Cem Kaner, Ph.D., J.D.
Law Office of Cem Kaner **408-244-7000 (v)**
http://www.badsoftware.com **408-244-2181 (f)**
P.O. Box 1200, Santa Clara, CA 95052 **kaner@kaner.com**

CONTRACTS FOR Y2K SERVICES: TRAPS FOR THE SOFTWARE TESTING CONSULTANT

In press, Software QA Magazine.

In the feeding frenzy that calls itself Y2K, there is a widespread expectation that we'll see many, many lawsuits.

I often hear a $1 trillion legal cost estimated for Y2K. (e.g. Coffou, 1997) This seems awfully high to me. The cost of tort suits in the U.S. runs about $29-36 billion per year. The “tort suits” are the personal injury and malpractice and fraud suits that are so often talked about as “excessive” by groups like American Tort Reform Association. And the total of all direct and indirect costs of civil litigation, as estimated by the American Tort Reform Association (a source that some of us think is biased and perhaps prone to exaggerating the number on the high side), is only $200 to $300 billion per year (Moskowitz & Wallace, 1996). So I don't know where anyone will find the lawyers and the courtrooms to handle a trillion bucks worth of lawsuits.

But even if the number doesn't get close to a trillion, it won't be small.

These lawsuits will make a major mark on the law of software quality, so I've started trying to figure out what kinds of lawsuits we'll see, what the basis for them will be, and how we might prevent some of them as software QA (quality assistance) staff. I explore the general issues in Kaner (1998).

One kind of suit you might be particularly interested in preventing is the one in which a former client or employer (contract) of yours sues you. To a large degree, I think that if you provide Y2K-related testing services, your risk will be determined by the terms of the contract you sign.

Several contracts for software testing services contain some risk. For example, these contracts often include boilerplate clauses in which the contractor (or consultant) indemnifies (essentially, insures) the client against any loss arising from or related to the work done by the contractor. I advise against accepting many versions of these clauses (See Kaner, 1996b) but many people sign these contracts anyway. After all, how often do contractors get sued, even ones who do awful work?

Y2K service contracts are different.

Y2K service contracts carry more risks.

You should therefore be more careful when you sign a Y2K service provider contract (such as a contract to provide Y2K-related testing services).

I think that the contracts will be riskier for the following reasons:

- The companies who are just starting their Y2K efforts now are not likely to have successful projects. Several of them have head-stuck-in-the-sand management (why are they starting so late?) who are unlikely to accept the blame for project failure. Contractors will be their scapegoats.
- There are tricky intellectual property rights involved in this type of maintenance, and the law is being clarified in ways that will make you more likely to slip into an unintentional but serious violation.
- Companies are frightened that their staff will leave as the market rates for programmers rise. Contract clauses that make it harder for you to walk away from a project will be more likely and more likely to be enforced.
- If we have even a tenth as many lawsuits as are being predicted, defending companies and their insurers will run out of money. Anyone who looks like an available "pocket" (source of money) will be brought into the lawsuit by plaintiffs who are trying to recover their losses and who realize that the main defendant won't be able to pay the judgment even if it loses.

With those concerns in mind, here are some defensive contracting suggestions. In many cases, I refer to clauses that will probably appear frequently in contracts written by your consulting clients. In each such case, I recommend that you carefully consider the possibility of modifying or crossing out the clause. I also suggest clauses that belong in your standard contract and as additions to contracts that are based on your clients' forms. If the client company won't negotiate with you, I recommend that you carefully consider whether or not to provide it with services.

Testing will play a major role in Y2K work. There will be a shortage of skilled testers, just as there will be (is) a shortage of skilled programmers. You can and should use your scarcity to negotiate appropriate terms in your contracts.

