See You in Court

Faced with limited resources, 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. Outsourcing, despite its popularity, isn't problem free. Outsourced or package projects that fail too often result in lose-lose litigation. Lawsuits frequently indicate that the acquirer failed to act on early warning signs, instead merely hoping for the best.

By Karl Wiegers, Software Development
Jan 01, 2003
URL: http://www.ddj.com/architect/184414935

Software Development

See You in Court

Faced with limited resources, 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. Outsourcing, despite its popularity, isn't problem free. Outsourced or package projects that fail too often result in lose-lose litigation. Lawsuits frequently indicate that the acquirer failed to act on early warning signs, instead merely hoping for the best.

If you're thinking about asking someone outside your organization to build your next information system or embedded systems product, read on. Unresolved concerns, requirements that leave much to the imagination and the Bobby McFerrin approach to status tracking ("Don't worry, be happy!") are just some of the ways that outsourcing breaks down, but the right practices can keep you and your supplier out of the courtroom. I'll use the mythical acquisition company Unfortunato Ltd. to illustrate actual experiences I've observed on contract development and package solution projects.
Define Expectations
The ultimate cause of project failure is that one—or both—party's expectations aren't satisfied. The acquirer has expectations about the capabilities and quality of the delivered product and the project's cost and schedule, the supplier has expectations about the information and materials that the acquirer will provide, and 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 evolves, so a written specification can never fully replace regular interpersonal communication. However, many projects stall because of vague, incomplete requirements and the unexpected time that's 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.

In addition to describing the desired system functionality, *specify quality attributes, constraints, external interfaces and performance goals.*

1. Come Again?
Our fictional company, Unfortunato Ltd., wrote a 307-page software requirements specification (SRS) that included the following fuzzy terms and phrases: "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 interfaces," and "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"?

2. Say What You Mean
Unfortunato's SRS also was riddled with
Sometimes these are collectively referred to as "nonfunctional requirements." Quality attributes include such characteristics as reliability, usability, maintainability, integrity, testability, installability and portability. (For more detail, see my book *Software Requirements*, Microsoft Press, 1999.) 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 render the delivered system unfit for its intended use.

**What's In and What's Out**

Some disputes stem from questions of scope, so it's essential to define the project scope. Expect some requirements growth on every project, because it's nearly impossible to determine all requirements 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 newly 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.

---

3. **Halfway Home?**

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.

4. **Walk Before You Run**

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...
The iterative and communication-intensive nature of requirements development dictates that you must plan time for multiple cycles and reviews on requirements. 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.

Clear 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, the supplier should identify one person to be responsible for coordinating the relationship with the acquirer; the supplier should stipulate 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.

Identify the Project Decisionmakers
On every project, many questions and issues arise. A supplier developer might suggest a high-value feature that the acquirer hadn't considered, or 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 you should determine who will be making these decisions at the outset. As consultant Ellen Gottesdiener pointed out in her article, "Decide How to Decide" (Jan. 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'll 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 pressure.2

My wife and I have long practiced an effective strategy in our relationship that 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 any relationship, business or personal. It's 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.

6. Leave a Message, Again
Unfortunato's project was delayed because the supplier's staff couldn't 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 "who ya gonna call" confusion and the ensuing delays made it impossible for the supplier to stay on schedule.

7. Count to 10
The e-mails and faxes between Unfortunato and their supplier 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 resolution before asking big brother to step in, but define how and when big brother will take over if necessary.

8. Missing Milestones
Unfortunato's project schedule didn't 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 didn't become apparent until much later.
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-10 risk list should drop off the bottom as you actively mitigate known risks throughout the project. According to Capers Jones' 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.

See my article "Know Your Enemy: Software Risk Management" (Oct. 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, Sept. 2000). The contract should specify the frequency of supplier status reports; weekly or biweekly reports are preferred over monthly reports. Typical 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.

9. Slip-Sliding Away
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, making 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.

10. Devil in the Details
On Unfortunato's project, the supplier's monthly status reports showed only the same two risk items with a low probability of materializing, month after month, as the project spiraled into litigation.

11. Deadlines and Disconnects
Unfortunato's project included a major
Examine the status reports for warning signals and groundless optimism.\textsuperscript{11}

Expect the supplier to \textit{replan realistically when milestones are missed}.\textsuperscript{12}

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.\textsuperscript{13} Plans can—and should—change as reality happens, but you need more visibility into progress than the Bobby McFerrin approach provides.

**Taste Before Serving**

Writing requirements will help you to evaluate whether the supplier's deliverables are acceptable. Therefore, \textit{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" (\textit{Cutter IT Journal}, Apr. 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.

---

\textsuperscript{11} Examine the status reports for warning signals and groundless optimism.

\textsuperscript{12} Expect the supplier to \textit{replan realistically when milestones are missed}.

\textsuperscript{13} 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 provides.

**Taste Before Serving**

Writing requirements will help you to evaluate whether the supplier's deliverables are acceptable. Therefore, \textit{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" (\textit{Cutter IT Journal}, Apr. 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.

---

\textsuperscript{11} Examine the status reports for warning signals and groundless optimism.

\textsuperscript{12} Expect the supplier to \textit{replan realistically when milestones are missed}.

\textsuperscript{13} 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 provides.

**Taste Before Serving**

Writing requirements will help you to evaluate whether the supplier's deliverables are acceptable. Therefore, \textit{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" (\textit{Cutter IT Journal}, Apr. 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.
The Deadly Dozen: A Checklist

Avoid outsourcing headaches by heeding these 12 warning signs.

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 a high risk for outsourcing nightmares, project failure or litigation.

- Scheduled status reports are late, spotty or do not jibe with visible signs of progress.
- Uncompleted action items, unresolved issues, failed dependencies or festering conflicts are multiplying.
- Unqualified supplier or acquirer staff are assigned to the project; key supplier or acquirer staff are replaced by others.
- Acquirer is not actively managing and monitoring the relationship with the supplier.
- Unrequested requirements are implemented; requested requirements are omitted without negotiation and agreement.
- Scheduled reviews do not take place; reviews that should have been scheduled, aren't.
- Decisions aren't made in a timely fashion by the right people; decisions aren't communicated promptly.
- Incomplete deliverables are received, or contractually required deliverables—working software along with docs—don't appear.
- Processes aren't working well or are bypassed inappropriately.
- Project-tracking charts for earned value, defect detection, defect closure and requirements changes are missing in action.
- Actual cost, schedule or effort results deviate from estimates-without explanation.
- Early milestones are missed.