Cem Kaner, Ph.D., J.D.
Law Office of Cem Kaner **kaner@kaner.com**
http://www.badsoftware.com **408-244-7000 (v)**
P.O. Box 580, Santa Clara, CA 95052 **408-244-2181 (f)**

"BAD SOFTWARE - WHO IS LIABLE?"

Cem Kaner, J.D., Ph.D.

Invited Address: American Society for Quality's Annual Quality Congress Philadelphia, May 1998

Keynote Address: Quality Assurance Institute Regional Conference, Seattle, June 1998.

Abstract

A new law will probably be introduced into state legislatures in the fall of 1998. It will govern all contracts for the development, sale, licensing, and support of computer software, and of most other information-related products. Customers' advocates, such as Ralph Nader, Consumers Union, and the Society for Information Management (which represents big customers) have been heavily critical of this proposed law. I've played a leading role on the customer side for the last two years. I believe that over the long term, this law will severely injure the competitive position of the American software industry by making it easier for software publishers to sell lower quality merchandise.

Introduction

What would happen if the government passed a new law that made it possible for your company to reduce the quality of its products without suffering a significant increase in external failure costs? Do you know any executives who would be tempted to cut your company's investment in quality? If this law applied to all of the products that your company makes today, or will make in the future, how much would these executives spend on efforts to improve your products' quality? Suppose this law applied to anyone selling products in the United States? Over the long term, what effect would this have on your (and the rest of your industry's) competitive position in markets outside the United States?

I believe that my industry (software) is facing this issue today, and I am deeply concerned about it. The proposed law is a huge (about 250-300 pages) amendment to the Uniform Commercial Code (UCC). The amendment is called Article 2B (Law of Licensing) and is loosely based on UCC Article 2 (Law of Sales), which governs sales of goods in 49 states plus the District of Columbia (Louisiana's laws are similar but not part of the UCC). Article 2B has been under development for about ten years and is scheduled for introduction in state legislatures this fall.

I'm a software quality advocate (see for example, Kaner, Falk & Nguyen, 1993, and many papers on my software testing Web page, www.kaner.com). I'm also an attorney who focuses on software law (see my consumer protection Web page, www.badsoftware.com). Over the past two years, I have spent hundreds of unpaid hours and tens of thousands of dollars trying to improve Article 2B. (For example, see Kaner, 1996c, 1996e, 1997a, 1997b, 1997c, 1997d, 1997f, 1997g, 1997h, 1997i, 1997j, 1997k, 1998; Kaner & Gomulkiewicz, 1997; Kaner & Lawrence, 1997; Kaner, Lawrence & Johnson, 1998; Kaner & Pels, 1997; Kaner & Paglia, 1997). This is a purely *pro bono* effort. I am not only not paid for it; I've lost a lot of business because of my work on 2B. (Many of my clients and former clients are software publishers, and some have not been delighted with me for working on this.) I'm working on this as a matter of principle.

I am drawing this proposed law to your attention today for three reasons:

- ♦ I am appealing to you, as fellow quality advocates, for help. ***I am not asking for money.*** I am asking for advice and possibly for help from a few volunteers. Additionally, I hope that ASQ will study the proposed changes to the Law of Sales (UCC Article 2) and the Law of Licensing (Article 2B) and advise the organizations drafting these changes of ASQ's concerns and point of view.
- ♦ For those of you who are software customers, I am alerting you that an industry-full of your suppliers are about to be put under new rules that will make it harder for you to control the incoming quality of their products.
- ♦ For those of you who are not concerned with software, I want you to understand that if Article 2B goes through, you can bet that Article 2 will be revised to conform to the same contracting principles. These changes will affect how we all do business.

Whenever I talk to software quality advocates about software liability, some people like what I have to say, and some people hate it. On the positive side, some people like the idea that lawsuits for defective products drive up external failure costs. This changes the balance in quality cost analyses and encourages companies to ship products that are less defective. On the some-people-hate-me side, we have reactions of people like W. Edwards Deming, one of my heroes (except for his ideas about lawyers). He named seven "deadly diseases." Number 7 was "Excessive costs of liability, swelled by lawyers that work on contingency fees." (Deming, 1986, p. 98).

There is no question that some lawsuits have been frivolous and that some industries have been harmed by excess litigation. Just as there is no question that some lawsuits have been well justified and that many people have been harmed by some clusters of defective products.

We should seek a balance of rights and responsibilities between buyers and sellers. As background for seeking that balance, this paper looks at customer dissatisfaction, quality costs, and risk analysis in the context of legislative development (creation of laws). With that context set, the paper will return to Article 2B.

You might end up disagreeing with my analysis of Article 2B. I'll feel that this paper has met its objectives if it helps you form an opinion about 2B (pro or con) with more sophistication about the goals of legislative development than you have today.

Customers Have Genuine Problems

Part of what Article 2B will do will be to make it dramatically harder for customers to sue software publishers than is the case today. Some people think this is terrific idea, because they believe the myth that most business-related lawsuits are the frivolous creation of evil plaintiff's lawyers. Supporting this has been a wave of statistics that appear to show that customer lawsuits against businesses have skyrocketed. Consider for example, the following data from the *1994 Annual Report of the Judicial Council of California* to the governor and legislature. I use this particular data set because it's been used in political campaigns to argue that there has been an explosion of frivolous lawsuits. (1983-84 is the first of the 10 years in this study. Superior Court filings usually involve cases of $25,000 or greater.) At first glance, this looks *like* a litigation explosion (see Exhibit 1).

Exhibit 1—Superior Court Civil Filings, 1983-1993

1983-84	1992-93	**increase**
561,916	684,070	**122,154 cases (22%)**

But when we look at Exhibit 2, we see that despite California's increase in population, there are fewer business, consumer protection, and personal injury suits, not more. (There are also dramatically fewer customer-side lawyers and consumer protection advocates today than there were ten or twenty years ago. It is astonishing to me how thinly spread today's consumer protection community is.)

The increase in civil court filings comes from "Other Civil Petitions" in which the Judicial Council includes "petitions filed under the Reciprocal Enforcement of Support Act" and various other family-law related petitions. The statistics *do* reflect an explosion of actions, but it's an explosion of actions (often filed by the district attorney's office) to enforce child support orders. This is not a business problem.

Exhibit 2—A Closer Look at the Filing Statistics

Other civil petitions (Child Support):		
1983-84	1992-93	**increase**
121,968	267,980	**146,012 (120%)**
Personal injury, death, property damage:		
1983-84	1992-93	***decrease***
96,731	88,346	***-8,385 (-9%)***
Other civil complaints (includes business litigation):		
1983-84	1992-93	***decrease***
111,802	107,377	***-4,425 (-4%)***

Many programs have serious bugs and unhappy customers, partially because it's impossible to fully test the software (Kaner, 1997g). If that was the main problem, we could substantially reduce it by adopting processes that reduce the probability of coding errors (Ferguson, Humphrey, Khajenoori, Macke, & Matuya, 1997; Humphrey, 1997). However, most (in my experience, all) software companies ship software with bugs that were found during testing but not fixed. Some of these problems have been extremely serious. In the mass market, which is what I know best, we've seen several recent class action suits by dissatisfied customers (for example, Leading Edge Computers, America OnLine, Compaq, and Corel).

There is significant customer dissatisfaction. Please see Kaner & Pels (1997) for *extensive* footnoting to sources for the next five paragraphs.

- ♦ Computers became a "real" consumer product in 1994, when they outsold television sets. By the end of 1995, computers and software ranked #8 in the top 10 list for complaints to the Better Business Bureau, outdoing used car dealers. The consumer market continues to grow. About 40% of American households have computers.
- ♦ In 1996, there were 200 million calls for technical support. At an average of about $23 per call, the industry spent about $4.6 billion on these calls. Over the past seven years, the ratio of support to total employees in hardware and software companies has grown from 1 in 12 to 1 in 6. The average amount of training for technical support staff before they are put on independent telephone answering duty is only 40 hours.
- ♦ In 1996, the industry left these callers on hold for about 3 billion minutes. The software industry has been one of the worst for leaving callers on hold. One study found that software companies leave callers on hold longer than any other industry studied, worse than government agencies, computer hardware companies, airlines, banks, utility companies, and others. Once you get off hold, you often reach a first-level person who can't answer your question. Wait some more for a "specialist." The Software Support Professionals Association estimates that the average time to reach the right support technician is 30 minutes for PC/Shrink-wrap products. (This person may not know the answer, but she is the right person to ask the question of.)
- ♦ ***Customer satisfaction with software companies' technical support has dropped steadily for 10 years.***
- ♦ Several software companies now charge customers $3 or more per minute for support. Some companies will waive the charge for a customer who calls about a legitimate (in the company's view) defect. Others will not. Customers get angry when they realize that they've been paying for support for a defect that the company chose not to fix when it shipped the product. For example, in a recent class action suit, a customer alleged that Compaq released a product with known software defects (*Johnson v. Compaq*, 1997. NOTE: I have not yet seen an answer from Compaq. Johnson's allegations sound very serious to me, but Compaq might have a strong and reasonable response. *Always take initial complaints with a big grain of salt*.). Apparently, Johnson spent $218 on support and his problems were never resolved. He posted complaints on the AOL/Compaq message board, Compaq allegedly removed the messages and complained to AOL that this customer was

abusing his account by posting inappropriate messages. The customer responded with a lawsuit. Several other lawsuits have involved a combination of bad software and support (Kaner, 1997e).