Defensive Contracting

The point of defensive contracting is to anticipate ways in which the relationship or the job might fail and to prepare for them. It is just like defensive programming—you anticipate remote risks and guard against them. As we've all learned from testing software that wasn't developed defensively, these "unlikely" events happen. Problems are more likely than normal to arise in Y2K situations, so we should pay more attention than normal to our contract terms.

Here are some of the key issues:

1. *Unclear Requirements*

Suppose that you contract with someone to help make their system Year 2000 compliant. *What does this mean?*

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 (New York's, for example) require the program to cope with bad data.

Error handling is just one of many definitional issues. How much special-date compatibility does the client expect you to test for? How far does the client expect you to go beyond what can easily be found with some of the better tools on the market?

Unless you and the client are clear about your definition, you can do the work in good faith and still end up in a dispute with the client over the quality and completeness of your work.

In general, as with all contracts, the more specific you can be about the scope of work, the intent behind the work, and the acceptance criteria, the better off you and your client will both be.

2. Warranty

The client might ask you to warrant (promise, guarantee) that, at the end of your efforts, the software will be Year 2000 compliant, or that all failures of compliance have been discovered.

Year 2000 errors can be very subtle. For example, changing the date from two digits to four changes the length of each record in a database. What if an obscure calculation of the location of something unrelated to a date works from hard-coded specifications of the length of the record? When the record length changes, this badly designed routine will calculate incorrectly and will look up the wrong piece of data. Can you catch every instance of this kind of error? How can you be sure?

Another warranty requested by some companies is that the code changes made to achieve Year 2000 compliance have not in themselves created new errors that are unrelated to the date. That is, they want a warranty of no side effects. It's hard enough to ensure this in situations in which you have all the source code and access to the people who wrote that code. It is nearly impossible to ensure this if the original programmers are gone, or if the program has been patched (rendering the source code listing outdated), or if the original programmers used a tricky design (common back in the days when memory was so expensive that good programmers developed special tricks to save memory and machine time).

It *might* be sensible to ask the programming team to issue such a warranty, but not the testing team. You aren't making the changes that assure compliance without side effects. You are merely looking for evidence that the changes were insufficient or unsafe. It is impossible to completely test the program (Kaner, 1997a, gives examples and explanation that might help persuade a client who doesn't understand this point.)

3. Certification

The client wants you to certify that the software is Y2K compliant. It will advertise your certification to other people (customers or shareholders).

In *Hanberry v. Hearst Corp*. (1969), the publisher of *Good Housekeeping* magazine (Hearst) was held liable for an injury caused by a defective product because it had given the product its "Good Housekeeping's Consumer's Guaranty Seal." *Hempstead v. General Fire Extinguisher Corporation* (1967) involved allegedly negligent certification of the safety of a fire extinguisher by Underwriters Laboratory. In *FNS Mortgage Service Corp. v. Pacific General Group, Inc. and International Assoc. of Plumbing and Mechanical Officials* (IAPMO) (1994), the court allowed a suit to proceed against IAPMO for negligent certification. In each case, the plaintiff (the person bringing the lawsuit) was a third party—a user of the product rather than the manufacturer.

If your certificate of Y2K compliance is relied on by users, and if the product destroys their business because of its Y2K noncompliance, you are in trouble. Think carefully about whether you want to be a certifier.

4. No Third Party Reliance

The *Hempstead* case illustrates an important point. People bought the fire extinguishers because they knew that the extinguishers had been tested by Underwriters Lab. Customers therefore argued that they should be able to hold UL liable, even though its contract was with General Fire Extinguisher, not with them. In general, it might help you fight off a claim that you owe a duty to a third party if you include the following language in your contract:

> <u>No Third Party Beneficiaries</u>. *Consultant provides the Services solely for the benefit of the Client. As the sole intended beneficiary, only the Client has the right to enforce this Agreement. This Agreement is not enforceable by any third parties, including, without limitation, the Client's customers, creditors, potential investors, shareholders, or other developers or testers who are working for, or in conjunction with the Client or with Consultant.*

For more discussion of third party issues that led me to draft this clause, see the American Law Institute's *Restatement (Second) of Contracts*, sections 302, 304, and 315.

