STATUS REPORT
NEW LAWS THAT WILL GOVERN SOFTWARE QUALITY

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

Laws that will govern the technical decisions that we make are being written by lawyers who, in the main, have little appreciation of the underlying technologies they seek to govern. The primary inputs to the drafting committees come from attorneys who represent the larger companies in the industry or the main trade associations. (The *professional* associations are not sending lawyers, just the *trade* associations.)

These lawyers have a keen appreciation of commercial risks and a sense that you can manage these risks by appropriate contracting and legislation. For example, if companies can be sued for defective products and your company doesn't like facing lawsuits, you can manage the risk of lawsuits *technologically* by making better products or by advertising them more honestly. Or you can manage the risk *commercially* by drafting contracts and laws that make it harder for your customers to sue you.

- In the extreme, the effect of a technological risk management strategy is 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.
- In the extreme, the effect of a commercial risk management strategy is a system that is indifferent to quality, so long as the more powerful person or corporation in the contract is protected if the quality is bad.

Both strategies are valid and both are important in modern technology-related commerce. And both present characteristic risks. Unfortunately, the current fashion in legislation is an almost exclusive focus on commercial risk management. Part of the problem is that so few of us on the technology side are making ourselves available to explain technological risk management to these commercial lawyers. If the only tools that lawyers have are hammers, they will pass laws declaring that all non-hammers are nails.

This session looks at three clusters of legislative efforts, not from the point of view of how good the proposed laws are, but from the point of view of encouraging you to think about how you can be involved in constructive improvement to these laws, whatever your political philosophy. The three clusters include software contracting, the professionalization of software quality engineering, and electronic commerce.

The Law of Software Contracting

Attempts to create a unified law of software-related contracting are finally near completion after about ten years' work, in the American Bar Association as a model law for software licensing and then in the National Conference of Commissioners on Uniform State Laws as a multi-state-government-funded effort to create a software law that will be adopted in all 50 of the United States.

We now have a huge (340-page) proposed addition to the Uniform Commercial Code (Article 2B: The law of licensing). It is scheduled for introduction into state legislatures in September, 1998.

As far as I can tell, I was the first advocate for small customers (consumers and small businesses) to get involved in this work, and I started only 21 months ago. The law is intensely biased against customers, to the extent that Brian Lawrence and I cautioned readers of an *IEEE Software* paper that if we software development folk don't get involved, we'll have a product that may as well have been written by Dilbert's boss and lawyers who work for him.

You can read my analyses at my web site, which will probably be `www.badsoftware.com` by then. If that address still doesn't work, you can link to the site under a temporary name by going to my technical consulting site, `www.kaner.com`.

In this talk, I'll explore three issues as examples of the need for stronger technical input. I think that you'll see these as problems no matter what you think of the overall philosophical approach of Article 2B:

Liability for viruses

Article 2B (Section 2B-313, *Electronic Viruses*) imposes a duty on software publishers to exercise reasonable care to keep viruses off the disks they sell. The statute then defines the duty. Publishers are required to use *one* virus checker to check for *known* viruses. Publishers of software that costs $500 or more, that is sold under a site license, or that is delivered over the Internet are not required to use a virus checker as long as their license agreement (the one you see after the sale) says that they didn't necessarily test for viruses. The statute also requires customers to check software that they buy for viruses. The customer who doesn't run the test can't collect any damages if her system is trashed by a virus on the publisher's disk. The customer who does check for viruses and finds one, or who gets a virus from the publisher's product after being prevented from checking for viruses by some characteristic of the publisher's product or instructions, is entitled to a refund (on return of the product) if the publisher's product does carry a virus.

1. To my eyes, the standard of care (one virus checker, known viruses) is absurd. What is "standard practice" and how does it vary across platforms?
2. To my eyes, the concern expressed by publishers that they can't control what happens in manufacturing or in delivery over the Net is absurd. Am I wrong that we can safely control and check against manufacturing masters today? What about secure delivery over the Net? Where are we on this and how practical is it for a small publisher? Should we have a blanket exemption for publishers who posted a clean product but the product as delivered (after interception) was virused? Or should we plan for more secure delivery systems?
3. Do we have quotable examples that we can provide to the drafters as models of what goes right or wrong?