There are many other types of lawsuits involving defective software. For example, in the case of *General Motors Corp. v. Johnston* (1992) a PROM controlled the fuel injector in a pickup truck. The truck stalled because of a defect in the PROM and in the ensuing accident, Johnston's 7-year old grandchild was killed. The Alabama Supreme Court justified an award of $7.5 million in punitive damages against GM by noting that GM "saved approximately $42,000,000 by not having a recall or otherwise notifying its purchasers of the problem related to the PROM."

RISKS has carried reports of many serious errors, such as an airplane that rebooted its software mid-flight. There have been many, many lawsuits over custom software whose defects seriously disturbed the customer's business, as well as widely publicized cancellations of large programming projects, such as a project for California's Department of Motor Vehicles and another for the US Internal Revenue Service. A recent news report claimed that over 13% of Americans are being underpaid from private pension funds, largely because of software errors.

Draft Laws are Products Under Development

A law is a product, just like any other product. It goes through a design and development phase, is put in service, is repaired (perhaps badly) several times throughout its life, and is eventually taken out of service.

A draft law is a product that is in the process of being designed. Like other products, draft laws are subject to the sometimes-conflicting requirements of different stakeholders and to constraints that make it difficult or impossible for the system developers to meet all of the requirements.

Three Requirements for Commercial Laws

There are several other requirements for any commercial law, but these three stand out when I think about Article 2B.

- ***The primary requirement for a commercial law is that it should facilitate commerce.***

To facilitate commerce, you want uniformity and stability:

- ***Uniformity***: If laws are the same from state to state, it's easier for buyers and sellers to understand how to do business with each other. When state laws vary significantly, companies may delay signing a contract to develop a product, or delay releasing a new product or upgrade until they understand the legal ramifications involved. If laws vary across states, this analysis takes too much time and inflates legal expenses. Another desirable uniformity involves non-uniform treatment of goods and services. UCC Article 2 governs sales of goods (as do many consumer protection laws). Different laws (not the UCC and not the same laws in every state) govern sales of services (such as auditing, testing, and custom programming). Suppose company X sells an accounting program that is largely finished, but must be slightly customized for each customer. Is this a sale of goods or services or both? Which laws apply? Article 2B is designed to bring both types of uniformity (across states, and across the goods-vs.-services border) to the law of software-related goods and services.
- ***Stability***. When legal rules are stable, you can develop a thorough understanding of their effects. You can know how much significance to attach to an advertisement or a warranty. You know who has to buy the insurance. You know what risks you're taking in selling the product, and thus how to build an allowance for risks in the product's price. You know your rights, and the other party's rights, and you know the limits of your rights. When laws are ambiguous, buyers and sellers get into disputes because both sides interpret the law reasonably, but in a way that is favorable to their own interests. The result is surprise, anger, litigation, and lost profits—waste. Laws' rough edges smooth out over time, as courts resolve the ambiguities. However, if the law changes frequently, it never gets smoothed out and business models stay confused.

  Laws that tilt too much toward one side of a transaction (such as a buyer or seller or lender) do not promote stability. If one side is shabbily enough treated, it will bring political pressure to bear to get a fairer shake. Maneuvering for significant short term advantages might be successful over the short term, but the backlash is inevitable.

One Critical Constraint

You can look at laws as classification rules. For example, in criminal law, there are two main classifications: guilty and not guilty. Depending on a set of facts, an accused person is to be classified one way or the other. Similarly in civil law, a warranty applies or it doesn't. A seller is liable to the buyer or it isn't. An insurance policy covers this situation or it doesn't. Given a set of facts, the law tells us which classification is correct.

- ***Unfortunately, a decision rule that is less complex than the situation being classified will make mistakes.***

Laws are not written to cover every possible detail. The law can't be specific enough to cover all possible cases. Instead, laws are written as statements of legal policy. The jury (or judge) looks at a specific set of facts and decide whether these facts fit a pattern laid out in a law. The jury decides whether conduct was "unreasonable" or "outrageous." They jury decides how a normal customer would interpret a specific advertisement, and whether a product's failure to live up to an advertised promise would be "material" (significant) to a normal customer. The laws (statutes passed by legislatures, such as Article 2B, and published court decisions that contain judges' interpretations of these statutes that add needed details) provide detailed guidelines for the jury, but given the same set of facts, two juries will sometimes reach different conclusions.

As a result, sometimes buyers will lose when they should win. Sometimes sellers will lose when they should win. All sides (buyers, sellers, lenders, debtors, etc.) have had, and will continue to have, great stories of unfairness to print in the newspapers.

There is no way around this constraint. The world is a complex place, and its ground rules (e.g. what is technologically feasible) change daily. The law cannot mechanically cover all possible circumstances. Instead, it has to be a generalization, to be applied to specific circumstances by people who exercise their good judgment.

How Should We Bias the Law?

Given that the legal system will sometimes misclassify a situation, we can try to improve the precision of the law, but this will only take us so far. Eventually we have to face the constraint—sometimes the legal system will yield an incorrect decision. Depending on how a law is written, it will favor one side or the other—if it is going to make a mistake, the mistake will more likely be at the expense of one side. For example, public policy in the United States biases the law in favor of criminal defendants in order to protect those who are accused but innocent. Many who are guilty will go free in order to minimize risk that innocent people will be jailed.

A law that protects sellers from frivolous lawsuits will probably deny some buyers the opportunity to bring legitimate cases to trial. A law that makes sure that genuinely cheated customers can bring suit will probably allow some con artists to bring phony or unreasonable claims to court.

Whether we want to build biases into the law or not, they'll be there. Biases in the law have effects on the people governed by the law. One of the core issues of legal drafting is to pick the right bias in order to support the right public policies. Here are three systems for determining who should pay when a product is allegedly defective or a service is allegedly performed incompetently.

The Fault-Based Approach

If the product was defective, or the service was performed incompetently, there's natural justice in saying that the seller should pay. This is a *fault-based* approach to liability.

First problem with the fault-based approach: How do we define "defective"? The word is surprisingly slippery.

I ventured a definition for serious defects in Kaner (1997a). I think the approach works, but it runs several pages. It explores several relationships between buyers and sellers, and it still leaves a lot of room for judgment and argument. More recently, I was asked to come up with a relatively short definition of "defect" (serious or not). After several rounds of discussion, I'm stalled.

I won't explore the nuances of the definitional discussions here. Instead, here's a simplification that makes the legal problem clear. Suppose we define a defect as failure to meet the specification. What happens when the program does something obviously bad (crashes your hard disk) that was never covered in the spec? Surely, the law shouldn't classify this as nondefective. On the other hand, suppose we define a defect as any aspect of the program that makes it unfit for use. Unfit for who? What use? When? And what is it about the program that makes it unfit? If a customer specified an impossibly complex user interface, and the seller built a program that matches that spec, is it the seller's fault if the program is too hard to use? Under one definition, the law will sometimes fail to compensate buyers of products that are genuinely, seriously defective. Under the other definition, the law will sometimes force sellers to pay buyers even when the product is not defective at all.

Second problem: I don't know how to make a software product that has zero defects. Despite results that show we can dramatically reduce the number of *coding* errors (Ferguson, Humphrey, Khajenoori, Macke, & Matuya, 1997; Humphrey, 1997), I don't think anyone else knows how to make zero-defect software either. (Many errors in software are designed in. The product meets the specification, but it is still unusable or untrustworthy.) If we create too much pressure on software developers to make perfect products, they'll all go bankrupt and the industry will go away.

In sum, finding fault has appeal, but it has its limits as a basis for liability.