I also like the following language:

> *<u>Class of Persons to be Guided by Information Supplied by Consultant:</u> Information that Consultant provides to Client is for the benefit and guidance of Client only and is not intended for communication to a broader audience. Client will not advertise or publicize Consultant's role in the testing of this product in any way that is calculated to increase the confidence of potential customers, investors, or other third parties in the reliability, usefulness, or value of this product.*

For more discussion of the issues that led me to draft this clause, see the American Law Institute's *Restatement (Second) of Torts*, Section 552.

If you're exceptionally successful as a consultant, you might include another clause, requiring the client to indemnify you (pay all your expenses and losses) in the event that you are sued by a third party, such as an end customer of the client. This is a logical clause to include, but indemnification clauses will catch the attention of the client's lawyer. If you ask for too much in your contract (or in your revisions to the client's contract), you risk losing the deal altogether or getting a deal that carries a lot of baggage from an adversarial negotiation.

5. No Life-Critical Applications

Your methods might or might not be appropriate for life-critical or mission-critical applications. Your schedule might or might not be appropriate for critical applications. If you intend to do critical applications, ignore this section. If you don't intend to do critical applications, make that understanding explicit in your contract. If it turns out that someone uses the tested software for life-critical tasks, and that software fails, this clause should help you argue that you're not the right person to confront over the failure of the software to meet the very high standards appropriate for critical software:

> *<u>Consultant's Testing is Insufficient for Life-Critical or Mission-Critical Software:</u> Consultant helps its clients search for problems that might make a product unsuitable for normal commercial use. Consultant strives to be particularly helpful for clients who are operating under challenging deadlines. Consultant does not provide, and Client is not requesting, exhaustive analysis and testing of the product.*
>
> *Consultant explicitly cautions Client that Consultant's methods are not sufficient for testing a product for potential failures that could threaten human safety or the economic survival of a business. Consultant's systems-level approach might be very beneficial as part of a larger testing effort for such a product, but Client is advised and understands that CONSULTANT DOES NOT CLAIM TO PROVIDE, AND DOES NOT CLAIM TO BE COMPETENT TO PROVIDE, THE ADDITIONAL WORK THAT WOULD BE NECESSARY TO ASSURE THAT THIS PRODUCT IS SAFE OR THAT IT IS RELIABLE TO A KNOWN AND QUANTIFIABLE DEGREE OF RELIABILITY.*

6. Highest Professional Standards

This clause says that you promise that your work will conform to the highest professional standards. This clause is an invitation to grief—how do you determine what *are* the highest professional standards in our field? (See Kaner, 1996a; 1997b).

Additionally, on a tight Y2K schedule, how do you actually conform to the highest standards (however you define them)? If you're going to work with the client to get a lot of work done in a short time, you might take some shortcuts. You don't want to be hung for them later.

Don't accept a "highest professional standards" clause. Instead, promise something reasonable.

7. You Have to Promise Something Reasonable

Your client is entitled to good work from you. The client will want some reassurance that you plan to, and promise to, provide that good work. See Kaner (1996b) for discussion of the following two suggestions:

Ocampo, Curtis, & Moss (1996) suggest the following language and provide some methods for resolving disputes about what is meant by "generally accepted industry standards." (But see Kaner, 1996a.)

> *Warranty: Contractor warrants that the services will be performed consistent with generally accepted industry standards.*

This might work for you, but not if the client wants you to cut all possible corners in a race against a deadline. If your client is paying you by the hour, maybe the best clause looks like this:

> *Quality of Performance: Consultant will provide effective, efficient testing services and will strive diligently to thoroughly test the program. Client and Consultant understand and agree that Client faces tradeoffs between its needs for high product quality, timely project completion, and limited development cost. Consultant and Client will work together to determine the extent and depth of testing that can reasonably be achieved in light of Client's other constraints and the Software's design and reliability.*

8. Fixed Price Bids

Beware of fixed-price contracts. The scope of your client's Y2K problem is probably not as well understood as it appears on the surface. Additionally, your client can argue, if you don't "complete" the job, that you're not entitled to *any* compensation because your contract called for a "complete" performance.