Embedded Software

Article 2B leaves to Article 2 (the Law of Sales) all issues involving quality of embedded software. Article 2 provides slightly stricter quality standards than 2B, and I have often been told that embedded software is, on average, developed under looser standards than shrink-wrapped software. If so, embedded software manufacturers will want to move their products over, so that they are covered by 2B, not 2.

How do we tell the difference between embedded and non-embedded software? I don't know. How, then, should this statute guide judges in deciding on a case-by-case basis whether the trial should proceed under Article 2B or Article 2? I don't know. I'm preparing a separate paper on this, for distribution to the Article 2B and 2 drafting committees. The first version is based on extended discussion in the `swtest-discuss` mailing list. That draft may then circulate to `comp.software.testing` and `comp.software-eng` newsgroups for additional discussion. Alternative memos to guide the committee will be very helpful.

Here is Article 2B's current attempt to distinguish embedded from non-embedded. Embedded software includes:

```
a sale or lease of a copy of a computer program that was not developed
specifically for a particular transaction and which is embedded in
goods other than a copy of the program or an information processing
```

machine, unless the program was the subject of a separate license with the buyer or lessee.

Definition of defect

Under most circumstances, a defect in a product has to be very serious ("material breach of contract") or the customer will not be entitled to a refund for the defect. Initially, the statute defined a material breach almost exclusively in publishers' terms, using examples like failure to make payments for the software or using the product in a way that interferes with the publisher's intellectual property rights. I pushed heavily for a more customer-focused and product-focused definition (Kaner, 1997a) and made minor headway.

SECTION 2B-108. BREACH OF CONTRACT.

(a) Whether a party is in breach of contract is determined by the contract. Breach of contract includes a party's failure to perform an obligation in a timely manner, repudiation of a contract, or exceeding a contractual limitation on the use of information.

(b) A breach of contract is material if the contact so provides. In the absence of an express contractual term, a breach is material if the circumstances, including the language of the agreement, reasonable expectations of the parties, standards and practices of the trade or industry, and character of the breach, indicate that:

(1) the breach caused or may cause substantial harm to the aggrieved party including imposing costs that significantly exceed the contract value; or

(2) the breach will substantially deprive the aggrieved party of a benefit it reasonably expected under the contract.

(c) A material breach of contract occurs if the cumulative effect of nonmaterial breaches by the same party satisfies the standards for materiality.

(d) If there is a breach of contract, whether or not material, the aggrieved party is entitled to the remedies provided for in the agreement and this article.

Uniform Law Source: *Restatement (Second) Contracts* § 241.

The reference to the Restatement of Contracts is interesting because it uses a different set of criteria that I prefer. As cited in the Article 2B draft comments following 2B-108:

The *Restatement (Second) of Contracts* lists five circumstances as significant:

(1) the extent to which the injured party will be deprived of the benefit he or she reasonably expected;

(2) the extent to which the injured party can be adequately compensated for the benefit of which he will be deprived;

(3) the extent to which the party failing to perform or to offer to perform will suffer forfeiture;

(4) the likelihood that the party failing to perform or to offer to perform will cure the failure, taking into account all the circumstances, including any reasonable assurances; and

(5) the extent to which the behavior of the party failing to perform or to offer to perform comports with standards of good

faith and fair dealing. *Restatement (Second) of Contracts* § 241 (1981).

Let me illustrate the difference. Suppose that you buy BugWare 2.0 from ShipIt Software. BugWare is a shrink-wrapped product, costing $100. They sold 100,000 copies. The program has several annoyances, but nothing so serious that it destroys your hard disk. You are dissatisfied with the product, you can prove that it has several defects, and you want a refund. Can you get one?

