



[an error occurred while processing this directive]

Internet and Computing

Licensees Should Clarify Software Revision Rights

Parties should negotiate use, ownership and exploitation rights in program alterations.

BY BARRY NEMMERS

SPECIAL TO THE NATIONAL LAW JOURNAL

The National Law Journal (p. C24)
Monday, October 28, 1996

ONE OF THE challenges of software licensing is that new technology often requires a modification of familiar license provisions. These changes often will be subtle. Some technology advances are well-publicized and offer the luxury of advance preparation.

Most new technology enters the marketplace without fanfare, and lawyers learn of them only during a license negotiation. As a result, many agreements contain provisions based on outdated assumptions about the technology in the licensed software.

The effect of two emerging developments on one traditional license provision illustrates the point. A proper software license for large-business applications anticipates issues of ownership and use of software written by the licensee to adapt the licensed software to its own particular business processes. These license provisions will need to be reviewed in agreements covering "object-oriented software" and in contracts for the identification and correction of so-called year 2000 problems in existing software.

With the rapid acceptance of client/server architecture and distributed environments, large and medium-size corporations have been replacing collections of aging software with large, integrated applications covering business processes such as finance, logistics and human resources. Often groupware, such as Lotus Notes, is implemented with the new software.

Invariably, soon after the company goes live with the new system, in-house information systems, or IS, staff will discover the need for new functionality or for changes to the existing functionality to better serve corporate processes. These changes initially are left to the consultants who were hired to implement the new system, but, as the IS people become familiar with the software, they begin to make these changes themselves.

Obtain Broad Rights

Many software license agreements have a provision severely restricting the licensee's right to make such changes. These provisions may appear simply as prohibitions on the creation of derivative works or as notice that any modifications will invalidate the warranty or terminate the maintenance service agreement. The licensee, however, usually can obtain much broader rights to make changes in the licensed software and even the right to market those changes.

The reality is that almost every licensee of large business applications software is going to

write, or have someone else write, code to tailor the licensed software to the company's particular requirements or to fix perceived defects or insufficiencies. The license agreement should meet this fact with terms that are equally realistic.

The licensee's code can have significant potential value to other licensees. It may make material advances in the existing functionality, which end users in any industry would find attractive. It may have advanced functionality of value only to members of a particular industry or line of business. It may have negative value in the sense that the code sequences would reveal valuable intelligence about the company's operations to a competitor.

Once the broader value of the sequence has been understood, either the licensee or licensor or both may want exclusive ownership or use and exploitation rights. Alternatively, one of them may want to limit the use rights of the other party. If the license agreement has not anticipated these possibilities, difficult negotiations are likely to follow.

The licensor has additional interests in maintaining control or obtaining ownership over licensee-written code. Licensee code that modifies the licensor's source code may have unintended, negative effects elsewhere in the program, particularly if the software is highly integrated, multifunctional business software. These cause-and-effect connections in the licensor's code may not be immediately apparent and the licensor may spend a great deal of time trying to correct an unexplained defect before discovering cause.

Ownership Provisions

The licensor also may seek a contract term that gives it ownership of all software written by the licensee with any assistance from the licensor, even if the licensor's participation is minimal. This term usually is included to shield the licensor from a claim of copyright infringement or trade secret misappropriation when the licensee's routine later appears in a subsequent release of the licensor's software.

If this is the licensor's motivation, it is likely to be willing to grant the licensee additional use rights in exchange for this ownership. This provision also can become an issue with consultants hired by the licensee to perform system implementation or post-installation service.

When both licensee and licensor want ownership or extended use rights, a solution usually can be found in a combination of such provisions as grants of exclusive use rights for a limited period of time, a commitment to delay marketing and a right of first refusal to license for marketing. The primary concern should be to cover, however generally, all authorship contingencies that may arise.

The licensor needs to keep in mind that the licensee has a statutory right to "make an adaptation" of its copy of the software solely as an essential step in the use of the program,¹ and it is unlikely that a contractual prohibition on such modifications would be enforceable. Absent contractual provisions to the contrary, such creations would be owned by the licensee if the work were done by its employees or by a nonemployee under a work-for-hire contract.² Routines created by the licensee that are not dependent on the licensed software and that run as a freestanding program should not usually be of relevance to the license.

It is essential that license definitions adequately distinguish between the types of anticipated changes in the licensed software. Such defined terms are not standard; often, all changes are covered by only one definition. One flexible approach is to distinguish between licensee code that changes the licensed software and licensee code that contributes functionality only through an interface or by connections to "hooks" placed in the software for this purpose by the software developer.

Thus, a "modification" would be new code that changes the source code of the licensed software. An "extension" does not change the source code but is dependent on the licensed software and uses existing interfaces or hooks into the licensed software.³

Soon, probably within the next year, important new business application software written using "object oriented" methodology instead of the traditional top-down, literal methodology will be on the market. Object-oriented software will present new contexts for determining ownership and exploitation rights of modifications and extensions of licensed software. Existing assumptions about ownership and use rights within corporate IS departments and about existing license clauses are likely to prove insufficient or even misleading to accomplish their intended purposes.

