CEM KANER, J.D., Ph.D.

Law Office of Cem Kaner **kaner@kaner.com**
P.O. Box 1200 **408-244-7000 (Voice)**
Santa Clara, CA 95052 **www.badsoftware.com** **408-244-2181 (Fax)**

Y2K, How Can I Sue Thee? Oh, Let Me Count the Ways!

In press, Software QA Magazine.[1]

I'm not a Y2K expert, just a lawyer with a lot of software experience who's trying to make sense of the legal issues being raised in the land of Y2K.

The legal situation is plenty puzzling. One the one side are lawyers telling me that there will be virtually no Y2K litigation because it will be almost impossible for software customers to sue software vendors for Y2K-related bugs. On the other side are consultants and lawyers screaming that the sky has burst and "The lawyers are falling, the lawyers are falling!" Estimates of $1 trillion in legal costs are common, though I pointed out in my last article in *Software QA* on Y2K service contracts (Kaner, 1998), that this number seems wildly overblown in terms of the maximum capacity of the American legal system.

The technical situation is even more puzzling. Why are companies so slow to address this problem? Here are a few articles from Information Week: *Study: Poor Year 2000 Showing For Largest Companies* (Caldwell, 1998; "A study of financial filings by the nation's 250 largest corporations scuttles any notion that large companies have their year 2000 projects well in hand."); *Most Fortune 500s Not Working On Year 2000 Fix* (Patrizio, 1997; "Only 16 percent of Fortune 500 companies surveyed have begun to implement a full-fledged strategy to become year 2000 compliant, and only 24 percent have a detailed plan in place."); *Year 2000 Vendors Looking for Work, ITAA Survey Says* (40% consulting firms are not meeting their Y2K sales and revenue projections). If you want to get much more depressed, just go to www.infoweek.com and search on "year 2000."

It seems plausible to suppose that some companies will not be ready on January 1, 2000. So, what will happen (in legal terms) if these companies have serious problems? Probably, those companies will find themselves in a mess.

Before looking further at the legal issues, let's review some technical issues. What is this Y2K problem anyway?

The Basic Year 2000 Problem

The first set of problems are the obvious date-related errors. Simply put, many computer programs will misinterpret dates. They will treat 2000 (00) as 1900. This section provides some examples of the kinds of consequences that this has brought or will bring.

The thing to note is that, as bugs go, these look like fleas. Pests. A few of them will spread serious disease but the rest are relatively small, if annoying. Individually, they can be dealt with. The problem is that there are so many of them.

Age-related errors

For example, Granny (born in 1900) might stop receiving her pension checks because the computer decides that she is a newborn, not 100 years old.

Can't do future things

For example, some stores can't process credit cards today, because their expiration dates are 00 or 01. (*Produce Palace v. Tec-America Corp.,* 1997*)*

Another example comes from the first Y2K lawsuit, described by Warren Reid, who was a consultant in the case (Scott & Reid, 1998). A magazine publisher sold subscriptions, often five-year subscriptions, to teenagers. Y2K-related bugs made it impossible for the publisher to sell subscriptions that ended later than December, 1999. So the subscriptions became four years long, then three years, rather than five years. Teens' preferences change as they grow older, so a person who is willing to buy a five-year subscription today will probably not be willing to renew a three-year subscription to the same magazine three years from now. The magazine simply lost all that business.

Premature expirations

For example, some medication is marked "Best Before 01/01/01" or "01/01/99". An inventory control program might misinterprets the 01 or the 99 and decide that the medication is almost a century old. It will then flag to the warehouse staff that they should discard the medication. This has already been a problem for food vendors.

Another example: You might buy an extended warranty for your car. It is supposed to expire in 2002, but the auto dealer's computer might decide that it expired in 1902.

Long intervals

For example, tests of some old telephone billing software revealed that a call made at 11:59 p.m. on December 31, 1999 and ending at 12:01 a.m. on January 1, 2000 would be interpreted as having lasted a full century.

Y2K AND MAINTENANCE

The problem that concerns me more than the original Y2K bug is the side-effect of the Y2K bug fix. We know that when we fix one bug in a program, there is a significant probability that the fix will not be fully effective or it will cause something else to break. Kaner, Falk & Nguyen (1993) cite some statistical estimates of average side-effect probabilities, but many IT shops will be in worse situations.