Under Article 2B, probably not. The "contract" is the shrink-wrapped piece of paper that comes inside the box. (Article 2B will validate the terms of these "licenses," including terms that would not be valid under current law.) So this contract, written by the publisher, will probably not define "material breach" in a way that includes misbehavior of the program. So, you have to prove that you've been substantially harmed or that the program doesn't provide the benefits that you reasonably expected. Maybe you can make this argument, maybe not.

Under the *Restatement of Contracts*, which 2B lists as its source for the material breach standard, you'd have an easier argument.

1. You can still prove that you've been deprived of benefit, but this is just one factor.
2. The next issue is important and totally missing from 2B. You are entitled to compensation for the benefit of which you're being deprived (yes, yes, Article 2B says it gives this too, *but the publisher can charge you $35 per call to ask for a $10 partial refund*.) Under the Restatement, if you have no realistic means of being reasonably compensated, that's a factor in determining that you're entitled to a full refund. Under 2B, it's not.
3. The Restatement's third factor, forfeiture, favors the mass-market customer. You suffer a forfeiture if I string you along for a year on a project and then at the end of the year I say, "Sorry, I don't like what you're doing, I'm not going to pay you. Go away." My rejection of your product is a rejection of all compensation for the work you did in developing the product. This issue simply doesn't arise in mass-market software. My rejection of one copy of a mass-market product involves just one transaction of many. You make or lose your investment based on a more general level of success in the marketplace, not on my one purchase.
4. The Restatement's fourth factor is also irrelevant in Article 2B but not to customers. If the publisher won't give you a bug fix for a mass-market product, 2B says too bad (publishers of non-mass-market products have a duty to attempt to "cure" defects, but 2B drops the duty for mass-market products). The Restatement says that the court should take into account the publisher's willingness to give a bug fix in determining whether the refund is called for or not.
5. And finally, the Restatement asks the judge to consider the publisher's good faith. You get to bring in all those advertisements and argue that even if they aren't enforceable warranties, they are evidence that the publisher is playing fast and loose with facts. You get to bring in any evidence that the publisher knew about the bugs you're complaining about before it sold you the product. Etc.

The mass-market customer is treated reasonably well under the Restatement, and badly under 2B.

The buyer of custom software should probably be treated differently. If you develop custom code for a wild and crazy customer, you might end up including some genuine atrocities in the product's user interface. (*You really want us to design it this way? Are you really, really sure? OK, you're the customer.)* If you follow the customer's instructions, the customer shouldn't be able to demand a refund if the instructions were stupid.

I tried to sort this out in Kaner (1997a), suggesting different ways of classifying program misbehavior that depended on the relationship between the developer, the customer, and the specification (if one existed).

I think that we need something to help guide judges determine what is a serious defect, under what circumstances, or we'll see random standards across states and countries. Issues of the amount of harm suffered by the customer and the extent to which the customer is deprived of benefit are so vague in the law that it will take years and years of litigation before *some* of us finally understand what they mean (the rest of us lawyers will still *not* understand).

My proposal went like this:

