Both Sides Always Lose: Litigation of Software-Intensive Contracts

Litigation of software-intensive endeavors is a major growth industry. The costs of litigation are rising faster than any other aspect of software development. Understanding five key patterns of such litigation is essential to maintaining the health of any organization that builds or buys software.

A contract is a kind of specification. Instead of describing a new system, it describes a business agreement. A contract suffers from some of the same difficulties that plague a specification: neither is ever entirely clear, entirely *right*, or entirely free from interpretation. None of these problems is fatal when there is goodwill and a commonality of interests between the parties. With these two essential ingredients, people work out their differences and come to a successful conclusion. But when the two essential ingredients are missing, that is when matters begin to go south.

Going South to the Tune of More than $100 mil

Take, for example, the case of a contract to build an ambitious reservation system a few years back. The disputants were the Fly-By Information Services Co. and the Magnificent Hotel Corp. (fictitious names used to protect the guilty parties and the authors). Fly-By was to build the system and Magnificent to specify the requirements and pay for the result.

The first sign that all was not right was the extraordinary difficulty of coming up with a contract. As with a specification effort, when contract negotiations are particularly rough, that is a sign that there is conflict brewing beneath the surface. This negotiation was fierce. Everybody hated everybody. It was like one of those marriages that quickly dissolves into bitter acrimony. You have to wonder, when the principals are so wrong for each other, why they bothered. So, too, the contract between Fly-By and Magnificent.

One way to deal with conflict is to paper it over in ambiguity. You do that when you write a speculation that conceals disagreement rather than pleasing one of the disagreeing parties at the expense of another. In an aggressively negotiated contract, the result is not what you would at first expect, a contract with every "i" dotted and "t" crossed. Instead, you tend to end up with ambiguities wherever there are truly unresolvable conflicts. That is what happened in the Fly-By case.

The rest of the story is especially grim. The case was litigated for years and finally settled, with nobody really winning. The settlement was $100 million-plus, but not enough to make anyone whole. No system was built or delivered. Everybody wasted years of their lives. The legal fees were staggering. The opportunity costs, the useful things they could have been doing instead of litigation, were even worse. Both sides showed huge losses.

Since the contract was weak, the case finally turned on a single incident: Fly-By had fired a succession of managers who tried to tell the big boss that the date was unworkable. Magnificent found out, brought in the fired managers as witnesses, then pointed triumphantly to one clearly written contract provision that said Fly-By was obliged to inform its partner if it had credible reason to believe the delivery would be late. That cost Fly-By more than $100 million. Everybody ended up with enormous losses, but Fly-By's were more than Magnificent's.

Even told in such sketchy terms, the case of Magnificent vs. Fly-By contained many of the patterns of a typical litigation where software is at the heart. The first of these is obvious to anyone who has ever been involved in software litigation:

Pattern 1: Both sides always lose.

Stepping Back for a Look at the Context

As impressive as growth of the software industry has been, it is outpaced by growth of software-related litigation. It is not unusual for a large software development organization today to have upwards of 50 active cases on its hands. Litigation costs are not usually charged against an information technology budget, but if they were, they would be—when spread across unlitigated and litigated projects—a larger component than coding. We are experiencing an epidemic of litigation on software projects. This is different from the general litigiousness that has often been noted about our society. Americans are all too prone to ask the courts to resolve disagreements, but this has not in the past been particularly true of American corporations. Corporations understood that everybody loses in litigation and that the costs can be ruinous.

The growth of software-related litigation is, we think, due to some factors that affect all corporations today, but have been particularly strongly felt in the software sector. The extensive layoffs in the early 1990s, together with the *lean and mean* attitude they engendered, have led to the glut of litigation today.

The Heart of the Matter

There have been downsizings in our economy before, but the ones that struck us so hard between 1990-95 were curiously vindictive. The people who were booted out were made to feel that they were not just unlucky, but somehow at fault. They were *fat* that needed to be trimmed. The people who remained were also made to feel bad and told in no uncertain terms that they would have to pick up all the work of their dismissed colleagues, and then some, or else risk being booted out.

It often fell to information technology (IT) management to carry out the downsizings and to convey upper management's attitude of righteous indignation about *fat* on the payroll. The effect on the workers was either catastrophic for those who were laid off, or depressing. That much was obvious. What was not so obvious was the effect on IT management. Our speculation is that the downsizing exercise made many IT managers more fearful and insecure than ever, and caused them to retreat into an attitude of embattled authoritarianism. This attitude has been bad for everyone (except Scott Adams, who has turned it into millions).

