



March 2005

Outsourcing Software Development: A Case for Effective Scope Management

by Kal Toth, Oregon Masters of Software Engineering, Portland State University



Abstract: Software engineers should leverage their technical know-how and process-oriented mindsets to help their companies control the scope of outsourced projects, especially off-shore. This is the 2nd article in a series on professional development programs.

Introduction

Old war stories sometimes ring true when you least expect. For a few years now, I have been using a case study to convince software engineers and managers about the need for effective control over project scope and to show them how this relates to relationship building with the customer, senior management and the software development team. Without effective scope control, project objectives, software quality, team productivity and customer acceptance are all put in jeopardy.

Since Edward Yourdon's "Decline and Fall of the American Programmer" in the early '90s and the bursting of the dot-com bubble in the late '90s, software technology companies have flirted, then dabbled, and now are embracing the outsourcing of software development projects to off-shore companies in India, China and Eastern Europe. Many have expressed alarm about lost jobs and the potential loss of American dominance in the information technology sector.

I generally agree that companies are thereby exposing both project success and their intellectual assets to certain risks. Companies need to consider taking counter-measures that will protect their controlling positions while leveraging lower-cost off-shore technical labor to make their products more competitive. And American software engineers and developers need to urgently position themselves and help their companies in this regard. Software engineers need to take advantage of their foundation technical knowledge and become truly effective software managers to make a real difference and increase their value to their companies.

The case study

The story I have used involves a customer and a software development company engaged in a contracted project over about a year. The customer and contractor had a history of previous projects and enjoyed mutual respect and confidence before entering into the contract. The required functionality for the project was broadly understood by both parties and they agreed to work very closely together to build the “right system.” The company had achieved success through great communications, highly skilled developers, great stock options and a very agile team. They appeared to have adequate software development competencies to build the system “right” – they also had access to highly skilled independent consultants and specialists who could jump in if necessary. And they enjoyed the advantage of being located in the same city and the customer had several domain experts that could be made available to the project for extended periods of time. Indeed, the ingredients for project success appeared to be firmly in place.

However, neither the customer nor the contractor saw a need to address unforeseen risks and to put appropriate competencies and processes in place to govern their working relationships during the lifetime of the project. And the company’s senior managers were not aware of the pitfalls of ignoring this aspect. As it turns out, previous projects were smaller, less critical, not fixed price and the company’s stock option plan was not nearly as attractive as it used to be. Needless to say, this project ran into a series of escalating difficulties that caused the project to fail and resulted in lost opportunities and reduced credibility for all parties involved.

So, how does this relate to outsourcing?

I expect that most readers will agree that contracting out software development is a form of outsourcing, and that outsourcing to off-shore software technology companies is a special case. All other factors being equal, it is not a stretch to postulate that the risks of off-shore outsourcing are much higher than the risks of contracting to a company in your own backyard. Language, time-zone differences and culture can get in the way of ongoing communications and collaboration, and due diligence to confirm the quality and reliability of distant contractors before entering into firm deals is fraught with uncertainty. Nevertheless, the potential savings on the bottom line cannot be ignored and companies will be drawn by competitive pressures to contract to off-shore software development shops.

Given that the risks appear to be sufficiently high, the prudent outsourcing technology company should be no less careful about contracting off-shore than the American customer contracting software development projects to American software companies. Hence, effective strategies for contract management of projects should also guide software technology companies, their software engineering managers and their senior managers that approve their budgets and demand accountability.

The problem of managing project scope seems to be particularly relevant to outsourced contracts and development agreements. Just like their customers, outsourcing companies need to be able to reliably govern scope within budget and schedule constraints while being flexible enough to deal with changing project priorities and product needs. Companies also need to protect their intellectual property – a topic for a future article perhaps.

Outsourcing success factors

So this takes me back to my case study. The unfolding events of this story progressively convince the reader that processes and know-how for controlling scope are critical, must be planned from the beginning and must be proactively supported along the road

towards successful completion of a project's objectives. In the case of software development outsourcing, however, the contractor has two relationships to manage: the relationship with the customer (and the customer's users) and the relationship with the off-shore software development shop – the “subcontractor.” Although this complicates things for the contractor sandwiched in the middle, the principles of controlling scope are nevertheless relevant to all parties involved. Scope control principles endeavor to maintain win-win-win relationships between the three parties involved. The following success factors, therefore, are critically applicable to both the relationship between the customer and the contractor, and that of the contractor and the off-shore software development subcontractor.