This trend is particularly emphasized in recent drafts of Uniform Commercial Code Article 2B (NCCUSL, 1998). This is a 225-page draft law that will govern all contracts for software sales, licenses, and services. Article 2B will probably be introduced in state legislatures starting in early November, 1998. Judges will interpret software service contracts through the Article 2B lens.

This paper doesn't have room for a long discussion of this point, so all that I'll say is that I'm not the only advocate for small service providers (individuals and small companies) who is concerned. Recent meetings of the UCC Article 2B Drafting Committee have been attended by the national chair of the Independent Computer Consultants of America (Sharon Marsh Roberts) and by two Contract Advisors working with the National Writers Union (Harry Youtt and Michael McCready). They've also seen traps for the small firm that offers a fixed price software or documentation service under Article 2B.

9. Damage Limitation

How much should you be responsible for (that is, how much money should you have to pay), if something that you worked on fails badly in the field? One good answer is "Zero." After all, you're just testing the product. You didn't make the bugs and you probably didn't get to set the schedule or to test the product as fully as you would have liked.

But suppose that your client can prove that the defect that is still in the product is there because of your bad testing. The client loses a lot of money (either directly, or via a lawsuit). How much should you have to repay?

Under traditional contract law, you would have to pay for the losses that are caused by your breach of contract, that you should have foreseen would be a natural result of your breach. These are called "consequential damages" (consequences of your breach). Unfortunately, in software this risk (huge damages) is often disproportionate to the size of the contract that you take on. It's widely believed that no one can afford to offer software services if they have to take on full consequential damage liability. (I don't fully agree, but that's a topic for a different paper.) The result of this cost/risk analysis is that it is normal to include clauses that exclude consequential damages in software-related contracts for services or for finished products.

When I say "normal", I mean that anyone with any negotiating power whatsoever kills consequential damages. In the most recent meeting of the Article 2B Drafting Committee (San Diego, March 27-29, 1998) "well over 90%" was a serious estimate of the percentage of software contracts that exclude consequential damages.

If your contract doesn't exclude them, then the traditional contract rules kick in, and you are on the hook for them.

With slight wording variations, the following clause is standard in the industry.

> *Consultant's liability to Client for damages arising out of this agreement shall be limited to direct damages and shall not exceed the amount of fees paid by Client under this Agreement. Consultant shall have no liability for special or consequential damages (including lost profits) of Client or any third party, even if Consultant has been advised of the possibility of such damages.*

The clause is often modified to specify that you are fully responsible for infringement indemnification, but not for any other types of consequential damages.

A refund of all fees is a harsh result for an individual consultant. Losing a year's consulting fees because you missed a bug is pretty serious. If the company thinks you're doing a lousy job, it should fire you, not wait until you've done a huge amount of work and then demand a full refund. You might consider separating out the consequential damages clause, like so:

> *Contractor's total liability to Company for damages arising out of this agreement shall not exceed one-half of the amount of fees paid by the client under this Agreement.*
>
> *Contractor's liability to Company for damages arising out of this agreement shall be limited to direct damages. Contractor shall have no liability for special or consequential damages (including lost profits) of client or any third party, even if Contractor has been advised of the possibility of such damages.*

Now you can haggle separately over the details of the two clauses. You might, for example, agree to pay consequential damages so long as they (and all other damages) are limited to a maximum total of 50% of your fees.

For additional discussion, see Kaner (1996b).

10. Insurance

Several clients' standard contracts ask you to certify that you carry an insurance policy. Often, they specify that the policy should be for at least $1 million.