Technological Risk Management

It makes sense to put legal pressure on companies to improve their products because they can do it relatively (relative to customers) cheaply. In a mass market product, a defect that occasionally results in lost data might not cost individual customers very much, but if you total up all the costs, it would probably cost the company a great deal less to fix the bug than the total cost to customers. (Among lawyers, this is called the principle of the "least cost avoider." You put the burden of managing a risk on the person who can manage it most cheaply.)

I call this *technological risk management*--because we are managing the risk of losses by driving technology. Losses and lawsuits are less likely when companies make better products, advertise them more honestly, and warn customers of potential hazards and potential failures more effectively.

At our current stage of development in the software industry, I think that an emphasis on technological risk management is entirely appropriate. We (the software industry) save too many nickels in ways that we know will cost our customers dollars. However, we should understand that the technological approach is paternalistic. The legal system decides for you what risks companies and customers can take. This drives schedules and costs and the range of products that are available on the market.

The technological approach makes obvious sense when we're dealing with products like the Pinto, which had a deadly defect that could have been fixed for $11 per car. It's entirely appropriate whenever manufacturers will spend *significantly* less to fix a problem than the social cost of that problem. But over time, this approach gets pushed at less and less severe problems. In the extreme, we risk ending up with a system that imposes huge direct and indirect taxes on us all in order to develop products that will protect fools from their own recklessness.

As society moved in this direction, many companies and individuals found the system intolerable. Starting in the 1970's we were hearing calls for "tort reform" and a release from "oppressive regulations." The alternative is commercial risk management: let buyers and sellers make their own deals and keep the government out of it.

Commercial Risk Management

This is supposed to be a free country. It *should* be possible for a buyer to say to a seller, "Please, make the product sooner, cheaper, and less reliable. I promise not to sue you."

The commercial risk management strategy involves *allocation* of risk (agreeing on who pays) rather than reduction of risk. Sellers rely on contracts and laws that make it harder for customers to sue sellers. Customers and sellers rely on insurance contracts to provide compensation when the seller or customer negligently makes or uses the product in a way that causes harm or loss.

This approach respects the freedom of people to make their own deals, without much government interference. The government role in the commercial model is to determine what agreement the parties made, and then to enforce it. (Among lawyers, this is called the principle of "freedom of contract.")

The commercial approach makes perfect sense in deals between people or businesses that actually have the power to negotiate. But over time, the principle stretches into contracts that are entirely non-negotiated. A consumer buying a Microsoft product doesn't have bargaining power.

Think about the effect of laws that ratify the shrink-wrapped "license agreements" that come with mass-market products. In mass-market agreements, we already see clauses that avoid all warranties and that eliminate liability even for significant losses caused by a defect that the publisher knew about when it shipped the product. Some of these "agreements" even ban customers from publishing magazine reviews without the permission of the publisher (such as this one, which I got with Viruscan, “The customer will not publish reviews of the product without prior written consent from McAfee.”)

Unless there is intense quality-related competition, the extreme effect of a commercial risk management strategy is a system that ensures that the more powerful person or corporation in the contract is protected if the quality is bad but that is otherwise indifferent to quality. Without intense quality-driven competition, some companies will slide into lower quality products over time. Eventually this strategy is corporate suicide, but for a few years it can be very profitable.

Ultimately, the response to this type of system is customer anger and a push for laws and regulations that are based on notions of fault or of technological risk management.

Legal Risk Management Strategies are in Flux

Technological and commercial risk management strategies are both valid and important in modern technology-related commerce. But both present characteristic problems. The legal policy pendulum swings between them (and other approaches). As we’ll see, Article 2B adopts a purely commercial risk management approach, even when dealing with large companies selling under near-monopoly conditions to mass-market customers.

Theories of Software Liability

A legal "theory" is not like a scientific theory. I don't know why we use the word "theory." A legal theory is a definition of the key grounds of a lawsuit. For example, if you sue someone under a negligence theory:

- You must prove that (a) the person owed you a duty of care; (b) the person breached the duty; (c) the breach was the cause of (d) some harm to you or your property.
- You must convince the jury that (a), (b), (c), and (d) are all more likely to be true than false. Ties go to the defendant.
- If you prove your case, you are entitled to compensation for the full value of your injury or of the damage to your property.
- If the jury decides there is clear and convincing evidence that the defendant acted fraudulently, oppressively, maliciously, or outrageously, you can also collect punitive damages. These are to punish the defendant, not to compensate you. The amount of damages should be enough to get the defendant's attention but not enough to put it out of business. Punitive damages are rarely awarded in lawsuits--in a short course for plaintiffs' lawyers on estimating the value of a case, we were told to expect to win punitive damages in about 2% of the negligence cases that we try, and to expect small punitive damage awards in most of these cases. If a jury does assess major punitive damages, the trial court, an appellate court, and sometimes the state's supreme court all review the amount and justification of the award.

Every lawsuit is brought under a specifically stated theory, such as negligence, breach of contract, breach of warranty, etc. For software-related lawsuits, I defined most of these theories, with examples, in Kaner, Falk, & Nguyen (1993) and added some detail in Kaner (1997k). Here's a quick look at a few theories under which a software developer can be sued.

Negligence

Every company has a duty to take reasonable measures to make its product safe (no personal injuries or property damage), or no more unsafe than a reasonable customer would expect (skis are unsafe, but skiers understand the risk and want to buy skis anyway). Under the right circumstances, a company can non-negligently leave a dangerous defect in a product.

Proof of negligence can be quite difficult. No single factor will prove that a company was non-negligent. A court will consider several factors in trying to understand the level of care taken by the company (Kaner, 1996b). Kaner, Falk, & Nguyen (1993) list several factors that will probably be considered in a software negligence case, such as:

- Did the company have actual knowledge of the problem? *(No one likes harm caused by known defects.)*

- How carefully did the company perform its safety analysis? *(The wrong answer is, "Safety analysis? What safety analysis?")*
- How well designed is the program for error handling? *(The law expects safety under conditions of foreseeable misuse. 90% of industrial accidents are caused by "user errors." Manufacturers have to deal with this, not whine about "dumb users.")*
- How does the company handle customer complaints? *(Jurors will sympathize with mistreated customers.)*
- What level of coverage was achieved during testing? *(There are so many different types of coverage. Using judgment is more important than slavishly achieving 100% on one type of coverage.* Kaner, 1996b.*)*
- Did the product design and development follow industry standards? *(In negligence, failure to follow a standard is relevant if and only if the plaintiff can show that this failure caused the harm.)*
- What is the company's bug tracking methodology? *(Does it have one?)*
- Did the company use a consistent methodology? *(If not, how does it make tradeoffs?)*
- What is the company's actual level of intensity or depth of testing? *(Did it make a serious effort to find errors?)*
- What is its test plan? *(How did the company develop it? How do they know it's good? Did they follow it?)*
- What does the documentation say about the product? *(Does it warn people of risks? Does it lead them into unsafe uses of the product?)*

Negligence is the type of lawsuit that many of us think about first when we think about litigation over defective products. It is an interesting lawsuit for us as quality advocates because the software development process is clearly relevant in the suit. However, most software lawsuits are for breach of contract or fraud, not negligence. That's because a defective product doesn't trigger a negligence theory unless the product does personal injury or property damage.

Fraud

The company made a statement of fact (something you can prove true or false) to you. It knew when it made the statement that it was false, but it wanted you to make an economic decision (such as buying a product or not returning it) on the basis of that statement. You reasonably relied on the statement, made the desired decision, and then discovered it was false. In the case of *Ritchie Enterprises v. Honeywell Bull* (1990), the court ruled that a customer can sue for fraud if technical support staff convinced him to keep trying to make a bad product work (perhaps talking him out of a refund), by intentionally deceiving him after the sale.

Negligent Misrepresentation

Like fraud except that the company made a mistake. It didn't know that the statement was false when it made it. If the company had taken the care in fact-finding that a reasonable company under the circumstances would have taken, it would not have made the mistake. *Burroughs Corp. v. Hall Affiliates, Inc.* (1982) is an example of this type of case in a sales situation. You have to establish that the company owed you a duty to take care to avoid accidentally misinforming you. This is often very difficult to prove, especially if the company made a false statement about something that it was not selling to you. However, independent test labs have been successfully sued by end customers for negligently certifying the safety of a product (Kaner, 1996d).

FTC Enforcement

The Federal Trade Commission can sue companies for unfair or deceptive trade practices, unfair competition or other anti-competitive acts. Many defendants settle these cases without admitting liability. Recent FTC cases have been settled (without admission of liability) against Apple Computer (*In the Matter of Apple Computer*, 1996) and against the vendor of a Windows 95 optimization program that allegedly didn't provide any performance or storage benefits (*In the Matter of Syncronys Software*, 1996). Occasionally, the FTC sues over vaporware announcements that appear to be intended to mislead customers.

