See You In Court[1]

Karl E. Wiegers
Process Impact
www.processimpact.com

More and more companies are outsourcing their software development projects, often to offshore contractors. Some are replacing custom green-field development with commercial package solutions, which they must customize, extend and integrate into their application environment. Outsourced or package projects that fail too often result in lose-lose litigation. Lawsuits often indicate that the acquirer failed to act on early warning signs, and instead merely hoped for the best.

If you're contemplating asking someone outside your organization to build your next information system or embedded systems product, read on. Here I recommend several practices that just might keep you and your supplier out of the courtroom. I'll use the mythical acquiring company Unfortunato Ltd. to illustrate actual experiences I've observed on contract development and package solution projects.

Define Expectations

The ultimate cause of a project failure is that one—or both—party's expectations are not satisfied. The acquirer has expectations regarding the capabilities and quality of the delivered product and the project's cost and schedule. The supplier has expectations regarding the information and materials that the acquirer will provide. Both parties have expectations about the nature of their collaboration. Unstated expectations can lead to erroneous assumptions, unfulfilled dependencies, unexpected risks and disappointed customers.

To lay the foundation for an effective outsourcing experience, ***document the requirements thoroughly and precisely***. Ongoing, face-to-face acquirer-supplier interactions are ideal, but this isn't feasible when the two are half a world apart. Requirements will change as your mutual understanding changes, so a written specification cannot fully replace regular interpersonal communications. However, many projects stall because of vague, incomplete requirements and the unexpected time that is needed to clarify ambiguities and fill in the holes. To maximize communication effectiveness, have both acquirer and supplier staff participate in requirements definition and review. Avoid implied requirements, which rely on telepathy and clairvoyance—not a sound foundation for requirements engineering. When it comes to outsourcing, if it isn't in the requirements specification, don't expect to find it in the product.

Ambiguity is a major source of requirements problems. Ambiguity means either that the reader can think of various interpretations, or that multiple readers arrive at different interpretations. Don't use intrinsically vague and subjective words in the requirements. Our fictional company, Unfortunato Ltd., wrote a 307-page software requirements specification (SRS) that included the following fuzzy terms:

[1] This paper was originally published in *Software Development*, January 2003. It is reprinted (with modifications) with permission from *Software Development* magazine.

- mostly
- as needed
- might
- make sense
- appropriate
- graceful
- at minimum
- slowly
- may be of use
- including but not limited to
- and/or
- various
- clean and stable interface
- several

Such vague writing is an invitation to unfulfilled expectations. How do you test “several,” estimate “including but not limited to,” or prove that you have a “clean and stable interface”?

Unfortunato’s SRS also was riddled with the word “support,” as in “the system shall support Microsoft Word.” How will the supplier know exactly what Unfortunato means by “support” in each instance? I don’t like to see “support” in a requirements document unless you are building a bridge.

In addition to describing the desired system functionality, ***specify quality attributes, constraints, external interfaces and performance goals***. Sometimes these are collectively referred to as “nonfunctional requirements.” Quality attributes include such characteristics as reliability, usability, maintainability, integrity, testability, installability and portability; see my book *Software Requirements* (Microsoft Press, 1999) for more detail. If you don’t clearly specify your objectives in these categories, the supplier might make design decisions that aren’t consistent with your objectives. Shortfalls in performance, security or reliability might demand a major architectural redesign, although no one budgeted time or money for this additional effort. Failing to satisfy nonfunctional requirements can make the delivered system unfit for its intended use.

Some disputes stem from questions of scope, so it’s essential to ***define the project scope***, the boundary between what’s in and what’s out. One contract conflict arose when, midway through the project, Unfortunato insisted that six additional data conversions were required. The package solution supplier claimed that the contract did not cover those conversions, so they would require additional payment. Now the lawyers are continuing the argument.

Expect some requirements growth on every project, because it’s nearly impossible to know all of them before construction begins. You can expect more scope creep if the requirements are developed hastily, by inexperienced analysts, with inadequate customer involvement or from a poorly articulated initial product vision. Rapidly changing or emergent business needs lead to even more requirements churn and growth; such projects aren’t good candidates for outsourcing.

To deal with the inevitable modifications and growth, ***define a mutually acceptable change control process***. Change always has a price, and your contract should specify who will pay for various kinds of changes, such as new requested functionality or corrections made in the original requirements. The payment responsibility will depend on whether the contract is fixed-price, time-and-materials or royalty-based.

The schedule for Unfortunato's failed package solution project included a one-week task named "Hold requirements workshops for bolt-ons," followed immediately by several tasks to implement individual extensions to the core package. The supplier forgot to include vital intermediate tasks to document, review and revise the requirements specifications. The iterative and communication-intensive nature of requirements development dictates that you must ***plan time for multiple cycles and reviews on requirements***. On this project, the acquirer and supplier teams were a continent apart. They experienced slow turnaround on the myriad questions that arose as the SRS cycled between the two parties. Failure to resolve requirements issues in a timely way put the schedule off track immediately and eventually helped derail the project. An alternative strategy is to take an iterative and incremental approach to the development process, using interim releases to clarify requirements, collect customer feedback and enable course correction.