> ***Reflecting the Relationships Between Licensor and Licensee***
>
> The licensor is analogous to the seller in a traditional sale. Under Article 2B, what is sold in the typical software transaction is a license to use the software, rather than a copy of the software. The licensee is the customer, who buys the license.
>
> I think there are four common classes of software transaction:
>
> 1. ***The customer writes the specifications and requirements and asks the developer to write a program as specified.***
>
>   In my view, the software developer meets its obligations if it writes a program that meets the specifications. If the specs say "2+2=5" then the program does not breach the software contract if it generates the wrong answer (5) whenever it adds 2+2. *Let the specifier beware.*
>
> 2. ***The developer writes the specifications and requirements, in preparation for custom development, but the customer is sophisticated.***
>
>   If the customer is a computer expert, then it is able to review the requirements and specifications just as well as the staff of the developer. The customer is also probably in a better position to understand its own requirements than the developer. Therefore it is reasonable to hold this customer accountable for reviewing the specifications.
>
>   Article 2 of the current Uniform Commercial Code defines a "merchant" as follows:
>
>    ```
>    2-104 (1) "Merchant" means a person who deals in goods of the kind
>    or otherwise by his occupation holds himself out as having
>    knowledge or skill peculiar to the practices or goods involved in
>    the transaction or to whom such knowledge or skill may be
>    attributed by his employment of an agent or broker or other
>    intermediary who by his occupation holds himself out as having
>    such knowledge or skill.
>
>    2-104(3) "Between merchants" means in any transaction with respect
>    to which both parties are chargeable with the knowledge or skill
>    of merchants.
>    ```
>
>   Article 2B uses essentially the same definition:
>
>    ```
>    2B-102 (26) "Merchant" means a person that deals in information of
>    the kind, a person that by occupation purports to have knowledge
>    or skill peculiar to the practices or information involved in the
>    transaction, or a person to which knowledge or skill may be
>    attributed by the person's employment of an agent or broker or
>    other intermediary that purports to have the knowledge or skill.
>    ```
>
>   When Microsoft buys Apple computers, it is a merchant. When a large, local hospital buys a bunch of Apples, it might be a big business, but it is not a merchant.
>
> 3. ***The developer writes the specifications and requirements, in preparation for custom development, but the customer is not sophisticated.***
>
>   When doctors, dentists, insurance brokers, small grocery store owners, and other small business people buy software, they have no clue how to specify the software, no clue how to evaluate a requirements document, no clue how to test the software, and no clue how to cost-

effectively find a consultant who has these skills. In common computer parlance, these customers are called *Clueless*.

In UCC parlance, these customers are *non-merchants*.

These customers rely on the knowledge and experience of the developer. If the developer makes errors in defining the requirements or the specifications, which result in serious errors in the operation of the program, this is the developer's bug, not the customer's.

4. ***The developer writes a mass-market product. The customer has no input into the design or development of the product.***

   In the mass-market case, design errors belong to the developer, not the customer. Internal specifications that were used during development are largely irrelevant to the customer. The end product works in a reasonable way, as advertised and as documented, or it does not.

The Proposed Statutory Language

This proposal modifies Section 2B-108 of Article 2B. I am a novice at drafting statutes. The language will be cleaned up during the Article 2B review process. The proposal expresses my sense of the fundamental differences between these transactions.

Proposed SECTION 2B-108. BREACH OF CONTRACT.

(a) Whether a party is in breach of contract is determined by the terms of the agreement and by this article. Breach occurs if a party fails to perform an obligation timely or exceeds a contractual limitation.

(b) A breach of contract is material if the contract so provides. In the absence of express contractual terms, a breach is material if the circumstances, including the language of the agreement, expectations of the parties, and character of the breach, indicate that the breach caused or may cause substantial harm to the interests of the aggrieved party, that the injured party will be substantially deprived of the benefit it reasonably expected under the contract, or that the breach meets the conditions of subsection (c), (d), (e), (f) or (g).

(c) If the licensee provides the specification documents that are incorporated in the contract, then a breach is material if:

> **(i)** the software fails to perform in conformance with and in the time required by express performance standards or specifications;
>
> **(ii)** the software fails to perform in conformance with the specifications and this failure either deprives the licensee of a significant benefit of the product or results in costs to the licensee that exceed the price paid for the software;
>
> **(iii)** where the specifications are silent, the software's performance is unreasonable and it results in costs to the licensee that exceed the price paid for the software. The licensee has the burden of demonstrating that a reasonable licensor would consider the software's performance to be unreasonable.

(d) If the contract is between merchants, and it contains specification documents, then a breach is material if:

> **(i)** the software fails to perform in conformance with and in the time required by express performance standards or specifications;

(ii) the software fails to perform in conformance with the specifications and this failure either deprives the licensee of a significant benefit of the product or results in costs to the licensee that exceed the price paid for the software;

(iii) where the specifications are silent, the software's performance is unreasonable and it results in costs to the licensee that exceed the price paid for the software. The licensee has the burden of demonstrating that a reasonable licensor would consider the software's performance to be unreasonable.

