WHERE DID WE GO WRONG?

LITIGATING A FAILED SYSTEMS DEVELOPMENT CONTRACT

BY: MICHAEL J. SILVERMAN[1]

DUANE MORRIS LLP

Introduction

Your company has spent millions on a new super gigaplex, terabyte, flat screen e-commerce, Internet-enabled, enterprise-wide, super-robust computer system. The consultants who are developing the system are working like small farm animals – they seem to live in their cubicles, and they have a tired "deer caught in the headlights" look in their eyes. The system is late, and it does not appear to be working properly. Nevertheless, the developers assure you that they can work out the bugs while rolling out the system. Or perhaps you are the developer desperately trying to install a sophisticated system for a client who will not talk to you. Sound familiar? According to Standish Group International, Inc., 46% of all information technology projects are over budget or overdue; 28% fail totally. The Standish Group also predicted that only 24% of IT projects undertaken by Fortune 500 Companies would be successfully completed (*CIO Magazine*, "Lessons Learned – To Hell and Back," December, 1, 1998). These statistics have not improved over the years. In fact, recent data shows that many of these projects are becoming the subject of litigation. The Cutter Consortium reported that 78% of the IT organizations in its March 2002 survey reported being involved in a dispute that ended in litigation. Clearly, the risk of failure, or at least less than optimal return, on a company's system development investment is fairly high. Moreover, the vast majority of contracts with systems developers place the risk squarely on the shoulders of their clients. Most such contracts contain significant limitations on the types and amounts of damages the clients can get from developers. Despite the risks, systems developers are busier than ever creating new computer systems for their clients.

Most systems developers use sophisticated estimating, project planning and management, development and testing tools and technologies to increase the probability of delivering a system on time and within budget. For example, the larger developers have created vast, detailed systems development methodologies which their staff are supposed to follow. These methodologies provide a guide through the various phases of the development process: information gathering, needs analysis, business design (designing the system to meet business needs), technical design (designing the technical aspects of the system so that it functions to meet the business needs), construction of the system, testing, implementation and system support. These methodologies are supposed to encourage effective communications between the

---

[1] Michael Silverman is a partner in the Chicago office of the international law firm of Duane Morris LLP. He is the Chairman of the Firm's Information Technologies and Telecommunications Practice Group. Mr. Silverman's coordinates are mjsilverman@duanemorris.com; 312-499-6707.

developer and the client and help manage the expectations of both parties. Clearly, the Standish Group's statistics indicate that something is not working.

What happens when the project really starts to falter, and it becomes obvious that the system will not be delivered on time, on budget, and with proper functionality? Worse yet, what happens when the system is delivered, but does not work? What happens when the developer delivers a system which appears to have been tested, appears to be working, but begins failing after two weeks? If the parties cannot agree on a solution, the problem may end up in the laps of lawyers, an arbitrator, judge or jury. In larger development projects, client and developer personnel work closely on the project, so there are likely to be a storm of claims of poor performance by both sides working on the project. The business people on both sides will be involved in hours of time away from the business, working with counsel to prepare the case.

Damages Limitation and Fraud Claims

Most development contracts contain stringent limitations on the kinds of claims that the client can assert against the developer, and significant limitations on the damages which a court can award in a claim against a developer. For example, many development contracts provide that the developer will not be liable for any consequential or punitive damages. Thus, the client could not recover profits it may have lost (i.e. from lost sales, disgruntled customers, etc.) from being unable to roll out a system as a result of system failure. Many contracts also limit damages for which a developer may be liable to the amount of the contract or to amounts paid during a certain period of time.

Nevertheless, there are situations in which these kinds of contractual limitations can be overcome. For example, if the client is induced to sign a contract by a developer's intentional misrepresentations, the client could argue that the damage limitations were induced by fraud and are voidable. Proof that a developer made intentional misrepresentations to a client in order to induce that client to enter into the contract with the developer is difficult. To construct such a claim, the client should analyze the developer's promotional materials; its staffing requirements; the educational requirements for staff on the project; the project planning documents; work done by the developer as part of its investigation of the project prior to signing the contract; the developer's estimates of time, cost and hours required to complete the project; and any other practices or procedures that are called for by the developer's systems development methodology.

For example, developers often spend a good deal of time working with client personnel in order to understand the extent of the project and the client's needs so that the developer can make a proposal regarding the project. The materials relating to this investigation and any project reports from the developer to the client are a good place to start looking for information which might be used to build a case against a developer. The client must then compare the developer's internal documentation developed prior to entering into the contract, and its representations to the client in selling the project and in negotiating the contract. Often, the developer and client will enter into a contract for one phase of work with the intention of completing subsequent contracts to cover later phases of the work. For example, developers often contract to perform a needs analysis for a client with the intention that the parties execute another contract for the actual development of the system. Also, when projects start going bad, developers and clients often enter into additional contracts to cover work which may be needed to "shore up" the project and increase the likelihood that the project will be successfully completed. A practice of entering into successive contracts on a project or entering into a later contract to fix the project

can provide the opening a client needs to develop a claim for fraudulent inducement in entering into the later contract. If the developer is aware prior to entering into the later contract that the project is not going well, and the developer does not advise the client fully of the project status, the client could claim that it was fraudulently induced to enter into the subsequent contract.

In litigation between developers and clients, there will also be claims of breach of contract and possibly breach of fiduciary duty. The following are some of the claims and defenses a developer and a client are likely to assert.