Open the Communication Channels

An effective acquirer-supplier relationship is a collaborative partnership in which both parties work together in good faith toward their common objective. As with any relationship, a win-win outcome demands honesty and communication. Acquirers must be honest about their expectations and needs. Suppliers must be honest about project status. Each party has the right—and the responsibility—to state what information, materials, conditions and decisions it needs from the other if their mutual effort is to succeed.

To enable effective communication, ***define clear points of contact*** on both the supplier and acquirer sides. The acquirer should allocate the task of managing the supplier relationship to a single individual. Similarly, a supplier should identify one person to be responsible for coordinating the relationship with the acquirer; the supplier should have stipulated this in their response to your request for proposal. The prime communication channel then is through these two subcontract managers. Of course, you'll need other points of contact for contracting negotiations, resolving technical issues and the like. If geography and funding permit, have someone from the acquirer in residence at the supplier's site and vice-versa. This dramatically improves communication, particularly if both individuals are empowered to make appropriate decisions.

Unfortunato's project was delayed because the supplier's staff could not always reach the right people at Unfortunato to obtain necessary information or resolve an issue. Unfortunato identified multiple contact individuals for various functions, who sometimes played volleyball with questions from the supplier. This confusion about who ya gonna call and the ensuing delays made it impossible for the supplier to stay on schedule.

A related issue is to ***identify the project decisionmakers*** and their decision-making process. Many questions and issues arise on every project. A supplier developer might suggest a high-value feature that the acquirer hadn't considered. The supplier might request compromises on stated performance goals or other quality attributes. New risk factors might appear in mid-project. The appropriate people must quickly resolve such issues, so determine who will be making these decisions at the outset. As consultant Ellen Gottesdiener pointed out in her article "Decide How to Decide" (January 2001), the decisionmakers should also select an appropriate "decision rule," be it consensus, voting, delegation to an individual or something else.

Laying a proper foundation for a collaborative project will let you handle most issues amicably and professionally. Nonetheless, you should anticipate how you will deal with disputes and problems that aren't so easily resolved. Your contract should therefore ***establish an issue management and escalation process***. When you encounter a conflict, such as a dispute over project scope, first document the issue and have the subcontract managers attempt to resolve it themselves. Only if that fails should you push the issue up to the attention of higher management levels.

Escalation of unresolved issues also notches up the conflict level, not to mention blood pressures. I read a series of e-mails and faxes between Unfortunato and their supplier that gradually progressed from “You haven’t provided me with information X yet so I can’t proceed” to “Oh yeah? Well, my lawyer can beat up your lawyer!” Try conflict avoidance and conflict resolution before asking big brother to step in, but define how and when big brother will take over when necessary.

My wife and I have long practiced an effective strategy in our relationship, which also applies to software projects: We prefer to ***deal with a concern before it becomes a crisis***. Small issues that remain unresolved can fester into major infections and strain the business relationship. It is far easier to correct a problem when you first recognize it than to let it linger and grow. As software guru Jerry Weinberg observed in *The Secrets of Consulting* (Dorset House, 1985), “It may look like a crisis, but it’s only the end of an illusion,” the illusion that everything is going just dandy.

Keep Your Eyes Open

Recording the requirements and exchanging phone numbers doesn’t mean that you can coast in confidence through the rest of the project. You need visible evidence of the supplier’s progress, so you and the supplier can take prompt corrective action whenever reality doesn’t agree with expectations. To set the stage for meaningful progress tracking, ***agree on significant milestones***. Being a prediction of the future, estimates contain uncertainty, so the supplier might not hit every milestone precisely on time. Include several early milestones to provide visibility into what the supplier is up to. Unfortunato’s project schedule did not include any major milestones until several months after launch. This led to an information black hole, and problems with the early stages of the work did not become apparent until much later.

Every project has a critical path of tasks, which dictates the earliest possible completion date. The supplier’s proposed schedule should ***identify the project’s critical path***, so you can see what tasks will cause the most damage if they slip. The critical path for Unfortunato’s multimillion-dollar outsourcing project was not clearly defined. The Gantt chart lacked many of the chronological dependencies between tasks. This made it impossible to see the real effect of a slip in a vital early task. Consequently, the revised plan that the supplier prepared after missing some early milestones didn’t accurately depict a new, later project completion date.

Risk management is one of the most essential project management practices, so ***manage risks aggressively***. Both the acquirer and the supplier should participate in risk identification and analysis. The items on your top ten risk list should drop off the bottom as you actively mitigate known risks throughout the project. According to Capers Jones’s *Assessment and Control of Software Risks* (PTR Prentice Hall, 1994), the top risks for contract or outsourced software project are:

1. High maintenance costs (60% of projects)
2. Friction between contractor and client personnel (50%)
3. Creeping user requirements (45%)
4. Unanticipated acceptance criteria (30%)
5. Legal ownership of software and deliverables (20%)

Your outsourcing project might well have additional or different risks, so do your own analysis. On Unfortunato's project, the supplier's monthly status reports showed only the same two risk items with low probability of materializing, month after month, as the project spiraled into litigation. Someone wasn't paying attention. See my article "Know Your Enemy: Software Risk Management" (October 1998) for a concise tutorial on risk management.

