TRIGENT

Insights

# How to Succeed in Outsourcing Application Development

November, 2003

*How should companies go about outsourcing the development of software applications?*

*In this issue of* Insights*, we evaluate various processes employed in establishing an outsourced development funciton, examine some common misconceptions, and make recommendations for reaching a successful outcome.*

Overcoming Limits

The process of outsourcing software development offshore has come a long way since the mid-to-late '90s, the heyday of Y2K remediation. Offshore vendors who cut their teeth on Y2K and legacy applications developed best-in-class quality processes, and evolved to look more like the large U.S. based systems integrators. At the same time, thousands of new services firms entered the market to serve the needs of specific industries, address particular business processes, or offer specialized technical skills.

Today, Gartner Group estimates that by 2004, more than 80 percent of U.S. executive boards will have discussed offshore sourcing, and more than 40 percent of U.S. enterprises will have completed some type of pilot or will be sourcing IT services through a global delivery model, such as near-shore and offshore.

The offshore outsourcing marketplace today has something for everyone – if you know what you want, and how to get it. While the market is more mature, there are still pitfalls – even for software companies, which are in the business of developing software and delivering IT services. So let's take a look at some of the common approaches to outsourcing product development and some "Do's" and "Don't's" that can guide you through the various stages of your project and ensure its success.

## Getting Started: Laying the Groundwork for Success

Most people will agree that you can't just define a project and "throw it over the wall" to an outsourcer. Yet there is a common misconception that you start out by preparing a product spec for outsourced development.

There is no document, however comprehensive you think it is, that you can just hand to an outsourcer and be sure you will get what you need. The actual, functional product spec, the document used as the basis of design, will be the result of a process of progressive investigation and analyses, often marked by a sequence of documents:

> A Request for Proposal (RFP)

> A Proposal or Statement of Work (SOW)

> A business requirements specification and a functional specification

Outsourcers will expect to go through this process, and so should you. You may be experienced in writing product specs for your own development group. However, successful outsourced development involves more than defining a product spec. Success requires *careful definition of how your company and the outsourcer interact – and the right spirit governing those interactions.*

First, your goal, and the outsourcer's, should be to make your initial project a success. This goal is a prerequisite for meeting your other goals for outsourcing, such as cutting costs or getting to market faster. One way to succeed is to choose the right kind of pilot project, one amenable to success, so you can forge a relationship and learn how to work most effectively together. Good projects for successful pilots involve minimal risks to the business strategy or core product. They are relatively easy to define and break down, and have few dependencies. Examples include product enhancements, testing, API's, interfaces, or bug fixing.

Whether your project is simple or complex, to succeed you need to communicate – not only about the daily details and tactical activities, but, at the executive level, about strategy. Understandably, many companies are reluctant to reveal business strategy. But remember: the more an outsourcer understands the context, the better prepared it will be to solve problems, offer recommendations that might get you more for your investment, and help you to achieve your long-term goals.

Here's a true story. A software company hired an outsourcer to develop a "prototype," with a stated goal of attracting investor funding. However, the company also planned to use this prototype for product testing among potential users, a goal that was never communicated to the outsourcer. The outsourcer focused on delivering key features for a demo to investors. Because of this failure to clarify the project goal, the software delivered did not have the performance and scalability necessary for user testing.

With this in mind, here is a look at each stage and related documents in the process of selecting and working with an outsourcer – with an eye to what can hinder or help in the quest for success.

## Engaging an Outsourcer: To RFP or Not to RFP

RFP's ask for the outsourcers' qualifications, describe the business and product requirements, and require a project timeline and price estimate. With every candidate answering the same questions, an RFP seems like a fair way to judge everyone and a good way to get the best price.

However, if someone underbids but can't deliver the work, or if the product delivered doesn't meet the true business need, you don't save any money or get to market faster.

As a result, here's what NOT to do:

> Write an RFP as your product spec – and send it out to a dozen outsourcers

> Once the RFP is written by the people who want and will use the software, turn the RFP over to another group, for example, Purchasing, to evaluate responses and chose a vendor based on cost and, perhaps, a few other criteria, according to the company's guidelines.

What's wrong with this process? Many outsourcers will not commit the resources to respond to RFPs – for a couple of important reasons:

> First, RFP's don't provide information enough to write a meaningful resource and cost estimate. While many companies consider them to be a product spec, from the outsourcer's point of view, RFP's contain only 50% to 75% of the real requirements – at best. At worst, they contain errors in estimating the size of the work and the skill levels required.