Breach of Contract

In a software transaction, the contract specifies obligations that two or more persons have to each other. (In legal terms, a "person" includes humans, corporations, and other entities that can take legally binding actions.) Contracts for non-customized products are currently governed under Article 2 (Law of Sales) of the Uniform Commercial Code (UCC). Contracts for services, including custom software, are covered under a more general law of contracts.

UCC transactions carry implied terms. For example, products normally come with an implied warranty of merchantability (the product will be fit for ordinary use, it will conform to the claims on the packaging and in the manual, and it will pass without objection in the trade). This reflects a basic American public policy, that some modest standards of integrity should be applied to the sales of goods. To disclaim an implied warranty of merchantability (i.e., to legally effectively say that there is no such warranty), a merchant seller must put a conspicuous disclaimer in the contract. (A company that sells software in the ordinary course of its business would be a software merchant.) Court cases have almost always required this notice to be given to the customer in a way that makes it conspicuous (likely to be seen) before or at the time of sale (White & Summers, 1995, volume 1). Here is Clark & Smith's summary from The Law of Product Warranties (1984; supplemented 1994) p. 8-18:

> The disclaimer may be found in an operator's manual, or in sales literature not sent to the buyer until sometime later, or it may be hidden away in the package that also contains the goods, with no warning on the outside of the package. In all of these cases, it is as though the seller were determined that the buyer should not see the disclaimer until after the fact. Given this seller perversity, it is not surprising that the courts generally nullify such post-contract disclaimers.

The specific validity of shrink-wrapped software warranty disclaimers that are not visible to the customer until after the sale was considered in two opinions, *Step-Saver v. Wyse Technology and The Software Link* (1991), and *Arizona Retail v. The Software Link* (1993). The courts ruled that an implied warranty comes with a product at the time of sale unless it is conspicuously disclaimed, and that a conspicuous disclaimer that is not available to the customer until after the sale is merely a proposal to modify the contract, and is not part of the contract unless the customer agrees. A recent case, *ProCD, Inc. v. Zeidenberg* (1996), appears to say that anything in a shrink-wrapped license is valid but that case didn't consider warranty disclaimers. Nor did a similar second opinion, *Hill v. Gateway 2000, Inc.* (1997) issued by the same court. It will be a major departure from UCC history if a court enforces a warranty disclaimer that was not available to be seen by the customer before the sale.

The UCC also says that express warranties cannot be disclaimed. An express warranty is any statement of fact (something you can prove true or false) by the seller to the buyer about the product that becomes part of the basis of the bargain. The phrase "basis of the bargain" is to be interpreted expansively. The exact rules vary from state to state, but if a reasonable customer would interpret the seller's pre- or post-sale statements as factual descriptions of the product that the customer has bought, and would be even slightly influenced by the statements in deciding whether to buy or keep the product, then you should think of them as "basis of the bargain" statements. (See Kaner, 1995a, Kaner & Pels, 1996, and the court case, *Daughtrey v. Ashe,* 1992.)

The ***Magnuson-Moss Warranty Improvement Act*** goes beyond the UCC and specifies that consumers of goods are entitled to a clear statement of products' warranty terms. For goods costing $15 or more, merchant sellers are required to make warranties available to customers before the sale. *The Code of Federal Regulations* specifies ways in which sellers can meet this requirement (such as having a binder with a copy of each product's warranty, and a conspicuous sign that informs customers where the binder is.) The Software Publishers Association's own handbook of software contracts (Smedinghoff, 1993) states that for consumer software (any off-the-shelf product that is commonly used for personal, family or household use), the Magnuson-Moss Act probably applies.

Quality Cost Analysis

Any legal theory that involves "reasonable efforts" or "reasonable measures" should have you thinking about two things:

- We aren't just looking at a product in this case. The process used to develop the product is at least as important as the end result.
- The judge or jury will do a cost/benefit analysis if this type of case ever comes to trial.

We are, or should be, familiar with cost/benefit thinking, under the name of "Quality Cost Analysis" (Gryna, 1988; Campanella, 1990). Quality cost analysis looks at four ways that a company spends money on quality:

- prevention
- appraisal (looking for problems)
- internal failure costs (the company's own losses from defects, such as wasted time, lost work, and the cost of fixing bugs), and
- external failure costs (the cost of coping with the customer's responses to defects, such as the costs of tech support calls, refunds, lost sales, and the cost of shipping replacement products).

Exhibit 3 provides software examples of quality costs.

It's useful to realize that you don't have to set up a formal cost tracking system to get many of the benefits of quality cost analysis.

Juran developed cost of quality analysis as a persuasive technique. "Because the main language of [corporate management] was money, there emerged the concept of studying quality-related costs as a means of communication between the quality staff departments and the company managers" (Gryna, 1988, p. 42). You can argue from monetary data without ever developing complex cost-tracking systems. When a product has a significant problem, it will cost the company money. Figure out which department is most likely to lose the most money as a result of this problem and ask the head of that department how serious the problem is. How much will it cost? If she thinks its important, bring her to the next product development meeting and have her explain how expensive this problem really is. The data are approximate, inexpensively obtained, but they provide strong persuasive benefit.

In quality cost analysis, external failure costs reflect the company's costs, not the customer's. (For example, see Campanella, 1990.) Previously (Kaner, 1995b, 1996a), I pointed out that this approach sets us up to ignore the losses that our products cause our customers. Thus, in quality cost analysis of the Pinto, it was cheaper for the company to ship cars that were dangerously defective. Exhibit 4 shows a quality cost analysis for the Pinto. If you focus on the costs to the company, it is obvious that (from a cost of quality point of view) you should ship the Pinto as is, which is what Ford did.

However, the law cares more about the customer's losses than the manufacturer's. And from the customer's point of view, risks of death or serious burns are big potential costs. Ford valued the cost associated with a killed customer at $200,000. What do you think? Would you consider $200,000 an even trade for your son or daughter? (Even in 1960s dollars?)

A manufacturer's conduct is unreasonable if it would have cost less to prevent or detect and fix a defect than it costs customers to cope with it (Kaner, 1996b). And so, in the Pinto cases, the juries determined that the defects caused more cumulative harm to society than the total savings made by the company, therefore the company did not take reasonable care to ship a product that was not unnecessarily dangerous, and therefore it was liable for damages. In my view, the sad part in cases like these is that well-meaning employees probably think that they're doing the right thing. The quality cost analysis encourages employees to think about their companies' costs, and doesn't encourage them to think about their customers' costs. If people see traditional quality cost analysis as the right way to make quality-related business tradeoffs, then cases like the Pinto are a natural result. If we don't pay careful attention to the severity of defects for customers, we won't recognize that some circumstances are going to cost customers huge amounts and are candidates for big-verdict lawsuits. If our customers' losses are significantly worse than our external failure costs, then under a traditional quality cost analysis, we risk being blindsided by unexpectedly expensive litigation.

Exhibit 5 lists some of the external failure costs suffered by customers. None of these appear on the traditional external failure cost charts because they are absorbed outside of the company. They have been externalized and they affect the company only indirectly.

Exhibit 3. Examples of Quality Costs Associated With Software Products.

Prevention	***Appraisal***
• Staff training • Requirements analysis • Early prototyping • Fault-tolerant design • Defensive programming • Usability analysis • Clear specification • Accurate internal documentation • Evaluation of the reliability of development tools (before buying them) or of other potential components of the product	• Design review • Code inspection • Glass box testing • Black box testing • Training testers • Beta testing • Test automation • Usability testing • Pre-release out-of-box testing by customer service staff
Internal Failure	***External Failure***
• Bug fixes • Regression testing • Wasted in-house user time • Wasted tester time • Wasted writer time • Wasted marketer time • Wasted advertisements • Direct cost of late shipment • Opportunity cost of late shipment	• Technical support calls and preparation of support answer books • Investigation of customer complaints • Refunds and recalls • Coding / testing of interim bug fix releases • Shipping of updated product • Added expense of supporting multiple versions of the product in the field • PR work to soften drafts of harsh reviews • Lost sales • Lost customer goodwill • Discounts to resellers to encourage them to keep selling the product • Warranty costs • Liability costs • Government investigations • Penalties and all other costs imposed by law

Exhibit 4. Quality Cost Analysis for the Pinto

Benefits and Costs Relating to Fuel Leakage Associated With the Static Rollover Test Portion of FMVSS 208
<u>Benefits (reduced external failure cost) of fixing the design</u> Savings – 180 burn deaths, 180 serious burn injuries, 2100 burned vehicles Unit Cost -- $200,000 per death, $67,000 per injury, $700 per vehicle Total Benefit – 180 x ($200,000) + 180 x ($67,000) + 2100 x ($700) = $<u>49.5 million</u>.
<u>Costs (increased internal cost) to fix the design</u> Sales – 11 million cars, 1.5 million light trucks. Unit Cost -- $11 per car, $11 per truck Total Cost – 11,000,000 x ($11) + 1,500,000 x ($11) = $<u>137 million</u>.