Some companies no longer have source code for their main applications. Or the source is so badly commented, badly organized, and badly variable-named that it is impossible to figure out the program from the source. What are the chances that a bug fix to this type of code will be successful?

In his keynote address to the 10th International Software Quality Week (San Francisco, May 1997), Boris Beizer described the principles of nonlocality and nonproportionality. These are important principles for Y2K:

Nonlocality (and an example)

In the physical world, if you tweak one thing and thereby break something else, the thing that you broke will probably be located near to the thing you tweaked. In the software world, if you tweak one thing and thereby break something else, that broken thing might be anywhere in the code. This is the principle of nonlocality.

Every moderately experienced tester has experienced nonlocality the hard way. But here is a Y2K example to illustrate the point. Suppose you have a fixed-length record database, with the following fields in each record:

NAME	60 chars	LOCATION-NAME = 1
BIRTHDAY	6 chars	LOCATION-BDAY = 61
SSN	9 chars	LOCATION-SSN = 67

OTHER 100 chars LOCATION-OTHER = 76

We can create pointers to the starting point of each field within each record. For example, NAME starts at the first location of the record and so we set LOCATION-NAME = 1. The total length of each record is 176 characters. Let's create a variable, RECORD-LENGTH and set it to 176.

Now suppose that we give this database a Year 2000 fix. BIRTHDAY now gets 8 characters (4 digits for the year instead of 2). LOCATION-SSN will now be 69, because SSN starts at location 69 in each record. RECORD-LENGTH is now 178 instead of 176.

Suppose that you want to look up the Social Security Number in the third record. In the original version of the database, this was on the disk, 419 through 427 bytes from the start of the file. In the new version of the database, this field will be at locations 425 through 433. (I'm ignoring the file header, record headers, etc. These are just more constants.)

One way to calculate the location of this field is to compute it through the variables. The location of the third Social Security Number is 2 * RECORD-LENGTH + LOCATION-SSN, which is 352+67=419 (in the old file) or 356+69=425 (in the new file).

You might think that this is silly. Today, people would look up this information using a database management program or standard database management routines, rather than writing code to skip through the file one record at a time. But that's today. In the old days, data management programs were slow, buggy, hard to use, buggy, badly documented, buggy, and expensive. So many of us wrote our own access routines instead.

I did my first years of serious programming on PDP-8 computers that had between 4K and 12K of memory. We used a lot of tricks to squeeze big problems into itty-bitty places. Suppose that you were programming in 1976 (when I was) on a system like this and you had to write a critical error handling routine for this database. This routine must make an emergency write of the record in memory to the disk. You have a choice. You can compute where to save the record by looking up the values of RECORD-LENGTH and the location variables. Or you can blast your data to disk using values that you hard coded after the database design had stabilized. This is faster, requires less memory, and it will make a horrible mess if you change the structure of the database without updating this variable.

Back in the 1970's some of us sometimes rationally decided that the wisest choice under specific circumstances (such as critical error handlers) was to use hard coded values. Whether you think this could ever have been rational is up to you. The fact is, though, that we did it.

Now let's move forward in time 22 years to today. A generation ago, someone else wrote code (our database) that you have to fix for Y2K. The program was so useful that it is still in service. People have patched it over the years and the current listing of the code (if you have one) is unintelligible. So, you hire a software archaeologist to figure out how the program works and how to fix it.

The archaeologist (or her Y2K find-and-fix tools) might find the usual reads and writes to the database, but she might easily miss a critical error routine that reads and writes directly to disk without once mentioning the BIRTHDAY date field. If so, every time the error routine runs, it will corrupt the database.

Changes in one place in the code can have an impact far, far away.

Nonproportionality

Suppose that you have a ball on the table. If you push it, the ball moves. Push it gently, the ball rolls a little. Push it hard, the ball moves far and fast. Apply more force, have a bigger effect. Proportionality.

Suppose that you have a program on the table. If you make a little change, you might just have a little effect or you might have a big effect. A big change (in terms of numbers of lines of code) might have a modest impact on the program. In contrast, a one-line change took down much of the nationwide phone system a few years ago.

Little changes can have big impacts.

Integration testing

Have you ever worked on a project in which several programmers developed different parts of a system? If so, were you involved in the integration tests? Did the program pass them the first time?