(e) If the contract is not between merchants, and the licensor provides the specification documents that are incorporated in the contract, then a breach is material if:

(i) the software fails to perform in conformance with and in the time required by express performance standards or specifications;

(ii) the software fails to perform in conformance with the specifications and this failure either deprives the licensee of a significant benefit of the product or results in costs to the licensee that exceed the price paid for the software;

(iii) the software fails to perform in conformance with the end user documentation or other documentation delivered to the licensee and this failure either deprives the licensee of a significant benefit of the product or results in costs to the licensee that exceed the price paid for the software;

(iv) where the specifications and other documentation are silent, the software's performance is unreasonable and as a result, it either deprives the licensee of a significant benefit of the product or it results in costs to the customer that exceed the price paid for the software. The licensee has the burden of demonstrating that a reasonable person would consider the software's performance to be unreasonable.

(f) If the contract is for a mass-market license, then a breach is material if:

(i) the software fails to perform in conformance with the end user documentation or other documentation delivered to the licensee and this failure either deprives the licensee of a significant benefit of the product or results in costs to the customer that exceed the price paid for the software;

(ii) where the documentation is silent, the software's performance is unreasonable and as a result, it either deprives the licensee of a significant benefit of the product or it results in costs to the licensee that exceed the price paid for the software. The licensee has the burden of demonstrating that a reasonable person would consider the software's performance to be unreasonable.

(g) A material breach of contract occurs if the cumulative effect of nonmaterial breaches by the same party satisfies the standards for materiality.

If there is a breach of contract, whether or not material, the aggrieved party is entitled to the remedies provided for in this article and the agreement.

Whether you like this particular proposal or not, the point to recognize here is that if we want the courts to operate from a sound definition of software defect, we have to write it.

Professionalizing Software Development and Testing

Efforts to professionalize software developers and software quality advocates come and go. By "professionalize" I mean that the government will declare us members of a "profession," require us to go to school and take specific courses, require us to pass licensing exams, and subject us to a risk of malpractice lawsuits if we do something that someone thinks is incompetent. For example, there was an attempt to require all software developers in New Jersey to be licensed a few years ago. The bill failed, but another one like it will come up one of these days.

I sat on an ASQC committee that did the final drafting of the Body of Knowledge for the CSQE certification. Some of the people involved in that process expressed the belief that this certification was a step along the road to professionalization of Software Quality Engineers. My impression is that this is not an uncommon view in this community. This view rests on two interesting assumptions: that there *is* such a thing as Software Quality Engineering and that there *is* a shared (or shareable) body of knowledge that reflects a genuinely shared understanding of theory and practice in this field.

I'm not going to challenge those assumptions here (but see Kaner, 1996a,b for some discussion). But I do want to raise a little red flag.

As far as I know, there are no serious legislative efforts in the works today to professionalize software developers or software quality advocates. But as the Year 2000 hype accelerates, the press will pay even more attention to expensive software related problems. We already see serious bugs getting widespread press coverage. And the collapse of some large, expensive, software development projects for state and federal governments has gotten its share of bad press too. As the Year 2000 publicity--about the huge programming effort and national expense being required to fix a few bugs--gets even more public awareness, some bright politician will decide that it would be a great idea to license software developers and testers and make them liable for malpractice. If the publicity associated with programming failures is bad enough, the public clamor for politicians to "do something" to prevent this from ever happening again will be significant. And if *we* push the idea of professionalizing the field, the politicians might give us what we ask for, whether it makes sense or not.

As leading software quality advocates, I urge you to think carefully about your role in this. If we "professionalize" what will the effect be? Are we going to improve the technological management of the risks of bad software and bad software contracting this way? Or will we merely create a system that provides for commercial allocation of risks *onto* individuals (us) and away from the corporations that hire or contract with us and that make many of the demands that get the projects or products into trouble in the first place?