Exhibit 5. Customers' Externalized Failure Costs

- Wasted time
- Lost data
- Lost business
- Embarrassment
- Frustrated employees quit
- Failure when attempting tasks that the customer has only one chance to do correctly (e.g. sales call fails because of software used to assist in the demos.)
- Cost of replacing product
- Cost of reconfiguring the system
- Cost of recovery software
- Cost of tech support
- Cost of third-party technical consultation
- Cost of legal consultation
- Cost of litigation
- Injury / death

A New Law for Software Contracting

With all the background in place, we come back to the Uniform Commercial Code and Article 2B.

The Uniform Commercial Code (UCC) is maintained and updated by the National Conference of Commissioners on Uniform State Laws (NCCUSL), a legal drafting organization funded by the 50 US states that writes all "Uniform" laws. NCCUSL has significant influence with state legislatures. Unless there is serious public opposition, the odds are good that a NCCUSL-drafted bill will pass. The UCC is co-maintained by the American Law Institute, another nonprofit body of senior lawyers.

Background of Article 2B

The UCC is being revised to include a new Article, 2B (Law of Licensing of Information). This draft statute, which runs 250-300 pages, is to cover all contracts for the development, sale, maintenance, and support of software along with many other information-related contracts.

This work started in the American Bar Association, several years ago. It became a NCCUSL project around 1992. It crystallized as the Article 2B project in 1995. To this point there was little customer-side advocacy. I started attending these meetings at the second 2B meeting, in February 1996. Article 2B is scheduled for completion in July 1998 and submission to state legislatures in the fall of 1998.

There is great benefit in creating a uniform legal system for software products *and* services, which works the same way across all states. Unfortunately, this particular proposal for unifying the law is seriously flawed (Kaner, 1996c, 1996e, 1997a, 1997b, 1997c, 1997d, 1997f, 1997h; Kaner & Lawrence, 1997; see `www.cptech.org` for the *Article 2B Protest Page* developed by Todd Paglia, an attorney working with one of Ralph Nader's organizations, the Consumer Project on Technology; see `www.webcom.com/software/issues/guide/docs` for several critical articles written by Gail Hillebrand, an attorney representing Consumers Union. This site primarily includes articles from publishers and others favorable to Article 2B.)

I am most interested in the rights of individuals and small businesses and I therefore focus my analyses of Article 2B on packaged software, especially software developed for the mass market.

In the mass-market case, Article 2B changes the law in two fundamental ways. It adopts a contracting model that excludes negotiation and that doesn't reveal terms of the contract to the customer until after the sale is complete. It also adopts a licensing model, saying that when you buy software, you are really buying only a limited right to use the software rather than a copy of the software product. These two changes have significant repercussions.

Adoption of Non-Negotiable, Post-Sale Contracts

Under Article 2B, ***the publisher doesn't have to make the terms of its contract available to the customer until after the customer has paid for the product, taken it away, and started installing it on her computer***. These contracts are usually called "shrink-wrapped" contracts because they are inside the shrink-wrapped package.

There is no opportunity for negotiation of terms. The customer is deemed to have accepted the terms, if she uses the product instead of returning it. With extremely few exceptions, all of the terms in this "agreement" will be fully enforceable against the customer as if she had reviewed, discussed, and signed a paper contract before the sale. This is not the traditional approach.

The traditional analysis (UCC Article 2) is that the sales contract is formed when the customer agrees to buy the product and then pays for it. Terms not specified in the contract are provided under default rules by the UCC. Terms revealed to the customer after this contract is formed (for example, in the shrink-wrapped "agreement") are proposals to modify the contract. They will come into the contract if they are not material (significant changes), but they will not come into the contract if they are material.

One common term that is held to be a material alteration is the warranty disclaimer. As I mentioned in the section on Breach of Contract above, courts normally refuse to enforce warranty disclaimers that are shown to the customer only after the sale has been made. For software cases, see *Step-Saver v. Wyse Technology and The Software Link* (1991) and *Arizona Retail Systems v. The Software Link* (1993).

Article 2B rejects the notion that publishers should reveal material terms of the agreement to the customer before the sale. The Reporter's Notes to Section 2B-308 (Mass-Market Licenses) of the February, 1996 draft of Article 2B, state:

```
This section reverses Wyse Technology v. Step-Saver, where the
court used 2-207 to hold that a shrink wrap license in software
packages delivered after a prior telephone contract did not become
part of the sale contract. See also Arizona Retail Sys., Inc. v.
Software Link, Inc., 831 F. Supp. 759, 22 UCC Rep. Serv2d 70 (D
Ariz. 1993) (shrink wrap enforceable in transaction where no prior
agreement, but not enforceable where there was a prior telephone
agreement).
```

The view that publishers can enforce whatever terms they include in these shrink-wrapped "licenses" has been accepted in court only very recently (*ProCD, Inc. v. Zeidenberg,* 1996; *Hill v. Gateway 2000, Inc.*, 1997). Both of these decisions are from the same court. Much of the District Court's analysis of the ProCD case involved discussion of the then-current draft of Article 2B. Even though it is not law, Article 2B is already able to pollute the American legal system because it is being drafted by such a prestigious legislative drafting organization (NCCUSL).

If the seller doesn't have to tell you what the deal is until after the sale is completed, you can bet that the terms that the seller didn't tell you about before the sale will probably not be as warm and friendly as the terms he did tell you about.

Adoption of a Licensing Model

Under current law, packaged, non-customized, software is governed as "goods" under current UCC Article 2. Rodau (1986) is a well-regarded review of the legal literature on this issue and the many subsequent court decisions have been consistent with his summary. Also as noted above, under current law, even the Software Publishers Association's book on contracts states that the Magnuson-Moss Act (which applies to sales of consumer goods) probably governs purchases of consumer software.

A book is a good example of an information product governed by Article 2. It is a tangible piece of property (you can pick it up) and it contains copyrighted material. The primary value of the book is the copyrighted content not the paper on which it's printed. Off-the-shelf software is also copyrighted content that comes on a disk (let's set aside for now the complication that you can download both books and software electronically, skipping paper and disk). Books, records, videotape movies, and software are sold in the same types of stores, in the same way. You pick one up, take it to the counter, and pay for it. Or you order it mail order. Or you download it.

When you buy a book, you have certain rights under the Copyright Act and under the United States Constitution (article 1, §8, clause 8), which gives Congress the power "To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Rights to their respective Writings and Discoveries." The Constitution balances a public interest in making use of these works against a private interest in collecting money for them.

You don't have the right to make unauthorized copies of a copyrighted work, but you can quote short pieces of it, publish a review of it, use the information you get from it to research a paper that the author or publisher would disagree with, let a friend read the book, or analyze the author's writing style. These are examples of "fair use" and "first sale" rights.

Software publishers claim that mass-market software sales are not sales. Instead, they say that these transactions are licenses. Licenses involve restrictions on the use of intellectual property. For example, traditional licenses can have nondisclosure provisions (no book reviews), restrictions on how the product is used (no quoting, no use for unapproved purposes) and on who can use it (no friends), restrictions on reverse engineering (no analyzing the style), restrictions on transfer of the licensed product (no selling the book as used once you're done with it.)

This claim by publishers is very controversial. Mass-market customers do not view their purchases of MS Word (or of books) as licensing transactions. Few customers believe that they are subject to all of the restrictions that are commonplace in shrink-wrapped software "license" agreements.

The shrink-wrapped, mass market "licenses" are quite different from traditional license contracts, which are often used for larger, non-mass-market software transactions.

Traditional licenses (software, patent, trademark, etc.) are non-anonymous transactions. The licensor chooses who it will contract with. The product is not available to the general public. For example, competitors do not qualify. Licenses involved signed contracts, often heavily negotiated, and the licensor makes sure that the licensee signs the contract and understands the restrictive terms before concluding the deal.

In contrast, the mass-market license is not negotiable and not signed. The license is available to the general public. Anyone can buy it: if your competitor can afford it, your competitor can buy it. There is no up-front effort to get the mass-market licensee to agree to significant use restrictions. Instead, they are hidden inside the package.