Early awareness

Early discussions between contracting parties should lay out the necessity of managing scope, pointing out the risks of lost quality and project failures. The parties need to agree on a mutually agreed upon processes that they can all buy into.

Infeasible requirements

Yes, these could blow out both the schedule and budget. But thorough feasibility analysis and needs assessment early in the project should convince the customer that infeasible requirements should be waived, simplified or traded off against other requirements that may emerge. Outsourced development will require very close cooperation and coordination between the contractor and the off-shore subcontractor to ensure that the project is feasible and real needs are being addressed.

Breadth versus depth

When it comes to project scope, it is often useful to distinguish between breadth and depth. The customer in the business relationship is responsible for defining the initial requirements which are virtually always expressed at a high level – typically they are both ambiguous and incomplete. Nevertheless, these broadly stated requirements represent the “essential requirements” for the project.

Classification of changes

Proposed changes can be made, of course, to both breadth and depth of scope.

Changes that come along either clarify the broadly stated essential requirements, refine them, change them or add to them. Added requirements clearly will impact on effort and schedule – and trade-off negotiations may need to be pursued. Changed requirements should launch evaluation and negotiation if they affect effort or schedule.

The trickier aspect of this is distinguishing between clarifying information and additional information. Clarifications will positively affect project budgets and schedules but refinements may add to project scope. A critical role for requirements analysts is to anticipate implementation impacts at each turn and distinguish between clarifications and refinements.

On prioritization of emerging requirements

As a project unfolds, everyone involved (including analysts and developers from all parties) gets a better idea of what needs to be done and of possibilities for meeting expectations. But not everyone will have the same idea of what is essential when all is said and done. If it is critical to have the project delivered on time or within budget we need to ensure that there is a common understanding of this across all members and all teams. For this reason, it is important to be quite explicit about what is more important, and what is less-so. Attaching priorities to requirements will help achieve this and will support downstream trade-off decisions.

Here is a good bit of systems thinking that I have bumped into in the past. Functions and features identified during customer and user elicitation sessions should be tagged as “essential,” “desirable,” “nice-to-have”/“optional” or “TBD.” By mapping every function to the initial (broad) statement of requirements, it is possible to identify these as “essential,” unless the customer or user decide otherwise (“not as important as we originally thought”). Functions and features that do not trace explicitly to the agreed requirements or appear to refine rather than clarify should be annotated as “desirable” or “nice-to have,” pointing out that they appear to be less important and potentially out of scope (not part of the original agreement). Any features discovered in later elicitation

sessions should be automatically tagged as “desirable” at best or “nice-to-have.” If agreement can’t be achieved during an elicitation process, then these should be categorized as “unknown” or “TBD” and tabled for further discussion and negotiation at higher levels of decision making.

In summary

Anything that pushes the project outside the “essential breadth” of scope or beyond the highest project priorities should be questioned as being potentially out of scope and subject to negotiation (trade-offs, waivers, changes in effort and schedule, etc.). In my opinion, the secret to convincing the customer on trade-offs that will ensure joint success is driven by a critical qualifying factor – namely, impact assessment estimates. This is to say, that whenever we need to make decisions between what is in and what is out, we need to estimate the impact of functions, features and other attributes on budget and schedule. Positive assurances about meeting budget and schedule will comfort all parties. This will enable the customer and the off-shore subcontractor to appreciate the need for, and benefit of, good trade-off assessments, in particular, trade-offs between prioritized requirements – essential, desirable and nice-to-have.

About the author

Dr. Kalman (Kal) Toth, Associate Professor in Portland State University's Department of Computer Science, is the Associate Director of the Oregon Master of Software Engineering (OMSE) program. He delivers courses on software engineering principles, software project management, software design techniques and the program's software engineering practicum. He has over 25 years of technical, consulting and management experience in large and small technology companies including Hughes Aircraft, Datalink Systems Corp., BC Software Productivity Centre, and the CGI Group Inc.

He may be contacted at ktoth@cecs.pdx.edu or see <http://www.cs.pdx.edu/~ktoth/>.

For more articles visit www.sao.org/newsletter.