For example, RFP's may contain necessary information about desired features – without considering whether and how engineers can build them. Detailed information – combining users' and engineering perspectives – will only emerge after an outsourcer prepares a SOW, is selected, and gathers more information.

> Second, if the process of managing RFP responses is handed over to another group, often the stakeholders who generated it are not available for discussion with the candidates. Good outsourcers will want to talk to the stakeholders to find out information that is necessary for them to write a realistic proposal.

There are alternatives. If company policy mandates an RFP, you can research and select a discrete group of potential outsourcers. Then make sure candidates can have access to a stakeholder for clarifications as they prepare their response to the RFP. However, the candidates may still not have complete enough information to make a precise estimate of the time and cost of the project. So in selecting the least costly response, you may not be selecting the right outsourcer.

The best route is to find two or three appropriate outsourcing vendors, based on your specific business needs, such as technology skills, industry specialization, local presence, etc. Once you've selected an initial group, provide them with a high-level outline of business and functional requirements, and then *work with them* as they prepare for the next step, a Statement of Work (SOW).

Why not ask for a proposal at this point? A SOW includes the outsourcer's proposal and also contains all the legal details of the potential agreement – for example, the terms for licensing, confidentiality, intellectual property rights, and liability issues, to name a few. This inclusiveness allows you to negotiate all issues at once.

A SOW is also more substantive than an RFP. In a SOW the outsourcer describes what it will offer to address your needs (in the solution description) and how it will work with you (such as delivery mechanisms, schedules, and acceptance procedures). By selecting an outsourcer after you see the SOW, you will be making a judgment based on more evidence.

## The Initial Statement of Work (SOW)

The SOW should start with high-level descriptions of the business need and the outsourcer's solution (including a breakdown into components and platform requirements). It should also describe delivery mechanisms, acceptance criteria and procedures, and, of course, a timeline and price for delivering the solution. A SOW also covers legal, contractual issues such as ownership of property, terms and conditions of payment, limits of liability, etc.

The SOW will tell you a lot about the outsourcer besides its price. It will show how it will work with you. The more detail about how the product will be delivered, tested, and accepted is agreed on at this point, the fewer unpleasant surprises you'll receive in the future. In addition, here are some key points to look for:

> <u>Project description and scope</u> The SOW, or documentation accompanying it, should explicitly describe what the solution WILL NOT cover – to set realistic expectations. Examples include onsite installation and testing (will code be installed and tested at your offices or only delivered to you on a CD?) as well as application maintenance, which is generally not included in a development project unless expressly requested.

**Allocating Time for Project Phases**

| Phase | % of Total Project Time |
|---|---|
| Requirements analysis | 10% |
| Specifications | 20% |
| Coding and testing | 40% |
| Integration and system testing | 25% |
| Acceptance testing | 5% |

> Delivery and deliverables You'll want to see a definition of each deliverable, a description of the delivery process, and a specific delivery schedule. The description of the delivery process should define how and where the deliverable will be made (for example, deployed onsite or remotely) and packaged. It should also describe a high-level quality assurance process that will be completed before delivery. For example, there will be three rounds of system testing and less than a specified number of defects before the software is considered deliverable.

> Costing and resource scheduling The SOW should break down what type of resources will be employed at each phase of the project (e.g., developer, architect, quality assurance manger, project manager, etc.), and how much time they will spend. If a SOW lacks this detail, it will be difficult for you to determine whether adequate time and skills are assigned to major activities. Hence, a low estimate may be due to the fact that an outsourcer only allotted eight hours for acceptance testing with a junior developer, while another allotted 40 hours and included both development resources and a QA manager. How do you know enough time has been allocated to specific phases? Some rules of thumb: the Requirements Analysis phase requires approximately 10 percent of total project time, with Specifications comsuming another 20 percent. Coding and testing generally comprise about 40 percent of the project, with Integration and System Testing adding 25 percent, and Acceptance Testing the final 5 percent.

In addition, you should expect the price given in the initial SOW to include a firm figure for a discovery or analysis stage and an *estimated* cost for the design, development, and deployment phases, which will be finalized following the discovery phase.