Electronic Commerce

I'm still learning about electronic commerce and encryption technology. I'll have more details at PNSQC.

The core problem is the need to be able to make binding, enforceable agreements in a worldwide marketplace that does business electronically. Digital signatures are being proposed as a big part of the solution for this, in state legislatures, among the United States and other federal governments, and at UNCITRAL (United Nations Commission on International Trade Relations).

You can digitally sign something by using your private key (part of the public/private pair in public key encryption). The digital signature laws all hope to make this signature binding. The concern that I have involves fraudulent use of the key by someone who has somehow obtained a copy of it from the legitimate owner. My concern is based on study of Article 2B's electronic contracting rules and the

American Bar Association's Science & Technology Section's *Digital Signature Guidelines*, attendance at meetings discussing the guidelines, and monitoring of the e-mail correspondence on a mailing list associated with the United States' Department of State's Advisory Committee Study Group on Electronic Commerce. Some state laws have been passed, which I have not yet read. (I don't get paid for any of this work and I pay my own expenses, which are considerable. That limits the time that I have available for going through this material.)

Suppose that someone gets access to your computer, that your key is stored on your computer, and that they are able to copy enough of your system that they can crack the (relatively simple, because a human has to type it) password that you use to access your key. They can now pretend to be you, and they can run up transactions in your name. The sellers that they contact will check the validity of your key by contacting a Certification Authority (a private company that you register with). The Contracting Authority (CA) will say that the key that has been used is indeed the key associated with your name, and that you have not yet contacted them to repudiate (cancel) the key. The seller then ships merchandise to the crook, thinking that person is you.

If you used a MasterCard (or any of the other main credit cards) instead of a digital signature, you'd enjoy several protections. First, your liability for fraudulent use of the card is limited. MasterCard acts as an insurer. Second, MasterCard imposes several additional security restrictions, such as refusing to authorize delivery of some merchandise to any address other than your billing address, such as requiring merchants to phone MasterCard for authorization, such as limiting your credit limit so that your card can only run up a limited amount of credit, and so on.

Unfortunately, if you use a digital signature, you have none of those safeguards. There is no automatic limit on your losses from theft. There is also no mechanism for you to require sellers to contact you for authorization of large purchases, to set the equivalent of a credit limit that suspends the encryption key's validity as soon as the total spent while using it this month exceeds a limit, to restrict keys to specific uses (such as, only accept this as a non-monetary signature on court documents, or only accept this for purchases under $100 or for purchases that are of business-related merchandise) or to require sellers to ship only to your home address. I *think* that several of these issues could be solved technologically, by generalizing the format of the Certification Authority's standard verification record, in a way that allows key owners to specify terms and conditions under which sellers can rely on their key, and to include those as part of the record that is sent to the prospective seller. **It *appears* as though there are often hundreds of pages of terms and conditions that get associated with a digital signature, protecting the seller and the Certification Authority and, *apparently*, a host of other people--<u>everybody but the customer who will be subject to *infinite liability*</u> (lose the house, the dog, the bank account, go bankrupt, the works) <u>if someone obtains access to their key.</u>**

I have several examples of ways in which companies gain access to consumers' hard disks under circumstances that look like normal, harmless business transactions. The consumer would have no idea that her key had been copied, and would likely not discover that it was being used fraudulently for days or weeks.

So, I *think* that I'm looking at a design bug in this system that provides risk management to everyone but the person who needs it most (the customer who will be bankrupted if her key is compromised). Design bugs like this cost more to fix the longer they're allowed to stay in the system. There's benefit in making noise about these early, before the system is fully implemented around the world.

Someone recently asked me, "Who died and made you a software legislation tester?" I didn't interpret this as a friendly question, under the circumstances, but I liked the title. *Software Legislation Tester*. The world needs a few more of us. Many of you are capable of this. What's our next step?

