ESSAY

AN ESSAY ON THE CHALLENGES OF DRAFTING A UNIFORM LAW OF SOFTWARE CONTRACTING

by
Maureen A. O'Rourke[*]

This Essay, originally presented at Lewis & Clark Law School's 2006 Distinguished Intellectual Property Visitor lecture, discusses the challenges involved in developing a uniform law of software contracting. Technology and the law have developed since 1995, when the first efforts to codify such a law began. These earlier efforts were largely unsuccessful, and substantial uncertainty still exists in transactions involving software. In this Essay, Dean O'Rourke discusses the American Law Institute's Principles project that seeks to identify approaches courts could use in adjudicating disputes involving software agreements. The challenges of developing the Principles include the same theoretical, practical and political issues that destined other efforts to disappointment. By incorporating lessons learned from the earlier efforts, Dean O'Rourke hopes that the Principles project will prove more successful.

[*] Dean, Professor of Law, and Michaels Faculty Research Scholar, Boston University School of Law. Thanks to Interim Dean Lydia Loren and Professor Joseph Miller for their invitation to visit Lewis & Clark as the Distinguished Intellectual Property Visitor in March 2006 and to Elena Cappella, Professor Bob Hillman and Professor Lance Liebman for comments. The author is the Associate Reporter on the ALI's Principles of Software Contracting project. Professor Bob Hillman is the Reporter. The opinions expressed herein are the author's alone and do not represent the positions of either Professor Hillman or the ALI.

I. INTRODUCTION

In 1995, the National Conference of Commissioners on Uniform State Laws (NCCUSL) and the American Law Institute (ALI) agreed on the desirability of undertaking the drafting of a new Article 2B on Licensing of Information for addition to the Uniform Commercial Code (UCC).[1] By 1999, however, the two organizations had agreed that the subject was not sufficiently developed for codification within the UCC and the joint Article 2B project was ended.[2] Thereafter, NCCUSL, believing that a standalone enactment would be both preferable and achievable, continued work on the draft and eventually promulgated the Uniform Computer Information Transactions Act (UCITA).[3] To date, Maryland and Virginia have enacted UCITA, while some other states have adopted so-called "bomb shelter" legislation.[4] Such laws provide that courts in the enacting states may not enforce a contractual choice of law provision that selects a state in which UCITA is the governing law.[5]

With UCITA not gaining universal acceptance and software only increasing in economic importance, the ALI decided to begin a "Principles" project. In the ALI's framework, a Principles project does not set forth settled law like a Restatement project generally would. Instead, it seeks to state basic principles supporting the law in a particular field, identifying approaches that courts could use and incorporate into the common law.

Yet this Principles project inevitably faces many of the same challenges that destined other efforts to disappointment. These challenges span a number of substantive issues and are theoretical, practical, and political in nature. In this Essay, I discuss the difficulties of drafting Principles of software contracting and how the history of earlier efforts will continue to influence the shape of the project. I begin by situating the debate within the context of the history of both the software industry and the law. I then turn to the ALI project and discuss how that history can help provide guidance in particular areas in the hopes of completing a project that proves useful to courts and withstands the pace of technological change.

---

[1] *See* Charles Cheatham et al., *Report on the Uniform Computer Information Transactions Act (UCITA)*, 57 CONSUMER FIN. L. Q. REP. 37, 37 (2003) (noting that NCCUSL set up an Article 2B drafting committee in 1995 and that the committee first met in 1996). The UCC is a unique project because addition or modification to it requires the consent of both NCCUSL and the ALI. Thus, when the ALI decided it no longer wished to participate in the drafting of Article 2B, that proposal could not become a part of the UCC.

[2] Pratik A. Shah, *The Uniform Computer Information Transactions Act*, 15 BERKELEY TECH. L.J. 85, 88 (2000).

[3] Cheatham et al., *supra* note 1, at 37–38. (describing the promulgation of UCITA and also noting that its committee adopted a number of amendments to the Act in 2002).

[4] *Id.*; *see also* Jean Braucher, *Uniform Computer Information Transactions Act (UCITA): Objections from the Consumer Perspective*, CYBERSPACE LAW. Sept. 2000, at 2, 3. (noting Iowa's adoption of bomb shelter legislation).

[5] Braucher, *supra* note 4, at 3.