During development, a programmer gets his own code working (he thinks) and then he brings his code to the larger group and tries to run it in conjunction with the rest. This attempt is the integration test.

Systems typically fail integration tests the first few times.

Some companies do incremental integration. They write some code, add it to the system, test, fix, write more code, add it, test, fix, and so on. The groups that do a new build every day or every week are doing some type of incremental integration. Others write all of their code first and then put it all into integration testing at the same time. We call this approach "big bang testing." Load everything into the system together and watch it blow up.

Think about all these companies working on Y2K upgrades to their systems. It looks to me as though the world is working on the largest integration test in history. We're building or revising a huge number of systems that (for financial systems, for example) have to work together. As we put them in service, we are testing (intentionally or uh-oh, the hard way) our system's compatibility with the other systems that are in service.

Because so many companies are delaying their Y2K efforts, we are soon to be facing the biggest big bang integration test in history.

Let's add some more fun.

Definitions of Y2K-compliant vary. Several definitions of Y2K compliance explicitly state the assumption that all inputs to the program will provide properly formatted date data. (See, for example, the Information Technology Association of America definition and the IEEE (1997) draft standard and the United States Federal Acquisition Regulation.) The program can be "compliant" even if it can't handle bad input data. Other definitions require the program to cope with bad data. (See, for example, New York State Government, 1997), So, we have conflicting or incomplete requirements, and thus we have more opportunity for the big bang to be a really big kaboom.

TECHNICAL RISK: SUMMARY

Many date-related Y2K problems might be analogous to fleas, but in the process of nuking these bugs, we might be creating Godzilla.

No one knows how many failures we'll see in 2000, nor how serious they'll be. Our collective software maintenance process might yield Godzilla or it might merely yield a few coderoaches. Only time will tell.

WHY ARE THERE LAWSUITS?

A lawsuit arises out of a failure to live up to a responsibility (duty). Duties arise:

- in contracts
- in the laws of public safety (such as criminal law and products liability law)
- in the laws of fraud and deceptive practices
- in the laws of fiduciary (caretaker) responsibility (such as the laws that define the responsibilities of the Board of Directors to a company's shareholders.)

The critical thing to recognize is that if there is no breach of a duty, there are no grounds for a lawsuit.

When lawyers analyze a situation to determine what lawsuits are likely or possible, their typical analysis involves a few general questions:

- Who are the potential parties (people in the lawsuit)?
- What are their duties?

- Which duties might be breached due to a Y2K failure?
- What actions can aggrieved parties take as a result of a Y2K breach?

The problem with Y2K is this: Any relationship or duty that can be interfered with by a Y2K problem can become the subject of a lawsuit.

Because we don't know how many relationships or duties are likely to be interfered with, we don't know how to estimate how many lawsuits there can be or will be.

A LOOK AT THE RELATIONSHIPS

Insert Figure 1 about here

Any relationship that is polluted by a Y2K bug could turn into a Y2K lawsuit.

Businesses have a lot of relationships with people and with other organizations. Figure 1 illustrates four classes of relationships. Here are a few notes on some of the relationships.

Suppliers of Goods, Data, and Services

Your company might wish to sue a supplier who has caused you Y2K grief. Some examples of suppliers include software suppliers, software intellectual property owners, software service suppliers, outsourcing vendors, Y2K remediation services vendors, data suppliers, landlords, facilities managers, health services providers, insurers, suppliers of diagnostics and other tools, hardware vendors for computers and non-computer hardware, legal and financial service providers, utilities, shippers, and transportation companies. Here are a few examples.

Software suppliers and intellectual property owners

If your supplier's code has a Y2K bug, you might want to sue them, but you might be surprised. If you have a signed contract with them, it probably has a sharply limited warranty, with limited damages. (In a mass-market product that hides these terms inside the box, these limits might not be enforced by a court. But signed contracts are usually enforceable.) Further, you might not be able to collect much even if there is a warranty, because you have a duty to mitigate damages--once you are aware of a problem, it is your duty to minimize your losses from it. You can only sue for what you couldn't avoid. Also, you have a limited time in which to sue a supplier, probably between one and four years after delivery of the software. That's why we see suits against companies like Intuit and Symantec today. If the plaintiffs waited until 2000, it would probably be too late to sue. There might be very few successful suits against software publishers.