The normal expectation is that you are carrying General Commercial Liability (GCL) insurance, which provides coverage in the event of accidents of various kinds. General Commercial Liability insurance is not Errors & Omissions insurance. In most states, under most standard GCL policies, the odds are good that you will not be covered for claims filed against you that allege that you did incompetent or inadequate or irresponsible testing. (That's what Errors & Omissions insurance is for.) Nor will you be covered (probably) for claims filed that allege that you were part of a team that developed a seriously defective software product, especially if the defect is a Y2K defect (either direct error involving Y2K or a side effect of a Y2K remediation effort). Many insurance policies now specifically exclude Y2K-related liability.

In years past, I saw a disturbing way of handling these clauses. Consultants would sign contracts that say that the consultant carries the insurance, but the policy involved (when it existed at all) would be the consultant's homeowner's policy or personal umbrella policy. These don't cover business related risks.

Understand what you are doing when you sign a contract that says you carry insurance, but you don't carry insurance of the kind anticipated in the contract:

- Depending on the wording of the contract, you might be committing fraud.
- You are in breach of the contract from Day 1. This gives your client negotiating opportunities to deal with you in unpleasant ways.
- You are announcing to the world that you are a deep pocket. You have an extra $1 million of insurance coverage to offer, if someone wants to sue your client but realizes that the client can't pay off all of the damages it will be liable for. Remember: if the Y2K lawsuits really rack up into the hundreds of billions of dollars, some insurance companies will run out of money. Your million dollar policy offers a chance to the plaintiff to split her recovery across two insurance companies, something that might be very attractive. Therefore when your client is sued, you are more likely to be sued too. If your contract says that you have that insurance, but you don't have it, kiss your house and your bank account goodbye.

If you are going to sign insurance clauses, get insurance. And have the clause specify exactly the type of insurance that the client expects you to be carrying.

If you don't already have this insurance, explain to the client that your rates will have to go up to pay for the insurance premiums. When the client shows you a contract with these terms, which were never discussed when you first talked about doing this job for the client, realize that the client is negotiating the deal. You can negotiate too, by explaining that these unexpected provisions change your cost structure significantly, so your pricing has to change too. Some clients are very receptive to this reasoning.

11. Implied Warranties

When people sell merchandise, the Uniform Commercial Code supplies implied warranties of merchantability and (sometimes) fitness for use. These were written for transactions in goods, not services. But software goods and services law are getting closer. (In UCC draft Article 2B, they converge in the same statute.)

I don't know what an implied warranty of merchantability or fitness would *mean* in the context of a testing services contract. Let me suggest that you might not want to pay your lawyer a fortune for the privilege of being the first kid in your state to find out (in court) what these warranties mean. Until the law stabilizes, I suggest that you make your promises explicitly and kill the implied warranties:

> *DISCLAIMER OF IMPLIED WARRANTIES: EXCEPT AS SPECIFICALLY PROVIDED IN THIS AGREEMENT, CONSULTANT PROVIDES ALL DELIVERABLES, PRODUCTS, AND SERVICES "AS IS" AND WITHOUT WARRANTY. CONSULTANT HEREBY DISCLAIMS WITH RESPECT TO ALL SERVICES AND OBLIGATIONS PROVIDED IN THIS AGREEMENT ALL IMPLIED WARRANTIES TO THE MAXIMUM EXTENT ALLOWABLE BY LAW.*

The capital letters are there to satisfy UCC requirements of conspicuousness. They're ugly, but you should use them for this clause.

12. Infringement Warranty

The infringement warranty is a guarantee from you to the client that you aren't using copyrighted materials, trade secret materials, or patented materials, in ways that could get the client sued when it uses your work product. Ocampo et al. (1996) discuss this in great detail. The bottom line is that you shouldn't promise more than you can control.