Article 2B adopts the publishers' view and says that these mass-market "licenses" are real licenses, just as valid as the non-anonymous ones. Therefore, every term that should be valid in a traditional license should be valid in a mass-market license.

Article 2B explicitly allows "contractual use restrictions" which "include obligations of nondisclosure and confidentiality and limitations on scope, manner, method, or location of use to the extent that those obligations or duties are created by the contract." (Article 2B § 102(a)(10) November 1, 1997 draft.)

ProCD follows this reasoning. *ProCD* was a fair use case involving a database that contained a telephone directory. This is factual, non-copyrightable content. If the customer had purchased a copy of the database, he could have posted its content to the Net to his heart's content. Instead, the court called the transaction a license and said that use restrictions are entirely permissible in licenses. A license, you see, is merely a contract between a licensee and a licensor. There are no public rights implicated in private contracts. And therefore, there is no loss of public rights in a restriction that deprives an individual licensee from making a use that would be, for a buyer, a fair use of the licensed information.

Mass-market licenses are new beasts and their legal status today is uncertain. Everything about the transaction looks like a sale, except for the piece of paper buried inside the box. It looks to me as though the mass-market licensing model is merely an end run around the Constitutional balancing of buyers' and sellers' rights. The sellers want the money, but they don't want to give the buyers their fair use and first sale rights. Several other attorneys (for example, Karjala, 1997; Lemley, 1995; Rice, 1997) make essentially this point, with more extensive legal discussion.

It is probable that *some* restrictions contained in mass market licenses will be struck down by federal courts as unconstitutional (or as violative of federal public policy). But we don't know which ones. The *ProCD* and *Gateway 2000* decisions suggest that the federal courts might not strike very many (if any) of these restrictions. Until courts rule consistently that a given restriction can be safely ignored, I will have to take it seriously in an Article 2B contract, just like every other licensing lawyer. Things that I would advise clients to ignore today, I will have to say, "maybe that is enforceable" under Article 2B.

Consequences for Software Quality

Quality management using quality cost analysis normally seeks an optimum level of expenditure on quality, by balancing the amount of money we spend on making the product better against the amount we save by having made it better (see Campanella, 1990). Under this approach, if you reduce the cost of bad quality (failure costs), you reduce the incentive to invest in improvement.

When I analyze Article 2B from the viewpoint of quality costs, I see a comprehensive scheme to reduce external failure costs without improving quality.

External failure costs are often underestimated. There are many types and some are hard to measure. For convenience, I'll lump them into three categories:

- ***Customer support costs***, such as the cost of answering all those phone calls, the cost of refunds, free upgrades, etc.
- ***Lost sales.*** Companies often underestimate the extent to which defects cause a loss of repeat business and a loss of goodwill.
- ***Legal costs,*** such as lawsuits, fines, and lawyers' fees.

Let's consider how Article 2B will impact each of those costs. One point to consider as background is that consumer protection laws that apply specifically to sales of goods (such as the Magnuson-Moss Warranty Improvement Act) will no longer apply to software, because software transactions will no longer be "sales of goods." Don't take the consumer safety net for granted.

Customer support costs can be reduced by charging the customer for support and by limiting the customer's rights to refunds and upgrades. Here are examples of the problems posed by Article 2B. This is not a complete list.

- Mass-market publishers are increasingly charging $3 per minute, or $20-$150 per call or per incident for support calls. Some companies require the customer to pay fee-based support from the first minute of the customer's possession of the product, even for actual defects in the product that are already known to the publisher. Most publishers allow a grace period of 30 or more days during which the customer can call about problems without paying a fee (Software Publishers Association, 1995) but some do not.

  This practice is not new under 2B, but we should understand it in the context of 2B. Article 2B allows publishers to put damage limitation clauses in the license that exclude incidental expenses. Incidental expenses include all costs of reporting a defect and returning the product. ***A customer who spends money on support calls isn't entitled to a refund of these charges even if she was calling about defects that were known to the publisher at the time it shipped the product.*** Article 2 allows sellers to exclude incidental and consequential damages too, but there are safeguards. A court can throw out the exclusion if it determines that a shrink-wrapped exclusion clause is a material change to the contract. Additionally, Article 2's Official Comments advise courts that a non-breaching party (here, the customer) is entitled to a "minimum adequate remedy." Article 2B dropped all references to the notion of a "minimum adequate remedy." For the first time, the law will clearly say that the unscrupulous publisher gets to profit from its own defects, and the customer has no recourse.
- Article 2B will also make it easier for publishers to refuse to give refunds. First, under Article 2, a customer has a right to reject merchandise that has any defect. I explain the application of this "perfect tender rule" in Kaner & Pels (1998). The perfect tender rule was initially completely eliminated in Article 2B, but Consumers Union

fought this so adamantly that it has been restored for mass-market software, only. If you buy software that is not "mass-market" (including off the shelf software that is too specialized or too expensive to be class as mass-market, and $500 has often been cited as the "too expensive" amount) then you no longer have the right to reject a software product due to non-material defects that you discover during incoming inspection.

- Article 2B also reduces the opportunity for a refund by saying that the publisher is required to offer a refund only if the product is so defective that it has materially breached the contract (the contract written by the publisher). To my reading, the definition of "material" in Article 2B is much more publisher-friendly (requires a more seriously defective product) than the one it is allegedly based on, from the American Law *Institute's Restatement of Contracts*. But even for a significant bug, the company can force a customer to prove in court that the bug is so serious that the customer is entitled to a full refund and not just a small partial one. This affects the negotiating power of customers when they call for technical support.
- Article 2B also reverses consumer rights to a nondefective product that would otherwise entitle them to a free bug-fix version of defective software. Under the Magnuson-Moss Act, purchasers of a defective consumer product who want a fixed product and not a refund are entitled to a repair unless this is commercially unreasonable. Remember that the Magnuson-Moss Act applies to sales of goods, so software licenses (which are not goods) will fall outside of its scope. 2B doesn't give consumers this right. Instead, it requires publishers to try to provide bug fixes only for business customers of non-mass-market products.
- Finally, Article 2B drops the requirement that a disclaimer of the implied warranty of merchantability be conspicuous before the sale, when it might influence the customer's buying decision. California's Song-Beverly Act requires *pre-sale* conspicuousness for consumer products, but it applies to sales of goods. The Magnuson-Moss Act requires that warranties be available for customers to read before the sale, but it applies to sales of goods. Under 2B, software transactions aren't sales of goods so these laws go away. Instead, Article 2B redefines the word "conspicuous" so that a capitalized disclaimer that appears only in the inside-the-box license is "conspicuous" because it is capitalized, even if the store refuses to open the box and let the customer read the warranty (a very common refusal in software stores, which would be illegal under the Magnuson-Moss Act) before the sale is completed. ***What possible benefit is there to the public of a law that cuts off their right to know before the sale what guarantees the product comes with? And what do you think this will do for software quality?***

In sum, Article 2B will help publishers reduce their customer support costs in ways that don't improve the quality of their products.

Lost sales occur when a potential customer either buys nothing or buys a competitor's product. Article 2B supposedly leaves the regulation of the integrity of the industry to a supposedly free market that will kill bad companies through competition. But Article 2B will allow the publisher to limit the effects of competition by limiting the flow of bad information about its products, by delaying or blocking the development of competing products and by limiting the availability of competing support services for its products. I review these issues in more detail in Kaner (1997j). Here are some examples.

- One of the core tenets of consumer protection is the notion of informed choice. Critical information about a product is made available to the consumer *at or before the sale* so the consumer is in a position to understand the relative benefits from competing products. Article 2 requires that some terms be conspicuous and rejects material changes to a contract after the sale. This makes sure that the customer (not just a consumer in Article 2—any customer) becomes aware of particularly oppressive terms before the sale. In contrast, Article 2B says that the publisher doesn't have to show software customers the terms until after the sale, when it's too late to do comparison shopping. The customer who objects to the terms is allowed to take the software back, for a refund. In practical terms, this isn't of much value. Once the customer has taken the product home or to the office, and started loading it on her computer, she is no longer in a shopping frame of mind. She's much less likely to reject harsh legal terms in a license after she's committed himself to the product, and returning it would be an often-significant inconvenience. Further, her quest for better terms will be a matter of trial and error—buy a competing product, discover its terms, take it back, buy another product, etc.

▪ Article 2B will also allow software publishers to limit news publication of comparative reviews and critical reviews of their products. Remember, freedom of speech means the *government* can't stop you from talking. You can always sign contracts that require you to keep secrets.