Remarkably, these software suppliers might be able to sue you (the customer). Suppose that you buy software that has Y2K defects and you decide that you want to fix it. Your software license might prevent you from fixing it. Kaner & Pels (1998) discuss licensing terms that restrict customers from fixing bugs themselves or finding third party maintenance organizations to fix bugs in software. Nuara, Benard, & Rydberg (1998) provide an excellent discussion. Some lawyers are now staking out the position that these terms are fully enforceable as long as the original vendor is willing to fix the bugs in its software. Of course, you have to be willing to pay them to fix their bugs. Now Y2K becomes a revenue generator, rewarding companies that published defective code. Not surprisingly, the upcoming amendments to the Uniform Commercial Code (Article 2B, see NCCUSL, 1998) will reinforce the legal basis of these controversial restrictions. Also, not surprisingly, there will be a major effort to pass this bill in 1999. (Article 2B was scheduled for 1998. It took a huge amount of work, but several of us have succeeded in convincing the organizations supervising the drafting of Article 2B to continue revising 2B for another year before submitting it to state legislatures. For additional information, see Gillmor (1998), Gleick (1998), or my website, www.kaner.com.)

Software consultants

I wrote about the legal risks of software consultants, especially Y2K-related consultants, in Kaner (1998).

Outsourcers

Suppose that your company outsourced its data processing. The outsourcer signed a contract to do the data processing. Now it's saying that "you" have a Y2K problem because its software (which it might have gotten originally from you as part of the outsourcing arrangement) is not Y2K ready. Depending on your contract, the outsourcer might be fully responsible to process your data accurately, without interruption, despite the Y2K problem.

Landlords and facilities management

I understand that some elevators will fail on January 1, 2000. Apparently, these have a built-in clock. The elevator uses it to track how long it has been since it was serviced. When the clock's year switches to from 99 to 00, the elevator will decide that it hasn't been serviced for a century. It will close its doors and descend to the bottom floor, waiting for the repair technician to show up, do regular maintenance, and reset the service clock. I'm not sure how often this will happen--other smart elevators use a 365 day counter rather than a clock tied to the actual date.

But suppose that you end up in one of these elevators. How long will it take before the technician arrives? There might be a lot of elevators in your city, so the technician might be pretty busy.

And here's hoping that your technician is local. If she has to fly to your city, she might not be able to make it for a while. My understanding is that some of the FAA (Federal Aviation Authority) computers are not being updated and that at least one airline is talking about not flying on January 1, 2000. You might be in that elevator for a long time. When you finally get out, you might sue everything and everyone in sight.

Insurance

Insurance companies are scared and upset. They are not happy that their customer companies are postponing and postponing and postponing Y2K remediation activities. If your company doesn't do much Y2K fixing, and then all of your systems crash in January, 2000, the insurance company won't want to pay you off on your Business Interruption Insurance policy. They'll feel that you could have prevented this problem and therefore it should be your risk, not theirs.

Insurers are writing limitation after limitation into their policies, trying to control their Y2K-related risks.

But what if you make a diligent effort to fix your Y2K problems, allow appropriate time, do the best you can, but your system still decides to take a vacation on January 1, 2000? Now, you're uninsured. It would be like having your flood insurance cancelled for fall/winter 1997-1998 because the insurance company read weather predictions about El Nino.

I mentioned (Kaner, 1998) that some companies were pushing consultants to sign indemnification and insurance clauses in their contracts. Since then, some attorneys have coined a new buzz-phrase, *OPI*. Other People's Insurance. The idea is that if you can't get your own insurance for your own company, you should demand insurance from your consultants. This could allow you to pass through to them any litigation expenses associated with anything that they were involved with in any way, whether your problems are their fault or not. A neat trick, born of desperation.

Customers

Examples of customers and other recipients of your products include customers of goods, software, and services, receivers of bills and payments, owners of warranties (which expire prematurely) recipients of scheduled services (such as checkups), third party users (who buy your product from a reseller) and strategic partners.

A company might end up in litigation with customers for many reasons. I'll only mention one class of situation.

Some companies are going to triple bill you for products, demanding payment even after they've been paid. Some restaurants will run your charge card through their system twice. Some will take your orders and forget to ship your merchandise. Maybe you'll sue them.

Some of these companies will make mistakes like these because their software is broken.

