LEGAL ISSUES RELATED TO SOFTWARE QUALITY

Cem Kaner, J.D., Ph.D.
Law Office of Cem Kaner
Santa Clara, California, USA

Keynote Address
Seventh International Conference on Software Quality
American Society for Quality, Software Division
Montgomery, AL, October 8, 1997.

Abstract

This talk sketches the field of software-quality-related liability. I will briefly outline many legal theories under which a maker of defective products or a provider of defective services can be sued. Then I'll focus on three specific issues. (1) Negligence in development or support of software products; (2) Professional negligence in software-related (including testing) services; (3) New legislation to govern all contracts for software development, sale, licensing, and support.

This 45-minute talk barely scratches the surface. Occasionally, I teach a once-over-lightly survey course on this topic--we squeeze the material into 8 hours and students complain that we needed more time. Even squeezing the 340-page contract law into one day is a big challenge. Between October 1, 1997 and January 1, 1998, you can download supplementary materials from my law firm's (opening soon) web site at `www.badsoftware.com`. If that site isn't open yet, get the materials from my software consulting site, `www.kaner.com`. The materials include about 200 slides, including all of the ones used in today's talk, plus several original source materials (court cases, links to draft statutes, etc.), plus any paper of mine that I reference in this talk.

Introductory Note

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, 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).

Deming's view is popular. This theme has been promoted in expensive publicity campaigns. For those who share Deming's view, and think (as I've been told loudly at some conferences) that legal issues have no place in a discussion of software quality, please suspend your judgment for an hour. Whether you like it or not, we work in a world that has several laws that govern our products and services. There's value in understanding the ground rules.

Customers Have Genuine Problems

There is a myth that most business-related lawsuits are the frivolous creation of evil plaintiffs lawyers. Supporting this has been a wave of statistics that appear to show that customer lawsuits against businesses have skyrocketed. For example, the following data from the *1994 Annual Report of the Judicial Council of California* to the Governor and Legislature have been used in political campaigns. (1983-84 is the first of the 10 years in this study. Superior Court filings usually involve cases of $25,000 or greater.)

Exhibit 1—Superior Court Civil Filings, 1983-1993		
1983-84	1992-93	**increase**
561,916	684,070	**122,154 cases (22%)**

At first glance, this looks *like* a litigation explosion. But look again.

Exhibit 2—A Closer Look at the Filing Statistics		
1983-84	1992-93	**increase**
561,916	684,070	**122,154 (22%)**
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%)***

Despite California's increase in population, there are fewer business, consumer protection, and personal injury suits, not more. And there are dramatically fewer customer-side lawyers and consumer protection advocates. 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.

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 (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 (for example, Leading Edge Computers, America OnLine, Compaq, and Corel).

There is significant customer dissatisfaction. Please see Kaner & Pels (1997a, 1997b) for extensive footnoting to our sources.

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. By 1996, 34% of American households had 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 ten 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. He spent $218 on support and his problems were apparently 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 examples. For example, in *Johnson v. General Motors*, a child's death was caused by defective software in a PROM that controlled a truck's fuel injectors. 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.

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, I think 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. I defined most of these theories, with examples, in Kaner, Falk, & Nguyen (1993). Here's a quick look at theories under which a software developer can be sued:

- ***Criminal:*** The government sues the company for committing a criminal act, such as intentionally loading a virus on your computer or otherwise tampering with your computer. For example, several years ago, Vault Corp. announced plans to release a new copy protection program that would unleash a worm that would gradually destroy your system if you illegally (in the program's opinion) copied of the protected program. That was not clearly illegal at the time, but today such a program probably would be.
- ***Intentional Tort:*** The company did something very bad, such as deliberately loading a virus onto your computer, or stealing from you, or telling false, insulting stories about you. The government might be able to sue the company under a criminal theory. You sue the company for damages (money, to be paid to you).
- ***Strict Liability:*** A product defect caused a personal injury or property damage. In this case, we look at the product's defectiveness and behavior, without thinking about the reasonableness of the process used to develop the product. No punitive damages are available. For example, suppose that the program controlling a car's brakes crashes and soon thereafter, so does the car. In a strict liability suit, we would have to prove that the program was defective, and the defect caused the accident. In a negligence suit, we also have to ask whether the manufacturer made a reasonable effort to make the brakes safe.
- ***Negligence:*** The company has a duty to take reasonable measures to make the product safe (no personal injuries or property damage), or no more unsafe than a reasonable

customer would expect. Under the right circumstances, a company can non-negligently leave a dangerous defect in a product. In *Johnson v. General Motors,* the jury found that a bug in software controlling a light truck's fuel injector caused the truck to sometimes stall midway through an intersection after stopping at a stop sign. Johnson's truck failed, it was hit by a bigger truck, and his grandchild (a passenger) died. According to the court, GM had a replacement for the fuel injector chip but chose not to recall the trucks to fix them. The jury found GM negligent and also awarded punitive damages.