References

American Bar Association, Section of Science & Technology, *Digital Signature Guidelines.* First sections available free at www.abanet.org

Kaner, C. (1996a) "Software Negligence and testing coverage", *Proceedings of STAR 96* (Fifth International Conference on Software Testing, Analysis, and Review), Orlando, FL, May 16, 1996, p. 313. Available at `www.kaner.com` and/or at `www.badsoftware.com`.

Kaner, C. (1996b) "Computer Malpractice", *Software QA,* Volume 3, #4, p. 23. Available at `www.kaner.com` and/or at `www.badsoftware.com`.

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.kaner.com` and/or at `www.badsoftware.com`.

Kaner, C. & Lawrence, B. (1997, March/April) "UCC changes pose problems for developers", *IEEE Software*, available at `www.computer.org`.

National Conference of Commissioners on Uniform State Laws (1997) *Draft: Uniform Commercial Code Article 2B – Licenses: With Prefatory Note and Comments.* Available at `www.law.upenn.edu/library/ulc/ulc.htm` in the Article 2B section, under the name *1997 Annual Meeting*. By the time you read this, a later version will also be available and posted at that site or at `www.law.uh.edu`.

STATUS REPORT
NEW LAWS THAT WILL GOVERN SOFTWARE QUALITY

Abstract

The law of software quality is a complex area including laws governing crimes, negligence, fraud, deceptive advertising, unfair trade practices, unfair competition, anti-competitive practices, safety regulations, breach of contract, and various other areas. Legislators are now focusing more directly on computer-specific and software specific issues. One example is a proposed set of revisions to the Uniform Commercial Code that brings a wide range of issues into a single huge (340 page) statute that will govern almost all contracts for the development, sale, licensing and support of software. Another example is the wave of digital signature laws, laws that validate a computer-mediated technology in order to make possible a new way of doing computer-mediated business. These laws determine baseline standards for our products and services. As advocates of software quality, we should have an interest in anything that sets minimum software-related quality standards. Along with an interest, we can play an important role, because we understand the technical issues and the technological risk management issues much better than the attorneys who are trying to draft these laws. Our expertise has value for the legislative drafting process, and the absence of our expertise has resulted in draft laws that are weaker than they could be.

About Cem Kaner

I teach software testing courses, consult to software companies on testing, documentation and software project management, and practice law within the software community. I work to improve software customer satisfaction and safety *and* corporate profitability.

As a software developer, I've programmed, done UI design, managed software development projects, software test groups and documentation groups, sold software, and consulted to companies to build or rebuild project teams or small departments. I'm senior author of the book, *Testing Computer Software*, and am writing *Bad Software: A Consumer Protection Guide*.

As an attorney, I typically represent individuals and small businesses, whether they are developers or customers. Most clients are involved with software or with writing. I draft and negotiate contracts and advise clients on their rights after a breach of contract or a failure of a relationship. And I file expensive bug reports. I also do extensive legislative work, participating in multi-year projects that are drafting laws to govern software contracting and software-related aspects of electronic commerce. These include *the National Conference of Commissioners on Uniform State Laws'* (NCCUSL's) drafting committee *for Article 2B of the Uniform Commercial Code (Law of Licensing)*, NCCUSL's drafting committee for a *Uniform Electronic Transactions Act*, and the *Department of State's Advisory Committee on Private International Law: Study Group on Electronic Commerce*. (I am an actively participating "observer" at the NCCUSL meetings, not one of the 300 NCCUSL Commissioners.) I've also served as a Deputy District Attorney (full-time *pro bono* assignment) and as an investigator/mediator for a consumer protection agency.

I hold a Ph.D. in Experimental Psychology, with a focus on human perception and performance. I apply this in my work in Human Factors (how to redesign systems and machines so that they work better for people.) I hold a B.A. in Arts & Sciences (Math, Philosophy), and a J.D. (law degree). I'm Certified by the American Society for Quality in Quality Engineering.