Of particular interest: if you develop some particularly jazzy technique or technology for detecting Y2K-related errors, you might be a parallel inventor. The other inventor might get a patent. At that point, you are an infringer (oops). It's impossible to know what patents are cooking in the privacy of the Patent Office (and its counterparts around the world) and it is expensive, expensive, expensive to search the patent databases to figure out whether you're doing something that has been publicly discussed.

Few (if any) testing consultants know the intricacies of the patent systems well enough to reasonably safely promise that they aren't parallel inventors. Instead, you should promise not to infringe, to the extent that you know or should know that what you are doing might be an infringement. For language, see the next section (Indemnity).

13. Indemnity

Kaner (1996b) focuses on indemnification issues. I won't repeat the discussion here, but to keep this paper somewhat self-contained, I'll reprint my preferred indemnification clause. I think this is fair to all sides, and manageable for the consultant.

> *Consultant agrees to cooperate in the defense of Client in any claim made by a third party that the Software developed pursuant to this Agreement infringes on or violates any patents, copyrights, or trade secrets of such third party. Consultant agrees to indemnify Client against liability to third parties from any settlement or final judgment award, including without limitation reasonable attorney's fees and other expenses awarded, that arise from the use of subject matter by Consultant*

or by Client, to the extent that such subject matter is provided by Consultant, in which Consultant knows or reasonably should know, that others have rights.This indemnification is contingent on Client providing prompt written notice of such a claim to Consultant, and granting Consultant the right to participate in the defense of any such claim, and the right and opportunity to approve or reject any settlement of any claim for which Client will seek indemnification from Consultant.

14. Infringement of Third Party Rights

Imagine a client calling you to help it patch its software. The software is not Y2K compliant. The software vendor is a jerk (in your client's opinion) and the client never wants to deal with this vendor again. Additionally, the vendor's technical staff is booked solid for the next year, so its bug fixes won't come in time for an orderly testing cycle before the patched code is put in service.

The client decides to fix the software itself, with some help from consultants, including you.

In this case, you are the "first party" to the contract. The client is the "second party." The software vendor is not directly a party to your contract—it didn't sign anything or agree to the deal between you and the client. But it is a "third party" to your contract because it is affected by your contract.

This maintenance effort sounds reasonable. The client has the software. You'll do the work on the client's machine, where the software is licensed (or sold) to run. You're not making extra copies of the software or revealing anything about it to competitors. The vendor should have no objection to your doing this work.

Unfortunately, the world doesn't work quite this way. Some vendors want to lock their customers into getting maintenance services from them and some recent court decisions say that they can pull this off.

Here's an example. In the case of *MAI Systems Corp. v. Peak Computer, Inc*. (1993), MAI sold computers. Peak was a third party service organization. Customers would call in Peak to maintain their computers. If you had an MAI computer, the Peak staff member would come onto your site, turn on your machine, boot its operating system, run the diagnostics that came with the machine, and then do what was necessary. In this situation, MAI and Peak are competitors for your service business. MAI was able to stop Peak from servicing MAI computers because MAI's license restricted use of the software to not more than three of the customer's "bone fide employees." Even though Peak was working at the customer's site, at the request of the customer, running software licensed to the customer, and MAI supplied this software to the customer specifically to be run on this computer, Peak was a contractor, not an employee of the customer. Therefore, Peak's use of the software was a violation of the terms of MAI's license, and was therefore a copyright infringement.

The reason that MAI could do this is that it provided the software under a *license*. Under a license, a vendor can set rules on how people use its product, including restrictions that would be unreasonable, as the MAI restriction would be, if we had a sale of a copy of the software rather than a license of it. The sale of the copy is governed by federal Copyright law, not state licensing law, and great effort has gone into the Copyright Act to balance the rights of publishers and users of information.