> Milestones and acceptance criteria For all project points at which there is a deliverable, the SOW should:

> **Identify milestones that will be used to ensure the work stays on track** and meets project and business goals.

> **Define *measurable* acceptance criteria**. Acceptance should not hinge on a subjective judgment such as "I don't think it does enough."

> **Specify roles and responsibilities to assure accountability**. Be sure to designate someone as an "acceptor" responsible for making a timely decision. Also have a specific role on both teams to manage communications.

> Change control Provide a mechanism for bringing together both parties to discuss any changes to the requirements – for a fair discussion of how the change will affect the schedule and the price.

## Business Requirements and Functional Specifications

Based on the qualifications of individual outsourcers and their SOWs, you will choose a vendor to complete your project. At this point, most outsourcers will conduct some level of discovery or requirements analysis, talking to end users as well as management. Whether the outsourcer validates your assessments or corrects them, its independent judgment will help assure the project can and will address the true business needs.

Based on this analysis, the outsourcer may revise the SOW – including the price for the design, development, and deployment phases – and write up business requirements and functional specifications. The business requirements tell what the product does. The functional spec details how it does it.

The business and functional requirements are generally combined in one document, with specific functional requirements for even the most discrete technical elements tied to the specific high-level business requirements they serve. The goal is to ensure developers are grounded in the business context. By understanding the business reasons for what they are doing, they can foresee problems and devise better solutions. Although specified personnel, such as a project manager or architect, are responsible for ensuring the software serves the business need, linking the technical and business requirement in writing keeps everyone on track.

With this in mind, here are some of the topics a good requirements document, combining business and functional requirements, should contain:

**Business Rationale:** This spells out the business problem to be addressed and defines and enumerates both the business requirements necessary to solve the problem and the business objectives. Business requirements include new system and user activities such as "will generate usage reports," "will provide user authentication functionality," or "will allow designated users to perform a named function or action." Examples of business objectives are "will decrease the lead-to-sales-closing time," or "will reduce the maintenance costs of the service center."

**Technical Rationale**: This outlines the hardware and software environment, platform considerations, any existing system information, and any known limitations and/or technical problems. It notes system functions and components, including changes and additions to existing software. Examples might include report, data, and security components. Also called out are out-of-scope features and exclusions, assumptions, and dependencies. For example, "the software must be accessed over the Internet," or "the solution assumes that user authentication is already in place." To be sure you get what you want – *be sure this information is defined*.

**General Requirements:** This section defines functional requirements and non-functional requirements.

> <u>Functional requirements</u>. Each system function or component listed in the technical rationale is broken out and explained. For example, for a report component, the description would name the reports generated by the solution.

> <u>Non-functional requirements.</u> This section describes the tools and technologies used by the proposed software, its hardware and software environment, and number of users and concurrent users. It also defines critical system characteristics such as performance (with metrics), integrity ("data for reports must be complete and valid"), and portability ("the software must comply with J2EE v1.5"). Characteristics also include what the new software should NOT do – for example, "it will not affect existing system X."

<u>Specific Requirements</u>: The more you break down the software and define how each part works the better. Details should include the purpose of each function, inputs and outputs, interaction with other systems, designated users, user interface, and operations broken down step-by-step – with their related screen shots. This is also the place to map each discrete technical feature to the non-functional requirements, business requirements, and business objectives it supports.

Other Sections or Appendices: In the “Specific Requirements” or in separate sections or appendices, the document should provide, if relevant, use cases and diagrams, lists of user roles and access privileges, and an explanation and diagram of the logical architecture.

Although creating and agreeing upon the terms of this document may seem to take a lot of time before a single line of code can be written, the result will be a detailed working document for the developers.

Finally, once you have ensured that the SOW, the business requirements, and the functional specification incorporate the suggestions made here and meet your standards, it’s time for sign off. If you and the outsourcer have had a shared understanding, you have the roadmap for successful design, development, and deployment. As long as you keep communicating, you will reach your destination.

## About Trigent

Trigent provides focused software solutions designed to help business and IT executives overcome limits - of technology, of resources, and of time - using a blended global delivery model.

## Trigent Locations

1 Apple Hill, Suite 204
Natick, MA 01760 USA
+1 (508) 651 5900

Khanija Bhavan
First Floor
49, Race Course Road
Bangalore 560 001 India
+91 (80) 2226 3000

**www.trigent.com**

TRIGENT