  Article 2B's definition of contractual use restrictions (102(a) (10)) lets publishers use confidentiality clauses in their license agreements. When you buy mail order or in a store, who can the publisher keep the behavior of the product secret from? Any person willing to pay the price can examine the product, including competitors. Yet we see shrink-wrapped clauses like “You agree to hold the Package within your Organization and shall not, without our specific written consent . . . publish or communicate or disclose to third parties any part of the Package” (Symantec). Some publishers are even more direct. For example, “The customer will not publish reviews of the product without prior written consent from McAfee.”

I don’t think that these clauses are valid under current law. Some people have told me that the courts won't enforce these restrictions under Article 2B either, because people have rights of free speech. Maybe they're right, but the battle will not be a sure victory. Publishers have the rights to create trade secrets and to enter into nondisclosure contracts with people. The courts enforce those contracts. What is new here is that the publisher is creating a nondisclosure contract with the whole world, one customer at a time and there is a law that appears to specifically authorize this. Anyone who can afford the contract can learn the secret, as long as they pay. This is a big extension of the law of secrets, but a court might be reluctant to say that such an extension, adopted by most or all state governments, is unconstitutional and therefore invalid. And it will take a lot of money to fight the lawsuits all the way up to the Supreme Court to establish this invalidity. In the meantime, some large magazines might reject the law and print critical articles anyway, but other magazines and individuals with their own Web sites might be much more cautious about speaking their mind. After a few well-publicized enforcement letters from publishers' lawyers, and perhaps a favorable court decision or two, many people will be afraid to speak their mind because the contract says they can't.

What benefit is there to the public of a law that lets publishers cut off their customers' right to read detailed, critical reviews of a product they are considering buying? How can we assure competition in the marketplace if publishers can block negative reviews of their products?

- Article 2B also appears to give publishers rights to restrict reverse engineering. When products are sold, they can be reverse engineered. This is well-established under both patent and copyright law. See for example, *Bonita Boats, Inc. v. Thunder Craft Boats, Inc.* (1989) and *Sega Enters. Ltd. v. Accolade, Inc.* (1992). However, restrictions on reverse engineering can definitely be included in traditional licenses, and so they will be valid in Article 2B mass-market licenses until and unless the federal courts say that this is a prohibited restriction. Some authors, such Gomulkiewicz & Williamson (1996), think that mass-market bans on reverse engineering should be lawful. (Gomulkiewicz is one of Microsoft’s lawyers.) Others predict that such clauses will be tossed out because they are anticompetitive (Lande & Sobin, 1996).

  What benefit is there to the public of a law that lets software publishers limit the research opportunities of their competitors in ways not allowed for other industries?

- Article 2B also appears to allow publishers to restrict development of products that are interoperable with their products, by restricting reverse engineering.

  Suppose that you have lots of documents that you've written using your current word processor. I offer to sell you a new word processor that you really like. Unfortunately, my program can't read your old files. Would you buy my program? Probably not. To make my program a viable competitor, I have to read those files. I probably can’t get the file formats from the other word processing companies, so to achieve interoperability with those products, to be able to read their files and write files that they can read, I’ll have to reverse engineer them.

  I've talked through this example with some publishers' lawyers who have expressed the position that they should be allowed to restrict reverse engineering of their file formats. They felt that they should be able to decide who to license this information to. They like the idea of being able to choose their competitors.

  Understand what this means for competition. A new company that has little existing technology will find it difficult or expensive to license competitive technology and, unlike the situation in every other industry, they will be barred from reverse engineering it. ***By making interoperability very hard to achieve, Article 2B makes it hard for a new competitor to enter the market. It is a kiss of death for low-budget startups.***

- My final example on competition involves post-sale support. A license can restrict who can use the product. A publisher who doesn’t want anyone to provide independent support services for its software can restrict use of its software to “bone fide employees” of its customers. Third-party support staff are not bone fide employees of the companies to whom they provide support services. Therefore, they can’t use the software that they’re trying to support. *MAI Systems Corp. v. Peak Computer, Inc*. (1993) illustrates this trick. It appears to work for negotiated licenses, but why should we want to allow this kind of restriction in mass-market, non-negotiable licenses?

Finally, there are ***legal costs***. Most disputes between customers and publishers are breach of contract disputes. It is a breach of contract to sell a product that is supposed to work and deliver a product that does not. Most disputes will therefore be governed by Article 2B. We've already seen that Article 2b will make it harder to sue publishers, because it eliminates non-mass-market customers' right of refusal of an imperfect product, it makes it easier to disclaim warranties, it restricts the customer's right to a refund to cases in which a breach of contract is "material" and it creates a new, publisher-friendly definition of "material." Article 2B makes it easy to exclude repayment to the customer of incidental and consequential damages and drops the goal of a minimum adequate remedy, so customers are less likely to recover much even if they do sue. Just in case these (and many other restrictions) aren't enough to discourage genuinely cheated customers from pressing their rights, Article 2B lets the publisher pick what state or country's laws will govern the license and what state or country a customer has to travel to in order to sue the publisher. There are slight restrictions on this if the customer is a "consumer" who uses the software for strictly non-business, non-professional purposes (the teacher who does research or writes assignments at home is not acting as a consumer, nor is the unemployed secretary who tries out some home-based network marketing scheme and uses a computer to manage her mailing lists and print fliers.) But even within these restrictions, a California corporation could probably enforce a shrink-wrapped license clause against a California consumer that allows customers to file suit only in New York. We've talked about this repeatedly and at length in the Article 2B meetings. Everybody understands that this will virtually eliminate access to small claims courts, and that small customers will be effectively barred from bringing many types of lawsuits by this rule. This is not an accident and not an oversight.

There are many more problems with 2B than this. I walk through them in detail in Kaner (1997h).

Conclusions

Throughout Article 2B there is tolerance for software defects. For example, in the Reporters comments we see "the perfect tender rule [should] be abolished with respect to software contracts because of the complexity of the software product and the fact that minor flaws ('bugs') are common in virtually all software." Watts Humphrey a leading advocate for software quality has protested this tolerance of defects at Article 2B meetings (and see Humphrey, 1997).

Article 2B takes substantial force away from my most powerful tool for arguing that bugs should be prevented or fixed, by helping publishers dramatically limit the costs associated with shipping bug-ridden products. As a quality advocate, I find this appalling.

REFERENCES

Arizona Retail Systems, Inc. v. The Software Link (1993) *Federal Supplement*, volume 831, p. 759 (United States District Court, Arizona).

Bonito Boats, Inc. v. Thunder Craft Boats, Inc. (1989*) United States Reports,* volume *489,* p. 141 (United States Supreme Court).

Burroughs Corp. v. Hall Affiliates, Inc. (1982) *Southern Reporter, Second Series*, Volume 423, p. 1348 (Supreme Court of Alabama).

Campanella, J. (Ed.) (1990). *Principles of Quality Costs*, 2nd Ed. ASQC Quality Press.

Clark, B. & Smith, C. (1984, supplemented 1994) *The Law of Product Warranties*, Warren Gorham & Lamont.

Daughtrey v. Ashe (1992) *South Eastern Reporter, Second Series*, volume 413, p. 336 (Supreme Court of Virginia).

Deming, W.E. (1986). *Out of the Crisis.* MIT Press.

Ferguson, P., W.S. Humphrey, S. Khajenoori, S. Macke, & A. Matuya (1997) Results of Applying the Personal Software Process, *IEEE Computer*, May, 1997, p. 24-31.

General Motors Corp. v. Johnston (1992), *Southern Reporter*, 2nd Series, volume 592, p. 1054.

Gomulkiewicz, R.W. and Williamson, M.L. (1996) A Brief Defense of Mass Market Software License Agreements, *Rutgers Computer & Technology Law Journal*, volume 22, p. 335. Available at http://www.SoftwareIndustry.org/issues/guide/docs/bdsw.html

Gryna, F. M. (1988) "Quality Costs" in Juran, J.M. & F. M. Gryna (1988, 4th Ed.), *Juran's Quality Control Handbook*, McGraw-Hill.

Hill v. Gateway 2000, Inc. (1997) Docket No. 96 C 4086 (1997 WL 2809), (United States Court of Appeals, 7th Circuit).

Humphrey, W.S. (1997) Comments on Software Quality. (unpublished) *Annual Meeting of the National Conference of Commissioners on Uniform State Laws,* July 25 – August 1, 1997, Sacramento, CA. `Available at www.webcom.com/software/issues/guide/docs/whsq.html`.

In the Matter of Syncronys Software. (1996), Docket C-3688. Complaint at `www.ftc.gov/os/9610/c3688cmp.htm`. Decision and order at `www.ftc.gov/os/9610/c3688d&o.htm`.