II. SOME HISTORY[6]

A. The Industry

The software industry developed relatively recently. In fact, the first computer manufacturers took some time to understand fully the value of software. Early manufacturers marketed large mainframe computers. They generally marketed software and hardware as one package, viewing software not so much as a source of profit on its own, but rather as a necessity that made the highly profitable hardware more salable. These manufacturers, including IBM, provided their customers with the operating system and tools to write their own applications. They also even encouraged customers to share their software solutions with each other.

In this early environment, transactions were few in number—if not necessarily in absolute terms, then certainly relative to the number of software transfers that occur in the modern networked world. Since a small number of businesses could afford to finance the purchase of a mainframe, manufacturers could rely on traditional contract law to safeguard their ownership interest in the software's source and object code. The model of contracting resembled that of classical contract law—an arms-length negotiated, signed agreement between informed parties of reasonably equal bargaining power. Usually, manufacturers would provide only object code to customers, safeguarding the source code as a trade secret.

Over time, companies emerged that marketed software as a product in its own right rather than bundled with the hardware. These companies could offer solutions to customers who could not afford to set up their own development departments to use the application writing tools provided by the hardware manufacturer. Eventually, in the late 1960s, IBM unbundled the software and hardware, charging separately for each. This effectively expanded the market for independent software vendors who could offer varied solutions, including those that competed with IBM's.

The introduction of the personal computer and other devices like game consoles opened up vast opportunities for software developers. These were enhanced over time as software powered more and more devices and the Internet emerged as a medium through which all types of digital information, including software, could be easily transferred. Software became a standardized, mass-marketed commodity, quite a departure from the early days when it was often customized to meet the needs of particular customers. The manner of contracting also changed. With the growth of a mass-market, the ability to negotiate in the classical manner necessarily dissipated. Mass-market software today is provided under standard forms that are not negotiated,

[6] *See generally* Maureen A. O'Rourke, *The Story of* Diamond v. Diehr*: Toward Patenting Software*, *in* INTELLECTUAL PROPERTY STORIES 194 (2006) and sources cited therein.

certainly not signed, and often great bargaining inequality exists between the software's provider and its customer.

B. The Law

As the industry itself developed, so too did the law. While contract and trade secrecy law might suffice in a world of few customers and negotiated agreements, this system was never ideal even in that context. Contract, with its privity limitations, would give the software developer rights only against its contracting party, not against the world. Trade secret law does not protect against another's independent development of the secret, and it also would not protect against a third party's unwittingly coming into possession of the secret. When production and distribution of software expanded, it became even more critical for software vendors to find protection that would augment what contract and trade secrecy law would offer.

Software vendors looked to intellectual property law, a natural source of protection of which trade secret law is a part. Initially, because of a longstanding (ostensibly judicially-created) bar against patenting algorithms and the Patent & Trademark Office's hostility towards patenting software, most software vendors did not believe software to be patentable subject matter. At the same time, they had doubts about its copyrightability as well. As a product that performed a function, it seemed unlike the creative, artistic works that one usually associates with copyright. Some worried particularly that copyright law would not protect object code, readable only by a machine and conveying nothing to a human audience, or operating systems which functioned primarily to direct the computer's operation. Additionally, at the time, the copyright law required a deposit of the work as a condition of protection. Software developers were leery about providing copies of the source code to the Copyright Office, fearing loss of their trade secrets in return for uncertain copyright protection. The Copyright Office began accepting registrations under its "Rule of Doubt": The registrant could deposit object code and receive a registration indicating that the Office could not determine whether copyrightable authorship existed.

By the mid-1980s, judicial decisions established the copyrightability of both source and object code and operating systems and applications.[7] The remaining question was whether copyright law would protect not just the literal code but also its structure. Originally, by analogizing to copyright law's protection of the plot of a novel, courts answered that question in the affirmative, granting broad protection to the organization of the program.[8]

[7] *See* Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240 (3d Cir. 1983) (copyrightability of operating systems); Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870 (3d Cir. 1982) (copyrightability of object code).

[8] *See* Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1234 (3d Cir. 1986).

However, by 1992 the Second Circuit reached essentially the opposite result in a decision that provided very little protection for a program's structure.[9]

At the same time, the legal debate over the permissibility and desirability of patenting software continued.[10] Finally, by 1998 it became clear that at least the Federal Circuit believes that patent protection can apply to aspects of software.[11]

Against this backdrop of legal uncertainty and evolution, vendors continued to rely on contract. Contract could provide insurance should a court interpret the extent of intellectual property protection to be less than what the vendor anticipated under its own reading of the law. Additionally, sometimes the vendor might consider the rights offered by intellectual property law inadequate. Contract is also quite useful in addressing important issues other than rights in the program itself: warranties, limitations on liability and choice of law in case of a dispute, to name a few. As noted above, as the industry became characterized by mass-marketed software distributed by independent software vendors as well as some hardware manufacturers, vendors used a contracting model dramatically different from that employed at the industry's origins and certainly dramatically different from that envisioned by classical contract law.