Other companies will do this intentionally, in an effort to defraud you. There are plenty of scam artists in the world and Y2K will give them a new opportunity. When they're caught, they'll plead "Oh, but my Y2K bug made me do it. I didn't mean to cheat anyone."

Governments and Regulators

Governments and regulators include (for example) local, state, and federal tax authorities, state and federal securities regulators, overseers of corporate management (such as organizations that regulate or monitor nonprofits), financial auditors, labor and plant safety regulators, advertising regulators, product safety regulators, courts and administrative agencies, and certification compliance auditors (such as ISO 9000 auditors).

For example, banking regulators are urging banks to develop better data processing controls. A bank that has weak processes and loses a fortune might not be able to maintain its status with the Federal Deposit Insurance Corporation. (FDIC, 1998)

Internal Stakeholders

Internal stakeholders include (for example) employees, shareholders, lenders, employment-related insurers, unions, officers and directors.

Suppose that you are the chair of a corporation that loses a great deal of money in 2000 because your company did not take adequate measures to make itself Y2K ready.

- Can you personally be sued because you are a director of this company?
- Can your company be sued by its shareholders?

The Business Judgment Rule

Directors are required to perform their duties with reasonable care, but they are allowed to make honest mistakes. The business judgment rule protects decisions that are intended to serve the corporation's business purposes and that do not involve fraud, illegality or a conflict of interest. A decision can be attacked if it is not based on a reasonable investigation--if the company doesn't take reasonable measures to estimate the extent of its Y2K problems, directors might be at risk if they make bad Y2K-related decisions. Also, some directors are on bonus-for-profits plans. They stand to lose bonus money if they spend money (that would otherwise be classed as profit) on Y2K repair. If these people are involved in a decision to economize on Y2K remediation, their decision can be attacked as based on a conflict of interest.

Securities fraud

Publicly traded companies are now required to disclose their Y2K-related risks in the reports they file with the Securities Exchange Commission. These disclosures are available to all shareholders. If these statements are falsely reassuring, and people buy the company's stock on the basis of these statements, then they can sue the company. If the executives knew or should have known that the statements were false when they made them, or they had no way to know whether the statements were true or false when they made them, then the shareholders might prevail in the lawsuit.

ROLES FOR TESTERS

We can help our companies survive this mess.

Companies may or may not turn to their software testing staff for help. Publishers' testers are, in the main, still testing new products rather than Y2K'ing. Software testers have special skills that will help companies limit their risks of Y2K litigation. First, here are some of our skills:

- ***Empirical investigation****:* We know how to look for data to answer technical questions.
- ***Application (end use) orientation*****:** We know how to ask the question, "Will this work for the end customer?" and how to prove that it won't (if it won't).
- ***Able to cope with insufficient information:*** We don't need detailed specifications and well documented code. They are desirable, of course, but we've been working without them for years already.
- ***Used to dealing with tight schedules:*** The ship date is January 1, 2000 or some business-specific date before that. We have to do the best we can within the schedule constraint. Sound familiar?
- ***Used to bad data and strange code:*** Welcome to testing.
- ***Good reporting skills:*** Our primary work product is a report (bug report) that describes a technical problem. We constantly write reports that deal with troublesome technical issues.

Now, some of the ways that we can help the company.

Every corporation needs an assessment of its Y2K-related risks. This includes:

- risks associated with defects in your company's IT (in-house, information technology) software that runs the business;
- risks associated with defective software and data of other companies that will interact with your company's code and data;
- risks associated with defective software in code that you sell, whether COTS, customized, or embedded in some other product.

The Board and the Executives need this assessment in order to make a reasonable business decision about their Y2K investment strategy. The company needs good data to shield itself under the Business Judgment Rule. Werder (1998) notes that the government's dominant strategy in prosecuting bank and thrift officers and directors was to attack the extent to which they made their decisions on an informed basis.

Additionally, publicly held companies need good data so that they can publish it to their shareholders. The more accurate the company's disclosures, the less likely it is to be sued. If the disclosures (predictions of Y2K ramifications) are inaccurate, then the better the process used to arrive at the predictions' estimates, the easier it will be for a company to defend itself against a claim of securities fraud.