What does this have to do with litigation? We believe it is at the very heart of the flurry of litigation that we began to observe by 1995. Embattled IT managers, fearful and under the gun to show improved performance, fell back on lines like. "Do not tell me it cannot be done in two years. I am the boss. It will

be done in two years."

Engineers, who knew that deadlines were unworkable or that quality would suffer, shrugged in the face of such insistence. They reasoned that management would learn in the long run that impossible is impossible. And so impossible targets were accepted. They made their way into contracts. The contracts were signed. The project workers tried their best and failed. Then the parties went to court.

This is almost exactly the story of the project that Fly-By conducted for Magnificent. Fly-By wanted the work and bid to win. It obligated itself to perform at a level that time would prove was unattainable. The voice of reason was drowned out by that of management, intent on using its force to impose its wishes. It succeeded, briefly.

Pattern 2: When authority trumps reality, reality always wins in the end.

"Golly, How the Truth Will Out."

An invariant feature of such litigation is that all project records are subpoenaed. That means that each and every manager and worker has to provide the entire contents of his or her files to be copied and provided to the other side. There are no exceptions. You may be reluctant to provide a copy of that bothersome little memo where the true defect rates were discussed, but as the court papers make clear, if you do not provide it and you are found out, you go directly to jail. Not the company, and not just officers of the company, but you. For most employees, this is a fairly persuasive argument. And so, even the most incriminating evidence generally gets delivered.

As litigation consultants and expert witnesses, we spend an inordinate amount of time poring over these project records. The surprising thing is that, in most of these cases, there is a great deal of what we might call *lying*. It is not at all unusual to find yourself with a memo from X to his boss stating that delivery will be delayed by at least six months and a memo from X to the client, dated the same day, providing comforting assurances that the project is on schedule.

Outright lying is a direct sin of commission. Equally common are sins of omission, situations where X knows something bad and neglects to tell it to the client. Omission is a different flavor of lie. Both are subject to this invariant of legal cases:

Pattern 3: Lying to your contract partner is morally indefensible, generally illegal, and always gets found out in litigation.

The old rule of honorable behavior—do not lie to the other guy—seems to have been replaced with a new rule: do not tell the other guy anything that is not true unless he has no possible way of knowing that what you say is not true. Same with sins of omission (what the lawyers call *unfair surprise*), the new rule seems to be that it is OK to neglect to mention an inconvenient fact that the other guy has no way of hearing about on his own. All this subterfuge comes back to eat you alive during litigation.

The Other Side of the Coin

We have spoken mostly of cases where the software builder overcommits. But it is also possible for the other party to be at fault. This happens when a software buyer imposes *wishful thinking* deadlines on a builder, conceals real requirements to keep the price down, and hopes to impose them on the builder as freebie changes, tries to get ambiguity into a contract to be exploited later, and so on. In general, buyers are every bit as likely as builders to try to trump reality with authority, and to tell little lies. And they are just as subject to the three patterns presented.

Buyers are particularly prone to one of the worst fallacies of contracting, the idea that risk always moves with responsibility. It does not. When you are the buyer and another organization agrees to build a system for you, signing the contract moves primary responsibility for successful implementation from you to the builder. Not all the risks involved in attempting the project move with that responsibility (no organization can completely buy its way out of risk). If the contractor fails to deliver, both parties lose. Since this is a real risk from the buyer's point of view, it is incumbent on that buyer to manage that risk. Software is a risky business and both sides of any software project need to do serious risk management. We think that most organizations understand this, but our data indicate that those who get into litigation are disproportionately likely to avoid it:

Pattern 4: Litigation is almost always a result of imperfect risk management by either or both parties.

Most of the litigations we have seen have involved organizations that failed to do any risk managment at all.

Everything we have discussed so far deals with events that precede the actual court case, the causative factors. That is interesting in the abstract, but if you are about to go to court, your mind is understandably occupied with other things, i.e. what do you do to avoid losing your shirt?

You are in a Litigation–What Do You Do?

Litigation may be a lose-lose business, but there are certain defensive strategies that may help to limit your losses. The most important one we know is to investigate and calibrate your project with respect to industry norms. For example, if the other side alleges that you have been fickle about the requirement and heaped ruinous numbers of change requests on the builder, your best defense is to show that your performance in this respect is better than the industry norm. It helps to know that the industry norm for changeability of specification is someplace between 1 and 2 percent per month of the original specified size (measured in function points). It helps to have a good source for such evidence, in this case *The Condensed Guide to Software Acquisition Best Practices* [1].