Specifically, vendors used the shrinkwrap. We are all now quite familiar with this method of contracting, whether in its original form or in its high-tech incarnations as a clickwrap or browsewrap. Although standard form contracts are ubiquitous in many areas, the combination of the standard form, mass market, and subject matter protected by federal intellectual property law gave legal commentators pause. Additionally, the need for debate about the appropriate "default" terms for such matters as warranties and limitation of remedies seemed apparent. The time had thus come for some sort of legal effort to rationalize the law applicable to software contracting.

III. FROM UCC 2B TO UCITA

One question that emerged early in lawmaking efforts regarding software contracts was how to integrate emerging precedent in the software context, as well as rules that commentators regarded as desirable, into existing law. The shorthand for this discussion became the "hub and spoke debate."[12] Options included: (i) taking all of the provisions common to sales of goods, leases of personal property and licenses of information and putting them into a "hub" with the "spokes" consisting of rules unique to each context; or (ii) a separate

[9] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706–11 (2d Cir. 1992) (setting forth an abstraction-filtration-comparison analysis and test for non-literal infringement of computer programs).

[10] *See generally* O'Rourke, *supra* note 6, at 203–18 (discussing the evolution of patent protection for software).

[11] *See* State St. Bank & Trust Co. v. Signature Fin. Group, Inc., 149 F.3d 1368, 1373 (Fed. Cir. 1998).

[12] *See* Amelia H. Boss, *Taking UCITA on the Road: What Lessons Have We Learned?*, 7 ROGER WILLIAMS U. L. REV. 167, 172–73 (2001) (describing the hub and spoke debate).

UCC article on licensing of information that, while addressing many issues similar to those dealt with in Article 2 on the sale of goods, would nevertheless stand on its own.

Eventually, the second option prevailed and a drafting committee began work on UCC Article 2B. The UCC is a unique cooperative effort between NCCUSL and the ALI, and changes to it require the consent of both organizations. Thus, for Article 2B to succeed, it had to attract support from both organizations.

Work on UCC Article 2B continued for a number of years. Eventually, however, NCCUSL and the ALI essentially agreed that the initial decision to incorporate rules on transactions in computer information into the UCC was misguided. They stated simply that it had "become apparent that th[e] area [of computer information transactions] does not presently allow the sort of codification that is represented by the Uniform Commercial Code."[13] NCCUSL and the ALI thus parted ways and NCCUSL continued its work under the new rubric of UCITA.

Matters, of course, were somewhat more complex than the organizations' politic public statement. In fact, the Article 2B project had become enormously controversial, attracting the wrath of groups as diverse as the movie industry and consumer protection groups. Commentators strongly disagreed about such basic matters as scope and permissible means of contract formation, as well as about what rules should be adopted in new areas like electronic self-help.

NCCUSL worked for a time on UCITA, making a number of changes in response to the comments and criticisms of a variety of groups. It eventually promulgated UCITA, and, as noted above, Maryland and Virginia adopted the Act while other states enacted legislation preventing courts in their jurisdictions from using UCITA as the rule of decision.

IV. THE PRINCIPLES OF SOFTWARE CONTRACTING

Against this backdrop, the ALI began a Principles project in the area of software contracting. The lessons of the two prior efforts—UCC Article 2B and UCITA—necessarily inform the Principles' drafting. In this section, I discuss how the earlier projects highlighted just some of the theoretical, practical, and political problems that any effort to state rules on software contracting faces. I also discuss the approach to certain issues that we have taken in the Principles as currently drafted and how history informs our perspective.