- ***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 that it was false. In the case of *Ritchie Enterprises v. Honeywell Bull,* 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. 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)
- ***Unfair or Deceptive Trade Practice:*** The company engaged in activities that have been prohibited under the unfair and deceptive practices act that your state has adopted. For example, false advertising, or falsely stating or implying that the product has been endorsed by someone, or falsely claiming that a new upgrade will be released in a few weeks, are all deceptive trade practices. You have to show that the company has repeatedly engaged in this misconduct -- we're looking for evidence of a "practice", a pattern of misconduct, not just one bad event. You can receive a refund and repayment of your attorney fees. Some states allow additional statutory damages. For example, in Texas, a successful plaintiff can collect up to three times her actual damages. This is the law under which Compaq is being sued. According to the plaintiff, the warranty stated that Compaq would not charge for calls about software defects, but its support staff told the plaintiff that he had to pay for all calls about software, whether they involved defects or not. Based on his observation of the AOL/Compaq message board and on other sources, the plaintiff alleged that Compaq was also refusing to provide free support to other people when they called about genuine software defects.
- ***Unfair Competition:*** The definition varies across states. For example, in California anyone can file an unfair competition suit, so long as they can prove that the company engaged in a pattern of illegal activity. In some other states, only a competitor can sue, and only for some narrower list of bad acts. In *Princeton Graphics v. NEC*, Princeton successfully sued NEC for claiming that its Multisync monitor (the first one) was VGA-

compatible. Princeton and NEC had the same problems with VGA, and Princeton chose not to advertise itself as VGA-compatible.

- ***FTC Enforcement:*** The Federal Trade Commission can sue companies for unfair or deceptive trade practices, unfair competition or other anti-competitive acts. Most defendants these cases without admitting liability. Recent FTC cases have been settled against Apple Computer and against the vendor of a Windows 95 optimization program that didn't provide any performance or storage benefits. Occasionally, the FTC sues over vaporware announcements that appear to be intended to mislead customers.
- ***Regulatory:*** The Food and Drug Administration, for example, requires that certain types of software be developed and tested with what the FDA considers an appropriate level of care. My understanding is that development process is important to the FDA.
- ***Breach of Contract:*** In a software transaction, the contract specifies obligations that two or more persons have to each other. (In Lawyerland, 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. The specific validity of shrink-wrapped software warranty disclaimers that were not visible to the customer until after the sale, were considered in two opinions, *Step-Saver v. Wyse Technology and The Software Link*, and *Arizona Retail v. The Software Link*. 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.

  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 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, 1995, Kaner & Pels, 1996, and the court case, *Daughtrey v. Ashe*.)

- ***Magnuson-Moss Warranty Improvement Act:*** Consumers of goods are entitled to a clear statement of the warranty's 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.

Software service providers can also be sued. A "software service provider" is a "person" that writes custom software, maintains or supports software, trains other people to use software, does software testing or certification, or enters into other contracts involving software in which a significant component of the benefit to be provided by the seller involves human labor. Service providers can be sued in many of the ways that I listed for products, above, but also for:

- ***Negligent or grossly negligent provision of services:*** The service provider has a duty of ordinary care to the customer, to make reasonable efforts to provide the service. Basically, this means that the service provider should try to do the job well and it should not make mistakes that any reasonable person would recognize as mistakes. A service provider that holds itself out as having specific expertise will have its mistakes judged by what mistakes other reasonable people with this specific expertise would have made, if they were making reasonable efforts to do a good job.
- ***Malpractice:*** This is a negligence standard, with a twist. The quality of services provided is judged against a professional standard. Mistakes or other failures in delivering service that would not have been made by an ordinary professional in the field are malpractice. The existence of professional standards makes proof of malpractice easier than proof of simple negligence. If you win a malpractice case, you'll probably get more money than in a basic service provider negligence case. There is currently no theory of computer-related malpractice (Kaner, 1996f) because we have no computer-related professions, and no generally accepted standards.