Claims Against the Developer

The System Did Not Meet Specifications. The client will certainly file a breach of contract claim for the system's failure to meet specifications. In pursuing such a claim, a court or arbitrator will review the actual functionality of the system and compare it to the business and technical specifications the system was supposed to meet. This analysis is tedious because the systems specifications are often voluminous and technical. Clients will likely want to use the business design specifications as the standard against which the systems should be measured. The business design specifications describe how the system should work in terms business people might understand. For example, "the system will process five thousand customers orders within a 24-hour period." On the other hand, the developer is likely to argue that the technical specifications should be used to set the standard against which the system should be measured. These technical standards are far more specific and describe the actual technical solution that was supposed to be implemented to meet the business needs of the client. Developers use technical design specifications as the standard because they may meet the technical requirements of the system (i.e., "the developer shall use an AS/400 computer"), even though those technical requirements may be insufficient to meet the client's business needs.

The Developer Failed to Follow its Systems Development Methodology. If possible, the client should try to get the developer to agree in the contract that it will follow its systems development methodology in performing its contractual obligations. This systems development methodology will provide a very detailed standard of performance documented in many volumes of technical jargon which a developer might have to prove it met in performing its work on the project. A developer may find itself in the awkward position of having to justify why it did not follow its own standards and methodologies.

The Developer Lacked the Expertise to Perform on the Project. Given the pace of change in the computer industry, it is difficult to find experienced personnel to work on a project. In the United States, thousands of computer programmer jobs remain unfilled. The client may observe that the personnel that the developer has put on the project do not seem to have the expertise or experience in the technologies being applied, the programming methodologies, tools, or languages being used on the project. A client is more likely to assert a claim that the developer lacks the expertise to perform the project when the project involves new technologies or new methodologies with which the developer is not traditionally associated. For example, a number of the large systems developers underestimated how quickly the business world would move away from the large mainframe computers and failed to staff up quickly with personnel who were experienced in client/server architecture and the use of PCs and smaller systems. A number of systems development projects suffered through the developer's learning curve during this time period.

The Developer Failed to Allocate Appropriate Resources and Personnel to the Project. A client may argue that the developer was not putting appropriate resources and staff on the project. Resource allocation problems may be more prevalent in projects that are not going well (because a developer's management may feel it is a waste to put more resources on a failing project), and resource allocation problems may be exacerbated where a developer is having its own financial trouble. Developers, on the other hand, are likely to argue that their resource allocations were appropriate, and they will argue that at a certain point, adding more people to a project can only make it worse because it reduces efficiency, creates management problems and makes it more difficult to have all the staff "on the same page."

The Developer Failed to Test the System Properly. At each stage of a development project, a developer is usually required to test portions or particular functions of the system it is developing. Often, however, may elements of a project are going on simultaneously, and it is difficult to test the individual elements and how those elements work together. Nevertheless, when a developer delivers a system that does not appear to function properly in accordance with the specifications (whatever those specifications may be), the client is likely to include a claim that the developer did not properly test the system. As mentioned above, the developer is likely to claim in return that the client signed off on the developer's testing methodologies, the test cases to be used during testing and the results of the testing. In this case, the client and developer will likely dispute their relative knowledge and expertise and the client's ability to effectively understand information provided by the developer.

The Developer's Defenses

The Client Caused Scope Creep. Though the client may be looking for misrepresentations or breaches by the developer, the developer is likely to raise a number of defenses regarding the client's conduct that caused problems on the project. For example, developers often claim that cost over-runs and delays on a project were caused by a client's failure to control "scope creep." Scope creep occurs when a client continually modifies the scope of the project by seeking greater functionality, changing project requirements and otherwise modifying the project after it has already begun. Where a client constantly tried to change the scope or requirements of a project, the developer may have a claim that delays and additional costs are the client's fault. Moreover, scope creep may cause the project to extend so long that the technologies originally identified for the project become obsolete. Obviously, the quality of the documentation of changes to the project scope, the reasons for those changes, and the developer's time and cost estimates relating to those changes will have a big impact on scope creep issues in litigation.

The Client Did Not Accurately Describe its Business Needs. In order to develop a complex computer system for a client, a developer requires a tremendous amount of information from the client about its business and the client's needs. Developers often claim that clients do not provide sufficient and proper information about their businesses and the requirements for the new systems so that the developer can deliver a system which meets the clients' needs. Often, these kinds of problems begin with poor specifications in a request for proposal and become magnified as the project continues.

Client Personnel Were Not Available. As part of most development projects, the developer requires client personnel to be involved in testing the system the developer is creating. This kind of testing is usually called "User Acceptance Testing," and it often requires a significant

personnel commitment by the client to have its people available to test the system and make sure it is usable in normal business conditions. Clients often underestimate how much effort it will require on their part, and how many of its people will need to be devoted to the testing, so that clients may not have enough personnel available to devote to User Acceptance Testing.

The Client Signed Off on the Deliverables. During a large scale development project, developers often present a portion of the project (a "deliverable") to the client for the client's review and acceptance. For example, before a whole system is completed, a developer may present to a client various portions of the system (such as screens which a user might see on the completed system), training materials, system documentation, test plans, business and technical design documentation, and other facets of the system development project. Clients are often required to sign off on these deliverables, but they often lack a full understanding of how the deliverables were developed, how they fit in with the entire project or how well they meet the requirements of the particular deliverable. Developers may argue that because clients signed off on the various stages of the project, they approved the work that was being done by the developer and cannot then claim that the developer was not doing its job.

Conclusion

Careful and thoughtful documentation by both the developer and the client increases the probability that a project will be successful, and if not, good documentation will increase the probability that any dispute resolution or litigation will be successful. Moreover, the claims discussed above demonstrate what is perhaps the biggest problem on any systems development project: lack of communication between the developer and the client. Knowledge of the claims and strategies of systems failure litigation will help both parties avoid litigation.

CHI\127684.1