Here are some of the other restrictions in MAI's license:

> <u>Customer Prohibited Acts</u> *. . . Any possession or use of the Software . . . not expressly authorized under this License . . . is prohibited, including without limitation, examination, disclosure, copying, modification, reconfiguration, augmentation, adaptation, emulation, visual display or reduction to visually perceptible form or tampering . . .*

Reverse engineering, even for the purpose of making this product interoperable with some other product or for the purpose of fixing the program, is prohibited. So, how do you figure out how the program works, in order to fix it or to test it? Maybe you can't (legally).

As the independent contractor, you sit in the shoes of Peak. You run the risk of a lawsuit for copyright infringement, contributory infringement (the client infringes by letting you use the software; your willingness to do this work encouraged the client to violate the license), and maybe for trade secrets violations or unfair competition.

This was, and continues to be, a controversial decision. Nuara, Benard & Rydberg (1998) cite some interesting cases and authorities that conflict with this decision. I think that it is an outrageous misuse of copyright, that invites surprise and abuse.

However, the UCC draft Article 2B draft adopts the most controversial bit of MAI, which held that when you run a program, you make a temporary copy of it in RAM and this copying can be restricted by the software vendor. With that in place, the rest of the restrictions are valid in a license. Assuming that 2B is approved by your state's legislature, you will be running an interesting risk when you maintain code for a client who has lawfully obtained that code from the third party.

You would be wise to get an assurance in writing from your client that it has the right to grant you access to the software that you will be testing. I don't (yet) have language for this that I'm willing to publish. If I was testing third-party software and if I had any doubts about the publisher's rights to let me work on it, I would also request an indemnification—the client would pay my expenses and litigation losses in the event that a third party sued me for using the software in the way that I had been directed by the client.

15. Warranty of Completion

This is a promise on your part that you will stay on the project to the end. In conjunction with a shifting definition of compliance, a warranty of completion can leave you stuck on a low-paying, unpleasant project that you thought was finished weeks or months ago.

The reason that you see clauses like this now is that people are afraid that you'll quit when you are (inevitably) offered a significantly better deal for a new contract. I've seen serious estimates/predictions that the pay of programmers will double every six months over the next two years. Testing makes up at least half of the Y2K work by most estimates. Therefore if programmer pay soars, it should also soar for contract testers. This will provide a powerful incentive for people to go job hopping.

The result is that some companies are making significant efforts being made to keep people through the year 2000. On the nice side, some companies are setting up a bonus pool for people who stick it out until January, 2000. One company has set aside $30 million to be split equally among the members of its programming staff who are still at the company in January, 2000. There are currently about 600 programmers on staff so if they all stay, they all get $50,000. If half of them quit, the rest get $100,000 for staying.

Alternatively, a substantial portion of staff members' (and contractors') pay might be withheld until the project is completed. You are legally free to leave before the project finishes, but you lose that withheld pay.

The completion warranty is another variation on the theme (as are the noncompete and intellectual property clauses below). This approach obligates you to stay with the company or it makes it much harder for you to leave and find work elsewhere.

I suggest that you live up to your commitments—what goes around comes around. But beware of these clauses because they can be stretched beyond what you believed your commitment would be. It makes sense to avoid (cross out) this type of clause unless it is clearly to your advantage.

16. Noncompete Clause

This is another way to keep you on the project. The client gets you to agree not to work for its competitors. Now you have fewer places to go to find your next contract.

Some of these noncompetes are pretty broad—you are pretty much stuck working for the company that talked you into signing the contract with the noncompete clause in it. Oh, you can quit. But you'll be driving a cab instead of testing for the next five years.

Historically, courts have frowned on these clauses and either thrown them out or interpreted them very narrowly. That trend is gradually reversing—I now take noncompete clauses seriously when I negotiate a contract for my own services or for a (legal) client.

My own normal practice is to cross the clause out. I explain that the (technical) client (who is retaining me as a consultant, not a lawyer) is always free to retain other consultants. I'm a business too. I have to be free to find other clients.

