8-25-1995

System Development from a Legal Perspective

Janice C. Sipior
Villanova University

Burke T. Ward
Villanova University

William P. Wagner
Villanova University

System Development from a Legal Perspective

Janice C. Sipior
Management Department

Burke T. Ward
Business Law Department and The Graduate Tax Program of the Law School

William P. Wagner
Management Department
College of Commerce & Finance
Villanova University
Villanova, PA 19085

INTRODUCTION

This paper discusses the potential for legal liability when software malfunctions and causes financial loss or harm to the user. The focus is on the systems developer as opposed to the vendor, although the same legal issues may apply to vendors as well. Various theories in the U.S. legal system may form the basis for legal action, on the part of the purchaser or ultimate user, based upon the characteristics of the individual case. The legal theories examined, for their application to malfunctioning software, are breach of contract and the tort theories of fraud and computer malpractice.

LEGAL THEORIES APPLICABLE SHOULD SOFTWARE MALFUNCTION

The development of software for sale constitutes a business transaction, subject to various elements within the U.S. legal system. This section examines the legal theories applicable should software sold for use outside of the developer's organization malfunction. The following analysis addresses breach of contract and the tort theories of fraud and computer malpractice.

Breach of Contract

A threshold question in any legal action based on breach of contract is whether the subject matter of the contract, i.e., the software, is within the scope of Article 2 of the Uniform Commercial Code. If the contract is within the realm of Article 2, the customer or the systems developer may find it preferable, depending on the circumstances of the case.

For Article 2 to apply, the contract must relate to a transaction in goods (UCC Sec. 2-102). A critical question to determine the applicability of Article 2 to software cases is whether the contract is for goods or for services. Since software usually involves both physical product and services, the courts are not in agreement as to whether Article 2 applies. Computer hardware is clearly considered goods, and thus within the Article's scope (UCC Sec. 2-105). In mixed sales/service transactions, the trend is to find that the

sales aspect predominates and the service aspect is more incidental to the contract (Arlington v. Schindler , 1979). This is true even in cases where the sale and service aspects are split into different contracts. In the case of Neilson Business Equipment v. Monteleone (1987), the Court integrated three separate contracts, i.e., hardware, software, and maintenance, into one agreement and applied Article 2 to the entire agreement. Further, the trend to find the transaction within Article 2 has been applied in cases involving only software (RRX v. Lab-Con , 1987). Although software is a codified form of intangibles such as data, information, or knowledge (service), it is sold on a physical disk, and thus is arguably a good within Article 2. The analogy in support of this argument is to a professor's lectures (service) being published as a book (goods) (Cardozo v. True, 1986). Although there is less judicial agreement, Article 2 has been held to apply to software leases and licenses as a transaction in goods (Communications Groups v. Warner Communications, 1988).

Express Warranties

Under Article 2, an express warranty is an affirmation of fact or promise relating to the goods, any description of the goods, sample, or model that becomes a part of the basis of the bargain (UCC Sec. 2-313). The use of words such as warranty or guarantee by the developer are not required to form an express warranty. For example, a demonstration showing a purchaser's equipment working with a particular type of computer was held to be an express warranty as to communication capability (Cricket Ally v. Data Terminal, 1987). A problem for the purchaser, regarding such representations, is that an integration clause in the written contract may prevent prior oral representations from being construed as express warranties. For example, if a developer states that his software will require minimal training time for employees, but does not include that statement in the written contract, an express warranty may not arise. The effect of the integration clause, in this example, is to disclaim prior representations and warranties not contained in the written contract.

Implied Warranties

Implied warranties are created by operations of law and not by the express representations of the developer. For software, the applicable warranties include the implied warranty of merchantability (UCC Sec. 2-314) and the impled warranty of fitness for a particular purpose (UCC Sec. 2-315). The implied warranty of merchantability essentially means that the product is of fair or average quality. For the implied warranty of fitness for a particular purpose to arise, the developer must have known the particular purpose for which the software was needed, and that the buyer was relying on the developer's skill and judgment. If this warranty arises, the product must be fit for the purchaser's particular purpose. These two implied warranties may merge when applied to software. The purchaser of software is usually relying on the skill and judgment of the developer, who is aware of the user's requirements and needs. Thus, the software must not just be of average quality, but meet the user's specific purposes, of which the developer or vendor was aware (Hollingsworth v. The Software House, 1986).

Remedies and Limitations

If the contract for software is breached by the developer, Article 2 provides the purchaser a variety of contractual remedies. Of particular importance is whether the buyer is entitled to consequential damages. In addition to general compensatory damages, such as recovery of the purchase price, consequential damages consider economic losses resulting from the user's particular needs not having been met, of which the vendor knew or had reason to know at the time of the contract (UCC Sec. 2-715 (2) (a)). Consequential damages usually include lost profits, but may also include other losses and expenses, such as extra labor costs.

The potential open-ended liability of the developer is usually restricted by protective provisions in the developer's contract. It is common in the computer industry to disclaim implied warranties. Article 2 permits such disclaimers if certain technical requirements are met (UCC Sec. 2-316). Further, the user's available remedies, including consequential damages, may be restricted by limitations contained in the developer's contract (UCC Sec. 2-719). In the software industry, it is not uncommon to find the user's remedies limited to replacement with an upgraded version, or to have the amount of damages limited to the purchase price, thus protecting against substantial consequential damages. For example, if a $50,000 software application is defective and causes the user's business to fail, such a clause would limit the user's recovery to $50,000, not the full amount of the user's loss.