In the Matter of Apple Computer, Inc. (1996), Docket C-3763. Complaint at `www.ftc.gov/os/9708/c3763cmp.htm`. Decision and order at `www.ftc.gov/os/9708/c3763ord.htm`.

Johnson v. Compaq (1997). Class action complaint filed against Compaq Computer in North Carolina. For the text, see `users.aol.com/Cclass450/index.htm`. This is a remarkable series of documents, and it is kept updated.

Kaner, C. (1995a) Liability for Defective Documentation. *Software QA*, *2*, #3, 8. Available at `www.badsoftware.com/baddocs.htm`.

Kaner, C. (1995b) "Software Quality & the Law" in *The Gate* (newsletter of the San Francisco Section of the American Society for Quality Control), July 1995, p. 1.

Kaner, C. (1996a). Quality Cost Analysis: Benefits and Risks. *Software QA*, *3*, #1, 23. Available at `www.badsoftware.com/qualcost.htm`.

Kaner, C. (1996b). Software Negligence and Testing Coverage. *Proceedings of STAR 96 (Fifth International Conference on Software Testing, Analysis, and Review)*, Orlando, FL, May 16, 1996, 313. Available at `www.badsoftware.com/coverage.htm`.

Kaner, C. (1996c). Uniform Commercial Code Article 2B: A new law of software quality. *Software QA*, *3*, #2, 10. Available at `www.badsoftware.com/uccsqa.htm`.

Kaner, C. (1996d). Liability for Defective Content. *Software QA*, *3*, #3, 56.

Kaner, C. (1996e) Warranty and Liability Hypotheticals for UCC Article 2B. (unpublished.) *Meeting of the NCCUSL Article 2B Drafting Committee*, Philadelphia, PA, April 26-28, 1996.

Kaner, C. (1996f). Computer Malpractice. *Software QA*, 3, #4, 23. Available at `www.badsoftware.com/malpract.htm`.

Kaner, C. (1996g). Contracts for Testing Services. *Software QA*, *3*, #5, 20.

Kaner, C. (1997a). What is a Serious Bug? Defining a "Material Breach" of a Software License Agreement. (unpublished.) *Meeting of the NCCUSL Article 2B Drafting Committee*, Redwood City, CA, January 10-12, 1997. (abbreviated version, *Software QA*, 3, #6.) Available at `www.badsoftware.com/uccdefect.htm`.

Kaner, C. (1997b). Remedies Provisions of Article 2B. (unpublished.) *Meeting of the NCCUSL Article 2B Drafting Committee*, Redwood City, CA, January 10-12, 1997. Available at `www.badsoftware.com/uccrem.htm`.

Kaner, C. (1997c). Proposed Article 2B: Problems from the Customer's View: Part 1: Underlying Issues. *UCC Bulletin*, January, 1-8. Available at `www.badsoftware.com/uccpart1.htm`.

Kaner, C. (1997d). Proposed Article 2B: Problems from the Customer's View: Part 2: List of Key Issues. *UCC Bulletin*, February, 1-9. Available at `www.badsoftware.com/uccpart2.htm`.

Kaner, C. (1997e). Liability for Bad Software and Support. *Proceedings of the Support Services Conference East,* Nashville, TN, March 12, 1997. Available at `www.badsoftware.com/support1.htm`.

Kaner, C. (1997f). Not Quite Terrible Enough Software. (unpublished.) *Annual Meeting of the Software Engineering Process Group*, San Jose, CA, May 1997. Available at `www.badsoftware.com/sepg.htm`.

Kaner, C. (1997g). The Impossibility of Complete Testing. *Software QA*, *4,* #4, 28.

Kaner, C. (1997h), Article 2B is Fundamentally Unfair to Mass-Market Software Customers. (unpublished.) Circulated to the American Law Institute for its Article 2B review meeting, October 1997. Available at `www.badsoftware.com/ali.htm`.

Kaner, C. (1997i) Status Report: New Laws that will Govern Software Quality. *Proceedings of the 15th Annual Pacific Northwest Software Quality Conference*, Portland, OR, October 28, 1997, p. 269.

Kaner, C. (1997j) Restricting Competition in the Software Industry: Impact of the Pending Revisions to the Uniform Commercial Code. Proceedings of Ralph Nader's conference, *Appraising Microsoft*, Washington, DC, November, 14, 1997. Available at `www.badsoftware.com/nader.htm`.

Kaner, C. (1997k) Legal Issues Related to Software Quality, *Proceedings of the Seventh International Conference on Software Quality.* Montgomery, AL, October 8, 1997, p. 2. An expanded version appears in *Software Quality*, #2, 1997-98, p.1.

Kaner, C. (1998) Speedbump on the information superhighway: The insecurity of digital signatures. *UCC Bulletin*, February *in press*.

Kaner, C., J. Falk, & H.Q. Nguyen (1993). *Testing Computer Software*, 2nd Edition, International Thomson Computer Press.

Kaner, C. & R. Gomulkiewicz (1997) Proposal on Warranty of Merchantability. (unpublished.) Circulated at the *Meeting of the NCCUSL Article 2B Drafting Committee*, Cincinnati, OH, May 30-June 1 1997.

Kaner, C. & B. Lawrence (1997). UCC Changes Pose Problems for Developers. *IEEE Software*, *March/April*, 139-142.

Kaner, C., B. Lawrence, & R. Johnson (1998). Splat! Requirements Bugs on the Information Superhighway. Software QA Magazine, *in press*.

Kaner, C. & T. Paglia, (1997) Letter to American Law Institute outlining the consumer community's priorities for its Executive Council meeting, December, 1997. (unpublished.) Available at `www.badsoftware.com/alidec.htm`.

Kaner, C. & D. Pels (1996). User documentation testing: Ignore at your own risk. *Customer Care*, *7*, #4, 7-8.

Kaner, C. & D. Pels (1997). Article 2B and Software Customer Dissatisfaction. (unpublished.) *Meeting of the National Conference of Commissioners on Uniform State Laws' Article 2B Drafting Committee*, Cincinnati, OH, May 30, 1997. A shorter version of this paper, for the software community, was published as Software Customer Dissatisfaction, *Software QA*, *4*, #3, 24. Available at `www.badsoftware.com/stats.htm`.

Kaner, C. & D. Pels (1998) *Bad Software: A Consumer Protection Guide.* (manuscript in progress.) Chapter 1 of this book is posted at `www.badsoftware.com`. This is the chapter that explains the application of the perfect tender rule to software.

Karjala, D.S. (1997) Federal Preemption of Shrinkwrap and On-Line Licenses, *University of Dayton Law Review*, *in press*.

Lande, R.H. & S.M. Sobin (1996) Reverse Engineering of Computer Software: The Antitrust Issues. *Harvard Journal of Law & Technology*, *9*, #2, p. 237.

Lemley, M.A. (1995) Intellectual Property and Shrinkwrap Licenses, *Southern California Law Review*, *68*, p. 1239.

MAI Systems Corp. v. Peak Computer, Inc. (1993) *Federal Reporter, Second Series*, vol. 991, p. 511 (United States Court of Appeals, 9th Circuit).

National Conference of Commissioners on Uniform State Laws (1997) *Uniform Commercial Code Article 2B, Law of Licensing*. Draft of September 22, 1997. Available at www.law.upenn.edu/bll/ulc/ulc.htm.

ProCD, Inc. v. Zeidenberg (1996) *Federal Reporter, Third Series*, vol. 86, p. 1447 (United States Court of Appeals, 7th Circuit).

Rice, D. (1997) Digital Information as Property and Product: UCC Article 2B. University of Dayton Law Review, *in press*.

Rodau, A. (1986) “Computer Software: Does Article 2 of the Uniform Commercial Code Apply?” *Emory Law Journal, 35*, p. 853.

Ritchie Enterprises v. Honeywell Bull, Inc. (1990) *Federal Supplement*, volume *730*, p. 1041 (United States District Court, Kansas).

Sega Enterprises, Ltd. v. Accolade, Inc. (1992) *Federal Reporter, Second Series,* volume 977*, p. 1510 (Ninth Circuit Court of Appeals).*

Smedinghoff, T.J. (1993). *The SPA Guide to Contracts and the Legal Protection of Software*. Software Publishers Association.

Software Publishers Association (1995) *1995 Technical Support Survey Report.* Software Publishers Association.

Step-Saver Data Systems, Inc. v. Wyse Technology and The Software Link, Inc., (1991) *Federal Reporter, Second Series*, volume 939, p. 91 (United States Court of Appeals, 3rd Circuit).

White, J.J. & R.S. Summers (1995), *Uniform Commercial Code* (4th Edition), Practitioner Treatise Series, West Publishing Corp.