You company should also test its billing software. My bet is that a whole lot of con artists are going to do a whole lot of creative billing and then defend themselves with a Y2K-related excuse. A few months into 2000, the regulatory agencies are going to be sick, sick, sick of hearing "But Y2K made me do it." Unfortunately, it's entirely possible that your company will also do some creative (that is, erroneous) billing, by accident. If you make those mistakes, your best defense against someone investigating you for deceptive trade practices is a solid set of test documentation that shows that you made a good faith effort to find errors in the code and prevent errors in the bills. Your company's goal will probably be to convince the investigator *as early as possible* that there is no case here. A good set of test documentation can help your company achieve that goal.

If your company makes embedded software, you should be testing, testing, testing for Y2K-related safety. A primary rule of products liability law is that if you can't make a dangerous product less dangerous, you should warn people against hazards that they might not expect. For example, a big sign at the elevator that

says, "Do not use this elevator after 11:00 p.m. on December 31" might save the company from costly, wasteful litigation.(If no one gets trapped in the elevator, no one sues.)

TEST DOCUMENTATION

Some of you know me well enough to know how much I despise spending time documenting test efforts. My goal is to do the minimum necessary, to leave more time for testing and advocating. *Unfortunately, for Y2K, the minimum might be far from zero.* In the face of a credible litigation risk, documentation can provide valuable defensive evidence. It makes a lot of sense to talk with your corporate counsel about the style and depth of test documentation and test results documentation that she would like to have available, in the event of a lawsuit. More detailed test documentation might not help you find more bugs, more quickly, or in a more organized way. But it might save your company from a devastating lawsuit.

CONCLUSION

I think that there will be a lot of Y2K-related litigation because so much business-critical code (and some safety-critical code) is being revised to deal with Y2K issues. The errors can affect almost any relationship that the company has with any other person or company. Despite that, I doubt that we'll see anywhere near as much litigation as we see threatened in the most common predictions.

But no matter what happens to everyone else, *we* don't have to treat Y2K like deer looking at headlights. We can help our companies take reasonable measures that will go a long way toward protecting them from the most expensive types of lawsuits (the ones involving fraud or breach of a fiduciary duty or injury to other people). We can help by doing what we do best, by investigating critically and reporting results carefully.

REFERENCES

"Year 2000 Vendors Looking for Work, ITAA Survey Says." *Information Week Daily*, September 5, 1997. (As cited in Unger, 1998.)

Caldwell, B. (1998, April 22) "Study: Poor Year 2000 Showing For Largest Companies", *Information Week, www.techweb.com/wire/story/TWB19980422S0004.*

Coffou, A. (March 20, 1997), "Testimony of Ann Coffou, Managing Director of Giga Information Group." U.S. House of Representatives Science Committee. *www.itpolicy.gsa.gov/mks/year2000/hearing.htm*.

FDIC (Federal Deposit Insurance Corporation) (March 18, 1998) "Year 2000 Risk." *Financial Institution Letters*. *www.fdic.gov/banknews/fils/1998/fil9829.html*.

Frazza, P.J., R.J. Jinnett, & M.D. Scott (1998) *The Year 2000 Crisis: Legal Issues Conference*. Glasser LegalWorks Seminars.

Gillmor, D. (May 26, 1998) "Software Industry Wields Fine-Print Attack." San Jose Mercury News, *www.mercurycenter.com/columnists/gillmor/docs/dg052698.htm*.

Gleick, J. (May 10, 1998) "It's Your Problem (Not Theirs)", New York Times.

Information Technology Association of America, "IT Product Able to Meet the Year 2000 Challenge." *www.itaa.org/proquest.htm*.

Institute for Electrical and Electronics Engineers (1997) *Draft Standard for Year 2000 Terminology* P2000.1/D3.4, section 3.2.4, "Year 2000 Compliant Technology."

Kaner, C. (1998) "Contracts for Y2K Services: Traps for the Software Testing Consultant." *Software QA*, in press.

Kaner, C., J. Falk, & H.Q. Nguyen (1993) *Testing Computer Software*, 2nd Ed. ITCP.

Kaner, C., D. Pels (April, 1998) "Copyright Laws and Y2K Maintenance." *Innovations in Software Support: Journal of the Software Support Professionals Association,* p. 2.

Kerr, C.L. (Ed., 1998), *Understanding, Preventing and Litigating Year 2000 Issues: What Every Lawyer Needs to Know Now*, Practicing Law Institute.