Quality Cost Analysis

Any time that you see a phrase like "reasonable efforts" or "reasonable measures" in a legal theory, you are seeing an invitation to do a cost/benefit analysis. The court will do, or will instruct the jury to do, a cost/benefit analysis if the case ever comes to trial. We are, or should be, familiar with cost/benefit thinking, under the name of "Quality Cost Analysis." I discuss this in more detail in Kaner, Falk & Nguyen (1993) and Kaner (1996a). Exhibit 3 provides software examples of quality costs.

In quality cost analysis, external failure costs reflect the company's costs, not the customer's. (For example, see Campanella, 1990). The law cares more about the customer's losses. Thus, in quality/cost analysis of the Pinto, it was cheaper for the company to ship cars that were dangerously defective. The law said yes, but these 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.

Exhibit 4 shows a quality/cost analysis for the Pinto. Exhibit 5 contrasts the external failure costs suffered by sellers and those suffered by customers.

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 • 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 • 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</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</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) = $137 million.

Exhibit 5. Contrasting Sellers' and Customers' External Failure Costs

Seller: external failure costs	*Customer: failure costs*
These are the types of costs absorbed by the seller that releases a defective product.	These are the types of costs absorbed by the customer who buys a defective product.
• Technical support calls • 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 • All other costs imposed by law	• Wasted time • Lost data • Lost business • Embarrassment • Frustrated employees quit • Demos or presentations to potential customers fail because of the software • Failure when attempting other tasks that can only be done once • Cost of replacing product • Cost of reconfiguring the system • Cost of recovery software • Cost of tech support • Injury / death

Issues for Today's Talk

1) Negligence: How do we decide whether a product was developed or tested negligently? This section summarizes factors and problems called out in Kaner, Falk, and Nguyen (1993) and Kaner (1996b, 1997g). A critical issue to keep in mind is that the plaintiff must prove that the failure to use some "best practice" *actually caused* the defect in the system.

2) Malpractice: Before calling for the professionalization of software quality advocates, please consider the problem that we need a solid basis for distinguishing unacceptable from acceptable practices. Otherwise, professional liability will be a lottery: you will be sued for practices that you consider good. Here are some examples of disagreements:

- I regard the development of detailed written test scripts as an industry worst practice.
- I am not aware of any evidence that ISO 9000-3 has resulted in improved software quality and I would not recommend it to a client as a framework for improving quality.
- I believe that the reasonable practices and standards vary substantially among the several different industries that we lump together as "software." I valued my training for the ASQ-CQE because it helped me appreciate the diversity of good practices across industries. I think the CSQE's view of the software world is erroneously monolithic.

I'm not alone in these views, but many of you will disagree with me. *That's the point of these examples.* We do not have a consensus on professional practices.

One last word of caution. If a person who is not a licensed professional identifies herself to potential clients as a professional, they can sue her for malpractice as if she were a member of that profession. If you call yourself a "quality engineer" (in California) or an "engineer" (some other states), you might discover yourself at the wrong end of an engineering malpractice suit. The suit will challenge judge and jury to figure out what the professional knowledge and standards a software quality engineer would have if there was such a profession and if it had generally accepted practices. Your lawyer would make a lot of money on this case.

3) Contract Law: The United States' state governments fund the National Conference of Commissioners on Uniform State Laws (NCCUSL) to write laws that should be the same across all states. Their best known product is the Uniform Commercial Code (UCC), most of which has been passed into law in all states. They are close to completing Article 2B, a new 340-page section for the UCC, which will govern all software-related contracts, along with most other contracts involving licensing of information (see Kaner, 1996c, for background). I've been active in this process for 19 months. There is great benefit in creating a uniform legal system for software products *and* services, that works the same way across all states.

As written today, Article 2B has significant problems. (Kaner, 1996c, 1996e, 1997a, 1997b, 1997c, 1997d, 1997f; Kaner & Lawrence, 1997). Here are examples:

- When you buy off-the-shelf software, you won't be buying a copy of the product. You'll be buying a license to use the product. Off-the-shelf products are treated in most courts as "goods" today, but will be treated as "intangibles." None of the consumer protection laws that apply specifically to sales of goods (such as the Magnuson-Moss Warranty Improvement Act) apply to licenses. The Act also gives software publishers more intellectual property rights than they have under the Copyright Act (see below).
- Software publishers will be able to disclaim warranties after the sale, in a contract that customers cannot see before the sale.
- Publishers will be able to use confidentiality clauses in their license agreements, even in licenses for off-the-shelf products that you buy mail order or in a store. Today, if you buy a book (or any mass-market product that provides information), the Copyright Act gives you rights that you don't get under licensing law. For example, you can publish a book review, telling people what you think of specific parts of the book and quoting short passages to illustrate your points. You don't have this right under Article 2B. The license can say (as a few mass-market licenses do now, probably unenforceably), that the behavior of the product is confidential and that you may not publish information (such as in a magazine review) about the operation of the product without the written permission of the software publisher.