Another approach is to rewrite the clause very narrowly, to cover only your client's most direct competitors. I don't like that approach—it is sometimes still too restrictive, especially if you have specific skills that appeal to a limited group of companies. But it is better than the broad clauses that you're likely to see in the contract in the first place.

17. Ownership of Your Knowledge and Tools

The final group of handcuffs that a client tries to apply to you are the intellectual property and nondisclosure clauses. The client's standard contract declares that it owns every thought that you have during the period that you provide services to the client. If you come up with a new algorithm while taking a shower at home, it belongs to the client. Additionally, all of the code that you write and all of the test planning forms that you create become the property of the client, and you grant it copyright over those work products and others like them. If you use any pre-existing tools or standard routines on the client's job, oops, you just gave away your rights to them. The net result of the broadest of these clauses is that you can't do much for your next client without risking (in theory) a lawsuit from your last one. This might make you more cautious about switching clients when a potential new client offers to double your pay.

These clauses are the most heavily negotiated in serious software development and consulting contracts. This is not just a Y2K issue and it is not much more troublesome in Y2K contracts than in all the others. I don't have language for this that I'm willing to publish. Here are some negotiation suggestions:

- Point out that you are an independent consultant or contractor and that you work on multiple projects. You'll gladly grant rights to material that you develop for the client at the client's actual expense. However, anything that you develop on your time is yours and anything you develop for a different client is that client's.
- Offer the client a nonexclusive license to use (in any way that it needs to use, including copying, modifying, and granting others the right to modify or copy) any of the code or tools or other intellectual property that you have provided it. However, you will retain copyright to the material that you brought with you to the job and to anything that you developed on your own time (not charged to the client) during the period that you were consulting to the client.
- Explain that the reason that you are efficient is because you develop tools that you carry from job to job. The client is benefitting from your knowledge, skill, and toolbox. You have to preserve that toolbox for the next client because it is your livelihood.

Some day, I'll write a paper that is focused on these intellectual property issues. In the meantime, my only suggestion is that if you are in doubt about one of these clauses, you should be cautious about agreeing to it. Consult your lawyer for suggested language and negotiating strategies.

References

American Law Institute, *Restatement (Second) of Contracts*, *www.ali.org*.

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*.

FNS Mortgage Service Corp. v. Pacific General Group, Inc. and International Assoc. of Plumbing and Mechanical Officials (1994), *California Appellate Reports*, 4th Series, vol. 24, p. 1564. (District of Delaware).

Hanberry v. Hearst Corp. (1969) *, California Reporter,* vol. 81, p. 519. (California Court of Appeal.)

Hempstead v. General Fire Extinguisher Corporation (1967) *Federal Supplement,* vol. 269, p. 109.

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. (1996a), "Computer Malpractice", *Software QA,* Volume 3, #4, p. 23. *www.kaner.com/malprac.htm*.

Kaner, C. (1996b), "Contracts for Testing Services: Indemnification and Warranties." *Software QA,* Volume 3, #5, p. 20. *www.kaner.com/contract1.htm.*

Kaner, C. (1997a) "The impossibility of complete testing", *Software QA*, volume 4, #4, p. 28.

Kaner, C. (October 8, 1997b) "Legal Issues Related to Software Quality", presented to the *Annual Meeting of the American Society for Quality, Software Division,* Montgomery, AL.

Kaner, C. (May 27, 1998), "Year 2000, How Can I Sue Thee? Let Me Count the Ways!" Keynote address to the *11th International Software Quality Week*, San Francisco, CA. By the time you read this, there will probably also be an associated paper at my website, *www.kaner.com*.

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.

National Conference of Commissioners on Uniform State Laws (NCCUSL) (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).

Ocampo, R.L., S.S. Curtis, & J.J. Moss (1996) *Negotiating and Drafting Software Consulting Agreements*. Glasser Legal Works.

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*.