Object-oriented software technology will complicate the already difficult process of identifying ownership of software code. To date, software has been written in a manner not unlike a literary work. The objective is identified, and an outline is developed followed by flow charts or data models. The code is then written in modules, not unlike book chapters--not necessarily in order, but necessarily integrated. A change in one module is likely to require changes elsewhere in the program.⁴

In object-oriented programming, however, code is written in small, discrete, unconnected units, each of which performs an identified function. Each object is self-contained, or encapsulated, and a user of an object will not know how the object was written to achieve that functionality, nor will the user care. The object is used by sending it a message to perform its function.

The programmer only needs to know what function the object performs and what message is required to initiate that performance. The same object can be reused in any context in that program. A truly distributed object can be used in other programs as well, to the point at which an object could be as generic as a spark plug in internal combustion engines. By modifying the message, the programmer can add properties to or alter the behavior of the object, in effect creating a subclass to the object's class.⁵

Object-oriented software will not yield readily to a traditional ownership or infringement analysis because it will not have source code printouts that can be read as one would read a book. It will require an analysis that focuses as much on the code's behavior as on its literary properties.

Determinations of ownership and probably use rights as well will not necessarily follow from whether the licensed software's source code has been changed or whether the licensee's new code stands alone. The licensee's code may run by sending modified messages to objects in the licensed software that cause the licensed software to perform in a different manner but without changing it. Yet the behavioral effect would be the same as if the source code had been modified by traditional programming.

Of course the licensee should know whether it has the rights to "reuse" the object in the licensor's code. Because there is so little object-oriented code with an installed base yet, it is not meaningful to speculate about the nature of any interfaces that object-oriented software developers may provide.

Licensees, together with counsel, however, should begin investigating these questions with their IS specialists as soon as the IS department begins to consider acquiring object-oriented applications. Counsel will want to understand the basics of object technology before drafting the necessary license language.

Millennium Conversions

A similar situation is likely to arise for some companies that develop software functionality to locate and correct two-digit date fields in their legacy software. To the extent this software was not custom-written for the company, procedures and software developed to search for and identify fields that are unable to distinguish between the year 2000 and the year 1900 may be of considerable value to other companies with a similar legacy code which have not yet found a solution.⁶ As the end of 1999 approaches, the value of such code will escalate.

Companies that hire consultants for this purpose may be tempted to recoup these high costs by obtaining the marketing rights to procedures or software developed at company cost. This will only be an option for companies that are among the first to find solutions to their year 2000 difficulties.

A consultant's solution to one company's millennium conversion problems may be marketable to other companies that are using the same software. But if the other companies are competitors, the first company may prefer to keep the successful procedure or software off the market.

Because the cost of correcting two-digit date fields is likely to skyrocket as the turn of the century approaches, even a year's delay may give the first company significant competitive advantage. The consultant may be willing to gamble the potential for increased fees against a possible loss of the opportunity, particularly if its client shares some of the risk.

If the first company did not cover the issues of ownership and exploitation rights to the recovery software at the time the consultant was hired, it is unlikely to have the leverage to obtain those rights after the consultant has achieved the solution. When the consultant is hired, the IS department and counsel are likely to be exclusively focused on avoiding a corporate year 2000 nightmare. At the very least, however, it should seek some price concession from the consultant for allowing the consultant to retain marketing and use rights.⁷

Form agreements for technology must be constantly reviewed and updated. As the speed of technological change escalates, counsel will need to be alert for changes that may require modification of even boilerplate provisions.

(1) 17 U.S.C. 117.

(2) In the first instance, the author is the person who actually creates the work. *Community for Creative Non-Violence v. Reid*, 490 U.S. 730, 109 S. Ct. 2166, 2171 (1989).

(3) Other terms may include a "patch," which fills a missing portion of the licensed software, either to meet unanticipated functionality requirements or to fix a defect in the software, or a "workaround," which is similar to a patch but implies the existence of a defect or shortcoming in the licensed software. Patches and workarounds are likely to require changes in the source code.

(4) See, e.g., Melville B. Nimmer & David Nimmer, *Nimmer on Copyright*, Sec. 13.03(F)(1) at 13-121 to -122 (1996).

(5) Orfali, Harkey, and Edwards, *The Essential Distributed Objects Survival Guide*, at 24-28 (1996); Barkan, David M., "Software Litigation in the Year 2000: The Effect of Object-Oriented Design Methodologies on Traditional Software Jurisprudence," 7 *High Technology Law Journal* 315, 320-28 (1991).

(6) It would be appropriate to review the original license to be sure there is no prohibition against any modification of the licensed software's source code. While the prohibition may not be enforceable, the licensor may attempt to use such a provision to obtain the modification project for itself.

(7) It is worth remembering that year 2000 programs may convey confidential information about the company's business operations simply by the way they are set up, for example. The confidentiality provisions of such consulting contracts should be drafted with particular care.□

Mr. Nemmers is a partner at New York's Chadbourne & Parke L.L.P., where he specializes in intellectual property and customs law.