A. Scope

The initial question that all three efforts necessarily faced was how to define to what the draft law would apply. This is a difficult theoretical issue. It

[13] Press Release, Nat'l Conference of Comm'rs of Unif. State Laws Inst., NCCUSL to Promulgate Freestanding Uniform Computer Information Transactions Act: ALI and NCCUSL Announce that Legal Rules for Computer Information Will Not Be Part of UCC (April 7, 1999), http://www.law.upenn.edu/bll/ulc/ucita/2brel.htm.

requires analyzing whether there is some set of subject matter that raises issues not adequately addressed by current law. Both Article 2B and UCITA had a relatively large scope—Article 2B began by applying to transactions in information but the drafters later narrowed it to apply to computer information transactions, and UCITA adopted the same scope. Computer information transactions were defined as, essentially, transactions involving information in electronic form which is obtained from or through the use of a computer or which is in a form capable of being processed by a computer.[14]

As technology evolved, however, it became clear that this definition could encompass all digital information, whether its eventual embodiment was as text, software, music, a motion picture, or something else. The music and motion picture industries particularly had devised a complex web of contracts based on the Copyright Act's statutory scheme. The major players in these industries had come to expect and rely on these contracts. Thus, they saw no need for a uniform enactment that could upset settled expectations. Although UCITA eventually excluded motion pictures and musical works from its scope,[15] the evolution of its scope provisions highlighted at least two issues: (i) any enactment must consider how advances in technology may affect its scope; and (ii) scope cannot be divorced from theory—there must be some unifying principle behind the subject matter that makes inclusion of it within a uniform law sensible.

The strategy chosen in the Principles is to identify the transactions giving rise to disputes and litigation because they do not fit well within existing law and to address them in a technology-neutral way.[16] Software is rather unique in its blend of the expressive and the utilitarian, and this dual status has raised questions in intellectual property law for some time. Its production requires a great deal of investment, but the end product can be easily copied in the absence of technological protection measures. Unlike music that is also distributed digitally, software has not been the subject of detailed subject matter specific legislation. Software also is often part of a networked architecture, unlike music or motion pictures.

For a blend of theoretical, practical, and political reasons then, the project limits its scope to the exchange of software for consideration. Theoretically, software occupies a unique niche that warrants separate legal treatment, or at least clarification. Practically and politically, by narrowing the scope of the Principles vis-à-vis UCITA, the project avoids the necessity of creating industry-specific carve-outs that add complexity to the effort. Yet, the Principles leave open the possibility for a court to apply them by analogy to matters outside their scope. For example, although digital databases are often

[14] UCC § 2B-102(a)(8)–(9) (1999), *available at* http://www.law.upenn.edu/bll/ulc/ucc2b/2b299.htm; UCITA § 102(a)(10)–(11) (2002), *available at* http://www.law.upenn.edu/bll/ulc/ucita/2002final.htm.

[15] UCITA § 103(d) (2002), *available at* http://www.law.upenn.edu/bll/ulc/ucita/2002final.htm.

[16] *See* PRINCIPLES OF THE LAW OF SOFTWARE CONTRACTS 12–18 (ALI, Preliminary Draft No. 2, 2005) for a discussion of Scope. The text here is derived from that section.

marketed through the same distribution channels and contracting methods as software, they are excluded from the Principles—they share only some of the characteristics of software that justify its separate treatment. However, a court might opt to apply the Principles by analogy to questions common to both subject matter.

Even with this scope that is narrower than UCITA, the Principles are likely to face one issue with significant political overtones—the treatment of open source software. Open source providers offer their code under a variety of terms, often requiring the recipient to agree to distribute derivative software under the same terms as the initial transfer. This condition is more than what is necessary to constitute a gift and thus is consideration under contract law. Open source advocates generally bridle at labeling their terms a "contract." They prefer to style them instead as copyright permissions, perhaps to avoid questions of enforceability. Open source is included under the Principles so long as the transfer is one for consideration as that term is defined under traditional contract law. The challenge going forward will be to develop exceptions to general rules, where appropriate, for open source and also to attract the support of open source providers.

B. Substantive Rules: Some Examples

In the area of substance, one question the Principles must answer harkens back to the hub and spoke debate. What approach should the Principles take in areas in which existing law seems sufficient? For example, courts have a great deal of experience with unconscionability, and changing its language might introduce unintended consequences. Yet unconscionability may have an important role to play in policing over-reaching terms in software agreements. In such cases, we have opted to repeat existing law and to try to provide guidance in the comments with reference to practices particular to software. For example, some standard form contracts applicable to generally available software prohibit the practice of benchmarking. Such a term may be unenforceable because unconscionable. This approach, along with the relatively narrow scope, makes the Principles less complex than UCITA and avoids tinkering with settled law.

In some cases, however, "settled" law might benefit from some clarification or change. For example, while mass marketing using standard forms is not unique to software, it may prove useful to state rules in that context that courts could opt to use for subject matter other than software.

