Controlling Software Acquisition Costs With Function Points and Estimation Tools

Ian Brown
Booz Allen Hamilton

Too often, organizations that contract for software development services are at the mercy of vendors for cost and schedule estimates. Once a program office releases a request for proposal (RFP) for software development, it must somehow evaluate the validity of cost and schedule estimates that come back with the proposals. Or, a program might have a limited budget or schedule but not a clear understanding of what amount of development is actually feasible within these limitations. This article proposes an approach that can help buyers of software take control of this situation by providing the ability to objectively evaluate software development proposals, select the best value for their needs, and effectively manage acquisition costs from kick-off to product delivery.

Just a few years ago, purchasing a new car was often a lopsided affair. A buyer might know what kind of car he wanted to buy and how much he could afford, but the sellers held all the cards because they controlled the situation with information. They knew cost details – sticker price, invoice, incentives, kickback numbers – all of which provided them tremendous advantage in the transaction. They had information on how specific features were priced and which ones generated the most profit. A buyer might ask for power seats and windows, a CD player, antilock brakes, and passenger side air bags, and the salesperson might give a price of $5,000. How was the buyer to know whether or not that price was too high or if it was a great deal? And what if the seller offered to throw in the special undercoating and super-absorbent floor mats – which the buyer does not need – for *free*? It was very difficult for a buyer to understand if he was getting the features for which he asked and needed at a fair price. He might be able to shop around, but in the end, not having good information as a point of reference, it was difficult to assess if the transaction was fair.

These days, however, information is more readily available for car buyers. Car pricing internet sites have become valuable sources of information for consumers. Car buyers can now prepare more effectively for the acquisition process by arming themselves with independent, comparative cost information *before* the assessment and negotiation activities begin. Overall, consumers are much more likely to be able to buy the car of their choice – with the features wanted and needed – at a fair price.

In many ways, acquiring software or software development services compares to the *old way* of buying cars. Access to information is rarely equal, and it typically does not favor the buyer. Once an RFP for software development is released, a program office can become completely dependent upon vendors' estimates of cost and schedule. Or, a program might have a limited budget or schedule but not a clear understanding of what amount of development is actually feasible within these limitations. When assessing proposals from vendors, programs are faced with several of the following questions:

- Have we been offered a reasonable price?
- Has this project been deliberately underbid?
- Is the proposed schedule realistic?
- How do we know we are getting the functionality we have asked for and need?

With these kinds of uncertainties, how can a program make informed decisions when purchasing software or software development services? A program must take control of the situation to more effectively assess whether submitted proposals are realistic while having a clear understanding of what functionality should be included in the delivery.

The purpose of this article is to provide an approach that can help buyers of software objectively evaluate software development proposals, select the best value for their needs, and effectively man-

age acquisition costs from kickoff to product delivery. The foundation of this methodology is the ability to objectively size the developed software and to understand the potential ranges of cost and schedule that could result.

This article proposes a particular methodology to estimating software development cost and schedule *in the context of independent evaluation of vendor proposals*. It is not the only valid software estimation methodology available to organizations, but experience has shown that this specific methodology is very well-suited for this particular situation, for reasons that are discussed later.
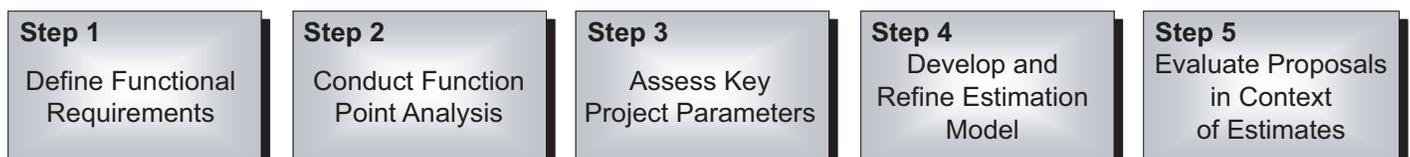
Methodology

The following five-step approach (Figure 1) is designed to be as objective as possible:

- **Step 1:** Define Functional Requirements.
- **Step 2:** Conduct Function Point Analysis (FPA).
- **Step 3:** Assess Key Project Parameters.
- **Step 4:** Develop and Refine Estimation Model.
- **Step 5:** Evaluate Proposals in Context of Estimates.

Generating cost and schedule estimates without intimate knowledge of a development organization's historical performance can be extremely challenging. This methodology combines standardized software measurement techniques with structured, well-documented estimation tools to enable true independent estimation. The goal is not to produce the *right* answers in terms of cost and schedule, but

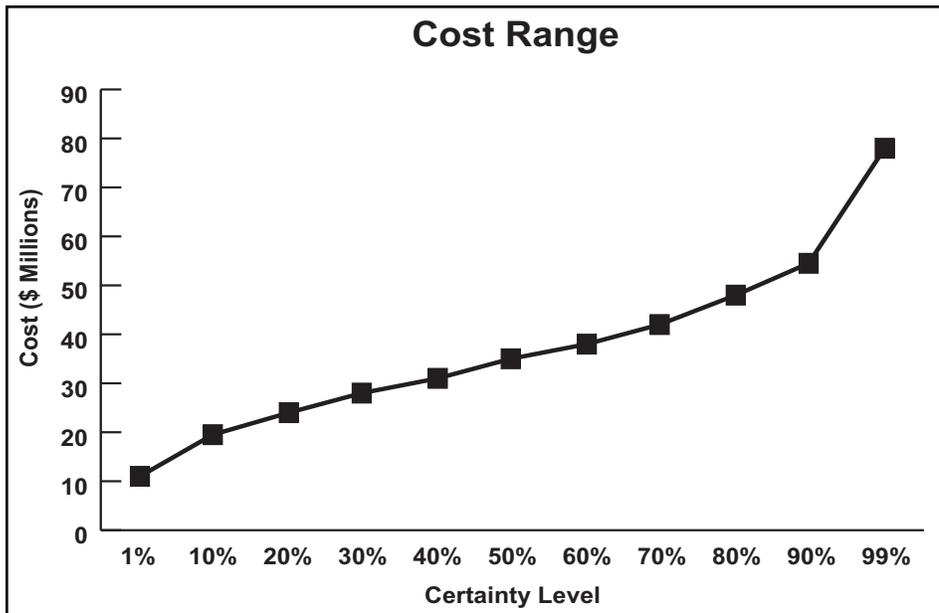Figure 1: *Software Acquisition Cost and Schedule Estimation Framework*