Keeping your eyes open means that you must ***track status accurately and often***. As Tom Bragg and Rand Allen recommend to suppliers, "If your contract or project authorization document does not require progress reporting, do it anyway!" ("Litigation Avoidance: A Lifecycle Approach," *Cutter IT Journal*, September 2000). The contract should specify the frequency of supplier status reports; weekly or biweekly reports are preferred over monthly reports. Typical report contents include:

- status indicators (green, yellow, red) and summaries for project performance metrics such as requirements status, change management, schedule, budget, resources, technical infrastructure, staffing, and training
- a major milestone table, indicating original planned dates, current planned dates and actual completion dates
- a summary of progress against and deviations from the plan
- a current risk list summary, highlighting major changes from the previous status report
- a summary of current issues and action items that indicates their status, who is responsible for each one, and the target date for resolution

Examine the status reports for warning signals and groundless optimism. Unfortunato's project included a major milestone for delivery of an architecture specification, which was originally planned for March 26. The April 1 status report from Unfortunato's supplier still showed March 26 as the planned delivery date for the completed architecture spec, even though that date had come and gone! These kinds of logical disconnects don't give an acquirer great confidence in the supplier's ability to meet its commitments. On the flip side, one reason that the architecture spec was late was because Unfortunato had failed to deliver critical requirements information to the supplier on schedule, which the supplier should have noted in the status report. Acquirers must fulfill their own commitments.

Expect the supplier to ***replan realistically when milestones are missed***. Unfortunato's supplier issued a revised project schedule midway through the project, but it was completely implausible. They should have pushed the delivery date out, which would have been realistic but would have incurred a contractual financial penalty. Instead, the supplier compressed more and more work into a small time interval late in the project, assigning parallel tasks to many resources who magically appeared for the first time. Unfortunato rejected this revised schedule and cancelled the contract, which ultimately resulted in a lawsuit.

Of course, your goal is not simply to execute the plan, but to achieve a successful delivery of the right product. The plan represents the supplier's best thinking about how to achieve this goal. Plans can—and should—change as reality happens, but you need more visibility into progress than the Bobby McFerrin approach to project tracking ("don't worry, be happy!").

Taste Before Serving

One reason to write requirements is so that you can evaluate whether the supplier's deliverables are acceptable. Therefore, ***define product acceptance criteria and procedures*** during requirements development. Evaluate interim or internal releases for acceptability instead of waiting for the big bang at the end. Acceptance criteria identify the conditions, performance and functions that the product must satisfy before the acquirer will consider it acceptable. Include a first cut at the acceptance criteria and procedures in your RFP, so prospective suppliers know how you plan to evaluate their work. Vague, subjective or undocumented acceptance criteria leave the door open for arm-wrestling over whether the product does or doesn't satisfy the contract. It's a short walk from the wrestling mat to the courtroom.

As Tom DeMarco and Tim Lister pointed out in their article by the same title, when it comes to litigation over software contracts, "Both Sides Always Lose" (*Cutter IT Journal*, April 1998). You can't simply fire off a requirements spec to a supplier and wait for them to ship a perfect product back to you. Solid requirements and project management practices, coupled with effective communication and collaboration, will help keep lawyers from sucking your bank account dry while dueling with your supplier's counsel over a broken contract.

Sidebar: Warning Signs of Outsource Headaches Ahead

Use this checklist to look for icebergs looming on your subcontracted development or package solution project. If you check more than a few boxes, consider yourself at high risk for outsourcing nightmares, project failure or litigation.

- ☐ Regularly scheduled status reports that are not delivered on time, lack expected information or do not jibe with visible signs of progress, such as completed deliverables.
- ☐ Uncompleted action items, unresolved issues, failed dependencies, conflicts that aren't being resolved effectively or other unfulfilled commitments.
- ☐ Unqualified supplier or acquirer staff being assigned, or key supplier or acquirer staff being replaced by other individuals.
- ☐ Acquirer not actively managing and monitoring the relationship with the supplier.
- ☐ Unrequested requirements being implemented or requested requirements being omitted without negotiation and agreement.
- ☐ Scheduled reviews that did not take place, or reviews that should have been scheduled but were not.
- ☐ Decisions not being made in a timely fashion by the right people, or decisions that are not communicated promptly to the affected individuals.
- ☐ Incomplete deliverables received, or contractually required deliverables that do not appear. Documents being received, but no working software being delivered.
- ☐ Processes that aren't working well or are being bypassed inappropriately.
- ☐ Project-tracking trend charts (such as earned-value, defect-detection, defect-closure and requirements-change charts) that don't show signs of completion being forthcoming.
- ☐ Actual cost, schedule, or effort results that deviate significantly from estimates without explanation.
- ☐ Missed early milestones, which don't bode well for completion of future milestones.