  What possible benefit is there to the public of a law that cuts off customers' right to read detailed, critical reviews of a product they are considering buying, and also 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 several ways, it will be harder to collect damages (money) from publishers if the product is defective. (Kaner, 1996e, 1997b). You will almost always be limited to a

refund, a partial refund, a bug fix, or nothing. Under the Magnuson-Moss Act, purchasers of a defective consumer product are entitled to a repair unless this is commercially unreasonable. This goes away under 2B. Instead, publishers must try to provide bug fixes only for business customers of non-mass-market products. Damages are important external failure costs. If a company knows it risks paying damages for a bug, it will more likely fix the bug. I have repeatedly proposed a narrow damages rule (e.g. Kaner, 1997f): If a publisher knows about a bug in a product before it sells it (or if the publisher would have known if it hadn't been "grossly negligent," such as doing no testing at all) then the publisher must either fix the bug, document the bug in a way that a normal customer will understand (how to avoid it, work around it, etc.), or pay for the losses that the bug provably cost the customer, up to a maximum amount. For low-priced products, I suggest a maximum of $500 or five times the price of the software. This increases external failure costs of the product only for those cases in which the publisher chose not to manage a known risk. This has gone nowhere. The law will allow publishers to knowingly leave serious bugs in the product, without disclosure, without risk of damages unless the product injures someone or does property damage.

- There are several other problems. Article 2B will dramatically change American intellectual property rules by allowing publishers of a mass-market product to ban reverse engineering. A defect must be more serious (relative to the definition of "material breach" in the *Restatement of Contracts* and several other major national and international sources of contract law rules) to qualify as "material." A product will usually have to be materially defective for you to get a refund. And in the current draft of 2B, publishers are virtually immunized from liability for viruses shipped with their products, even if they didn't even bother to use a virus checker before shipping it.

REFERENCES

Note: I'm writing this in hotel rooms as I work the Annual Meetings of the National Conference of Commissioners of Uniform State Laws (NCCUSL) and the American Bar Association. I haven't been near my library for a while, so some reference information will be incomplete. Sorry. If you need extra information, e-mail me at kaner@kaner.com. Through January, 1998, my papers are at www.kaner.com or www.badsoftware.com.

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

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

Humphrey, W. (1997). Comments on Software Quality. (unpublished). *Annual Meeting of the National Conference of Commissioners on Uniform State Laws*, Sacramento, CA, July 30, 1997.

Kaner, C. (1997g). The Impossibility of Complete Testing. In press, *Software QA*.

Kaner, C. (1997f). Not Quite Terrible Enough Software. (unpublished). *Annual Meeting of the Software Engineering Process Group*, San Jose, CA, May 1997.

Kaner, C. (1997e). Liability for Bad Software and Support. *Support Services Conference East*, Nashville, TN, March 12, 1997.

Kaner, C. (1997d). Proposed Article 2B: Problems from the Customer's View: Part 2: List of Key Issues. *UCC Bulletin*, *February*, 1-9.

Kaner, C. (1997c). Proposed Article 2B: Problems from the Customer's View: Part 1: Underlying Issues. U*CC Bulletin*, *January*, 1-8.

Kaner, C. (1997b). Remedies Provisions of Article 2B. (unpublished). *Meeting of the NCCUSL Article 2B Drafting Committee*, Redwood City, CA, January 10-12, 1997.

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.)

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

Kaner, C. (1996f). Computer Malpractice. *Software QA*, *3*, #4, 23 et seq.

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

Kaner, C. (1996c). Uniform Commercial Code Article 2B: A new law of software quality. *Software QA*, *3*, #2, 10 et seq.

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 et seq.

Kaner, C. (1996a). Quality Cost Analysis: Benefits and Risks. *Software QA*, *3*, #1, 23 et seq.

Kaner, C. (1995) Liability for Defective Documentation. *Software QA*, *2*, #3, 8 et seq.

Kaner, C. (1988). *Testing Computer Software*, TAB Professional & Reference Books.

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

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

Kaner, C. & Pels, D. (1997b), "Software Customer Dissatisfaction", *Software QA*, *4*, #3, 24 et seq.

Kaner, C. & Pels, D. (1997a). 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.

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

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