent does not arise because interfaces are not the sort of thing that can be patented, but only because a prior patent is argued to establish that Nintendo's interface was already in the state of the art, and therefore not novel as the statute requires. In general, the ability of a firm to patent an interface of this sort (or nowadays perhaps even a purely software interface) depends on meeting the standards for patents set forth in the statute. If an interface is novel, useful, and nonobvious, is patentable subject matter, and is claimed in a proper manner in a patent application, a patent can issue on it, and no one else can make, use, or sell it without the patentee's permission during the life of the patent.

The antitrust laws were created to remedy monopolistic conduct. Because of some recent amendments to the patent statute, firms may find it harder nowadays to establish that a patentee is misusing its patent rights by acting anticompetitively when they either refuse to license their patents or tie the sale or licensing of the patented product on the sale or license of another product (e.g., buying only Nintendo-approved games for use in the Nintendo machine whose interface is patented). This means that it will be more difficult now for small firms to assert patent misuse when patent rights in interfaces are asserted to block the development of compatible software. Only if a firm has substantial market power and this conduct is part of a pattern of anticompetitive conduct will an antitrust lawsuit have a chance of success.

While patents on software interfaces are not that common, they may become so in the future, especially if firms are unable to use trade secrets or copyright to protect interfaces in widely marketed software products. While there are some remaining questions about the extent to which computer program-related innovations are patentable, there is growing acceptance and use of patents for software innovations, and there does not seem to be any inherent reason why, if software innovations are patentable, interfaces cannot be.

One other effort to address the ownership of interfaces issue was ADAPSO's recently issued proposed guidelines on claims of proprietary rights in interfaces and languages, which advises firms to give clear notice if they intend to claim proprietary rights in interfaces or languages. While it is true that from a business standpoint, it is important to know up front whether an interface is proprietary or not, this set of guidelines from a legal stand point merely begs the question of whether under copyright or some other form of intellectual property law, interfaces or languages are or should be private property.

All of this is a long way of saying what is basically a short message: Intellectual property rights in interfaces are a highly controversial and unsettled area of the law. Because small firms are so dependent on the ability to use interface information to make compatible software products, this is an issue of particular importance to this segment of the software industry. Having experienced such unprecedented growth and innovation in the software industry, especially from small startup companies in environments where the practice was (at least as to disclosed interfaces) to allow others to use the interfaces, we should be very careful before moving toward a system in which competition in the development of compatible software products was limited or eliminated by allowing interfaces to be owned by one firm. Intellectual property law does contain some limiting principles that will, I believe, make it difficult for firms in the computing industries to rely on trade secret or copyright law for protection of interfaces. Patent law, however, seems to present more opportunities for protecting interfaces from competitive software developers. Nintendo certainly hopes so.

Pamela Samuelson is visiting professor of law at Emory University's School of Law in Atlanta.