MAI Systems Corp. v. Peak Computer, Inc. (1993) *Federal Reporter*, Second Series, Vol. 991, p. 511, United States Court of Appeals for the Ninth Circuit.

Moskowitz, H. & Wallace, R. (March 7, 1996) "Loser Pays. A Deterrent to Frivolous Claims." *The New York Law Journal*, p. 2.

NCCUSL (National Conference of Commissioners on Uniform State Laws) (1998) *Uniform Commercial Code Article 2B—Law of Licensing*. The latest draft is always at *www.law.upenn.edu/bll/ulc/ulc.htm.*

New York State Government (1997) *Year 2000 Warranty Standard*. *www.irm.state.ny.us/yr2000/contract.htm*.

Nuara, L.T., H.P. Benard, & D.K. Rydberg (1998), "Year 2000: Problem or Opportunity?" in Kerr (1998).

Patrizio, A. (1997, October 10) " Most Fortune 500s Not Working On Year 2000 Fix." *Techwire*, *www.techweb.com/wire/news/1997/10/1010y2k.html*.

Produce Palace International v. Tec-America Corp. (1997), Complaint filed with the Circuit Court for the County of Macomb, State of Michigan.

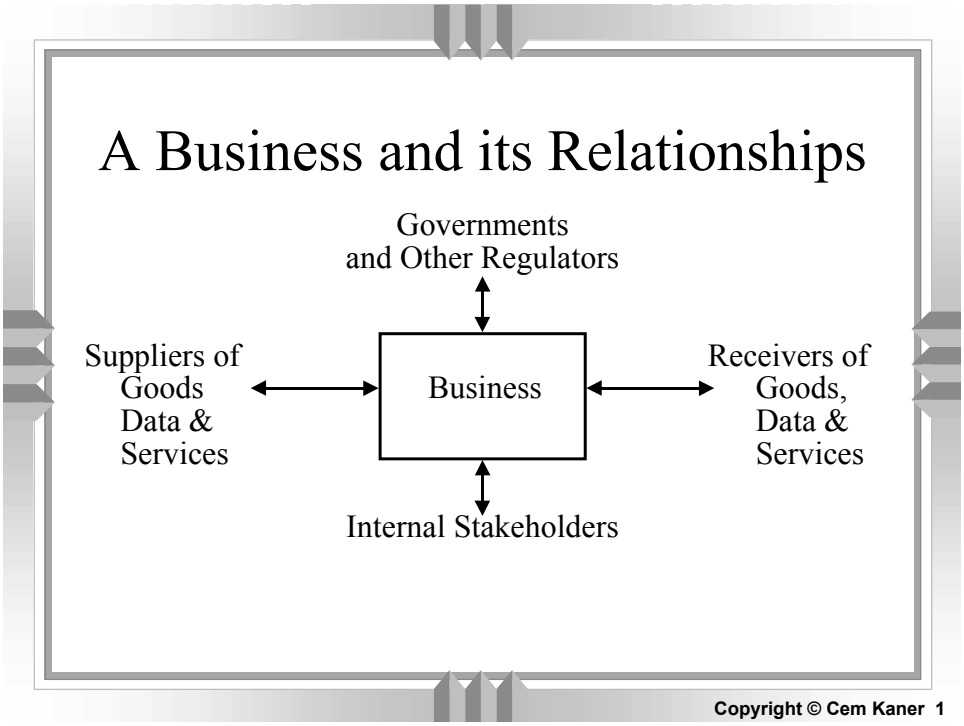
Scott, M.D. & Reid, W.S. (1998) *Year 2000 Computer Crisis: Law, Business, Technology*, Glasser Legalworks.

Unger, T.A. (1998) "Legal Issues for Year 2000 Software Compliance." In Kerr (1998).

United States Government (August 22, 1997) "Federal Acquisition Regulation Final Rule on Year 2000 Compliance." *Federal Register*, vol. 62, No. 163. *www.comlinks.com/gov/farf897.htm*.

Werder, R.I. (1998) "Preparing to Defend 'Millenium Bug' Litigation." *BNA Year 2000 Law Report*, vol. 1, No. 1, p. 39.

Figure 1: A Business and its Relationships



[1] Portions of this paper were presented as a Keynote Address at the Quality Week conference, San Francisco, May, 1998.