We take this approach in some areas, including, for example, choice of law. Generally, the law in that area is settled, with courts using the standards of the Restatement (Second) of Conflict of Laws or Article 1-105 of the UCC. There is now, though, some risk of disharmony in UCC transactions. The most recent version of Article 1 departs from the former "reasonable relationship" test and grants the parties broad freedom to choose the governing law. To date, however, states adopting Revised Article 1 have retained the traditional test.

In any event, the usual tests do not distinguish between standard form and negotiated contracts. The choice of law clauses in the former have been the

subject of frequent litigation in software contracts. Thus, we chose to draft a section on choice of law to set rules for standard form transfers of generally available software.[17] We adopt the reasonable relationship test with which courts are familiar, but limit application of the chosen law where it would be manifestly contrary to public policy expressed in the law of the jurisdiction that would otherwise govern in the absence of a choice of law provision. The notion is to acknowledge the special problems raised by standard form contracts and to provide protection to both consumers and businesses who buy software under a standard form.

There is nothing about the issue, however, that is unique to software. Thus, the choice of law provision might usefully be adapted to the standard form contract setting generally, should a court choose to do so.

Finally, in some cases, there is no settled law to clarify or modify. Two cases in point are federal preemption and electronic self-help.

Although there is, in fact, a fair amount of copyright and patent preemption law, it is quite confusing, and makes it difficult to extract consistent principles. Software agreements bring the preemption issue to the fore because they, perhaps more than other contracts, routinely contain provisions that seek to broaden intellectual property rights, either by granting rights the relevant statute does not provide or by shrinking the limitations on rights that are provided by the statute. Adding to the complexity, software can be both patented and copyrighted, making two bodies of preemption law relevant.

Indeed, Article 2B and UCITA both had great difficulty addressing the intersection between state contract law and federal intellectual property law. In fact, that relationship became one of the most contentious issues in drafting UCITA. Eventually, the drafters settled on two relevant provisions: (i) Section 105 (a) which provides, "A provision of this [Act] which is preempted by federal law is unenforceable to the extent of the preemption"; and (ii) Section 118 which permits reverse engineering for certain limited purposes despite a contractual provision to the contrary.[18]

Section 105, of course, states the obvious and adds nothing to the law already applicable. Section 118 adopts the view of some copyright cases that had permitted reverse engineering to uncover unprotected elements of the software.

We hope to provide more guidance than Section 105 but stop short of a blanket right to ignore a ban against reverse engineering in a software agreement. This approach runs the risk of alienating the interest groups that lobbied for the reverse engineering exception but has the virtues of consistency with current law and providing flexibility to courts. Our preemption section has

[17] We define "standard-form transfer of generally available software" as a transfer of "(1) small quantities of software to an end user; or (2) the right to access software to a small number of end users; if the software is generally available to the public under substantially the same standard terms." PRINCIPLES OF THE LAW OF SOFTWARE CONTRACTS 12 (ALI, Preliminary Draft No. 3, 2006).

[18] UCITA §§ 105, 118 (2002), *available at* http://www.law.upenn.edu/bll/ulc/ucita/2002final.htm.

focused on stating the law at a high level and providing expansive comments to provide courts with factors to evaluate in deciding whether or not to enforce a particular contractual provision.

The issue of electronic self-help differs from that of preemption because there is little law on the subject. Like preemption, however, it was extremely contentious in the UCITA discussions. The 1999 draft of UCITA permitted electronic self-help under limited circumstances and provided safeguards for the party against whom the self-help would be exercised. Many continued vehemently to oppose electronic self-help under any conditions and the 2002 version of UCITA prohibits it.

The Principles attempt to strike a compromise, permitting electronic self-help in a non-standard form, non-consumer transfer, but providing more extensive protection than the 1999 draft of UCITA. This honors the principle of freedom of contract but also provides consumer protection, a key concern of those opposed to UCITA.

V. CONCLUSION

Technology and the law have developed since 1995 when the first efforts to codify a uniform law with respect to computer information transactions began in earnest. Nevertheless, substantial uncertainty still characterizes aspects of transactions involving software. The ALI's Principles project hopes to dispel some of that uncertainty.

The Principles project has learned a great deal from the Article 2B and UCITA experience. The history of those efforts has influenced the scope of the still-evolving project and the manner in which we address certain issues. This does not guarantee the project's ultimate success but it provides some hope that we may draft reasonable rules that courts may choose to adopt.