Article 2 affords some protection to the user from disclaimers and limitation of remedies/damages provisions. If these provisions are found to be either unconscionable or cause a limited remedy to fail of its essential purpose, then the restrictive clause may be ignored and all other remedies become available (UCC Sec. 2-302). Although unconscionability is not defined in Article 2, it generally deals with the prevention of oppression and unfair surprise, and not disturbance of the allocations of risks because of superior bargaining power (UCC Sec. 2-302 Comment 1). For software contracts which are commercial rather than consumer, unconscionability is not typically found since the parties are usually business persons dealing at arm's length (Consolidated Data Terminals v. Applied Digital Data Systems, 1983). Similarly, if a limitation clause will cause a limited remedy to fail of its essential purpose, i.e., no effective remedy, the limitation clause is disregarded (UCC Sec. 2-719). For example, if in a software contract, the user's remedy was limited to replacement with an upgraded version, and the software could not be debugged, then the clause fails of its essential purpose and is not enforced.

Tort Theories

While most litigation with respect to software defects is based on some form of breach of contract theory, tort based claims are on the rise. A tort is a private or civil wrong other than a breach of contract, for which a court may provide a remedy (Black, 1979). As previously discussed, most software contracts are prepared by the developers and contain possibly enforceable disclaimers, limitation of remedies, or limitation of damages provisions. Particularly vexatious to a user who allegedly suffers from financial loss or

harm are contractual provisions that prohibit the recovery of consequential damages. Tort based legal action attempts to circumvent these disclaimers and limitations because the user's recovery is based on a specific tort, and not on the contract. While there are numerous possible tort theories, fraud and negligence (computer malpractice) appear to be the most common. It is possible that malfunctioning software may someday be subject to the theory of strict liability in tort, which imposes liability based on product defect. This is not yet the case (Miyaki, 1992).

Fraud

A cause of action for fraud generally requires that the plaintiff, user in software cases, prove (Analysts International v. Recycled Paper Products, 1987):

1. that the defendant, developer in software cases, made a statement,
2. of a material nature,
3. which was untrue or made in reckless ignorance of its truth or falsity,
4. was relied upon by the plaintiff to his detriment,
5. made for the purpose of inducing reliance, and
6. that the reliance led to the damages.

The usual claim is that the user was fraudulently induced to enter into the contract by the false representations of the developer, vendor, or consultant (RKB Enterprises v. Ernst & Young , 1992). A typical allegation of fraud is that the developer represented that the software could meet the user's needs when it knew the software could not, or failed to disclose previous significant problems with the software. If sustained, the allegation of fraud in the inducement is separate from the breach of contract claim, possibly allowing a broader or larger recovery for the user (Schleifer, 1986).

Negligence

The tort theory of negligence is applicable in cases within which four elements are present:

1. the existence of a legal duty of care owed to the injured party,
2. a breach of that duty,
3. actual and proximate cause of the injury by the breach of that duty, and
4. damages (Lyman, 1992).

Negligence applied to professionals is referred to as malpractice. Malpractice has been defined as "professional misconduct or unreasonable lack of skill... Failure of one rendering professional services to exercise that degree of skill and learning commonly applied under all the circumstances...with the result of injury, loss, or damage (Black, 1979). Usually in malpractice actions, there is a professional licensing system, code of ethics, and formal disciplinary system (Condo, 1991). These elements are found in such occupations as law, medicine, and public accountancy, but are not enforceable within the computer industry. The tort based theory of computer malpractice, that is, negligence

applied to computer professionals, is still in its embryonic stages. Without a clearly determinable professional standard, courts have generally not allowed a tort action for malpractice. A New York Court recently stated that, "... it should be noted that there is no cause of action for professional malpractice in the field of computer consulting. While computers are relatively new equipment, of a complex technical nature, critically important to business, we decline to create a new tort applicable to the computer industry" (RKB Enterprises v. Ernst & Young , 1992).

Despite the general disallowance of a computer malpractice cause of action, there is judicial authority supporting this theory. In Diversified Graphics, Ltd. v. Groves, the Eighth Circuit Court of Appeals, in affirming an $82,500 computer malpractice award, based its decision in part on the defendant's failure "to act reasonably in light of its superior knowledge and expertise in the area of computer systems." It should be noted that in its interpretation, the Court discusses that defendant, Ernst and Whinney, incorporated in its Guidelines to Practice the AICPA's Management Advisory Services Practice Standards. These general "standards" are for accounting firms and not specifically for the development of a computer software (in this specific case, an in-house turn-key system), and thus should not be used to establish a professional standard for computer systems services. This case is currently a distinct, but prominent, aberration.

CONCLUSION

System developers themselves have expressed greater optimism regarding their own abilities to avoid flaws in systems development (Neumann, 1993). Yet, even systems considered to be correctly programmed have had serious problems. Indeed, "there are no guaranteed assurances that a given system will behave properly all of the time, or even at some particularly critical time." (Neumann, 1993). Nonetheless, systems developers must continue to strive to achieve successful systems or be held accountable by the courts for financial loss or physical harm resulting from defective code.

REFERENCES

Available upon request.