As you may surmise, a successful use of the norms implies a certain fluency with metrics. Organizations that cannot or do not measure themselves in a fairly systematic way are always at a huge disadvantage in litigation. If you are deficient at measurement, and the other side is on top of it, the jig is up. Metrics is one of the three major subjects on which virtually all litigations turn:

Pattern Five: Most litigations end up focused on measurement, management, or requirements practice, or some combination thereof.

The things you do to win a litigation are doing careful measurements work, focusing on good management practice, and conducting exhaustive and thoughtful analysis of requirements. Paradoxically, these also are three of the principal things you should do to avoid litigation.

A Final Word

The more you think about litigation, the more you must think about the underlying contract. It is tempting to conclude that the lesson here is to get your lawyers involved early and often in order to come up with iron-clad contracts that simply cannot go wrong. While early legal work may be a good investment, it is not entirely realistic to think it can avoid all problems. If the basis of understanding is flawed, no contract will improve it.

Rather than focusing on the contract as a legal instrument, turn your attention to the understanding at the heart of it. When parties develop a real commonality of interest, all is possible. Litigation becomes less likely and success more likely. The project may not go smoothly from beginning to end, but it will tend to respond constructively to the problems it encounters and have the best chance of delivering a useful and effective system.

How can you know whether you are forging a partnership as opposed to signing a deal that will come back to haunt you? The best rule is to think of a contract as bad if one party can win while the other loses. Win-lose situations are a precursor to litigation, since only the most saintly organization could accept lose-lose when it could, under the guarantees of the contract, slip into a win-lose situation. This is the kind of contract you must avoid, even if your organization is the putative winner.

But this sounds crazy. Is not the heart of successful contracting an attempt to place yourself in a position where you can win at the expense of the other party? Too often it is, but such contracts usually turn out badly. The only contract that is truly healthy is one where you feel good signing either side.

That starts you out with the best chance of success. Measure carefully, manage well, and pay attention to requirements.

One more thing: do a little risk management, just in case.◆

Reference

1. *The Program Manager's Guide to Software Best Practices*, Crystal City, Va: Software Program Managers Network, Sept. 1995.

This article has been reprinted with the permission of Cutter Information Corp., provider of information resources for IT professionals worldwide. It originally appeared in *Cutter IT Journal*, April 1998, Vol. XI, No. 4 http://www.cutter.com/consortium

About the Authors

Tom DeMarco and Tim Lister, principals of the Atlantic Systems Guild, have been partners for nearly 20 years. They are the authors of *Peopleware: Productive Projects and Teams*, co-editors of *Software: State of the Art*, and joint designers of seminars on risk management for software; leading successful projects, and controlling software projects. Their consulting practice focuses on project management, measurement, and development methods. They are called on often to serve as expert witnesses and litigation support consultants.

Tom DeMarco
115 Shermans Point Road
P.O. Box 160, Camden, Maine 04843
Voice: 207-236-4735
Fax: 207-236-8432
E-mail: tdemarco@systemsguild.com
Internet: http://www.systemsguild.com

Tim Lister
353 W. 12th St.
New York, N.Y. 10014
Voice: 212-620-4282
Fax: 212-727-1044
E-mail: lister@acm.org

Quote Marks

"Computers are useless, they can only give you answers." —*Pablo Picasso*

"Act in haste and repent at leisure; code too soon and debug forever." —*Raymond Kennington*

"I have traveled the length and breadth of this country and talked with the best people, and I can assure you that data processing is a fad that won't last out the year." —editor in charge of business books, Prentice Hall, 1957

"Software and cathedrals are much the same—first we build them, then we pray." Samuel T. Redwine, Jr.

"I don't think it's that significant." —*Tandy president John Roach, 1981, on IBM's entry into the microcomputer field.*

"I do not fear computers. I fear lack of them." —*Issac Asimov*

"Computers make it easier to do a lot of things, but most of the things they make easier to do don't need to be done." – *Andy Rooney, 60 Minutes*

"Computers in the future may weigh no more than 1.5 tons." —*Popular Mechanics,* 1949, forecasting the relentless march of science.

"A computer does not substitute for judgement any more than a pencil substitutes for literacy. But writing without a pencil is no particular advantage." —Robert S. McNamara, *The Essence of Security*