Step 1	Step 2	Step 3	Step 4	Step 5
Define Functional Requirements	Conduct Function Point Analysis	Assess Key Project Parameters	Develop and Refine Estimation Model	Evaluate Proposals in Context of Estimates

May 2007

www.stsc.hill.af.mil **9**

Cost Range

Figure 2: *Cost S Curve*

rather to understand what a reasonable range of answers might be and how vendor responses to a RFP (typically submitted as point estimates) fit within that range. Additionally, this methodology can help a program understand the relative cost and schedule risk it accepts by selecting one proposal over another.
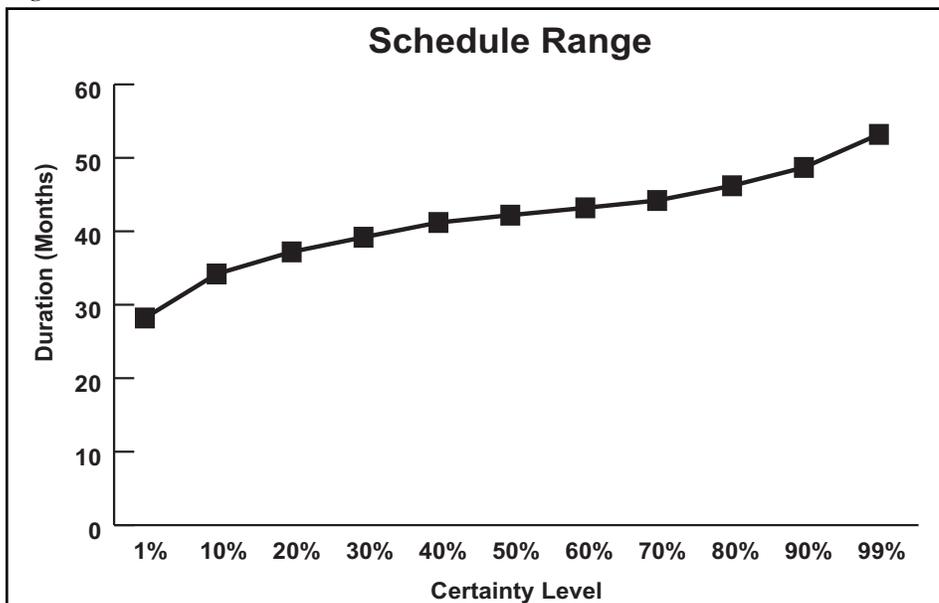
Step 1 – Define Functional Requirements

Although this article focuses on software estimation for acquisition, requirements definition must be included as a critical step. Functional software requirements have to be defined, documented, and baselined. This is an essential foundation of the acquisition process. Requirements fidelity is key to ensuring that vendors understand what must be delivered.

Figure 3: *Schedule S Curve*

Schedule Range

Requirements documentation should be provided to all potential bidders along with time for review and clarification. Requirements definition should happen before anything else. The importance of this step cannot be over-emphasized, as requirements are an integral part of nearly every aspect of delivery of the project, and they are necessary to conduct an initial FPA.

Step 2 – Conduct FPA

In order to estimate the cost and schedule of a development effort, one has to know the size of the intended software. Software size, too frequently overlooked in estimation exercises, is often expressed in source lines of code (SLOC). Many organizations have success with this size measure as a basis of estimate, but this

measure has some inherent difficulties associated with it, especially in the context of the methodology proposed here [1]. Different organizations count SLOC differently – there are no industry-defined standards that identify what should be counted and what should not. This makes an accurate, independent assessment of SLOC size difficult to produce. So, although it is a valid measure of software size, SLOC does not lend itself to the nature of this particular type of estimate and independent analysis.

Function points, on the other hand, are a standardized unit of measure as recognized by the International Organization for Standardization 20926:2003. The function point standard is maintained by the International Function Point Users Group (IFPUG) in a voluminous counting practices manual; IFPUG offers a certification program that recognizes experts in the field as certified function point specialists. Function points measure software size independently of technology, platform, or programming language. In short, function points objectively define the size of an application that is to be developed based on defined functional requirements. They can also help identify gaps in requirements analysis, avoiding early introduction of defects [2].

To take this a step further, if an FPA is conducted prior to releasing the RFP, the results can be provided to all interested vendors to provide a common assumption of size so that all bidders can work from consistent information in developing their responses.

Step 3 – Assess Key Project Parameters

Key project parameters define characteristics of important cost and schedule drivers for the development effort. These parameters include high-level assumptions, such as the platform, programming languages, application type, reuse, development standards, commercial off-the-shelf (COTS) use, and staffing approach. If known, more precise parameter values for specific product and performance attributes can be identified in order to tailor the estimate to more specific project characteristics, such as development environment, personnel skills, organizational process maturity, security requirements, and system volatility. These inputs should be expressed as ranges (low, middle, high) to account for uncertainty and potential variation among bidding organizations. The less that is known about these more specific factors, the wider the range of assumptions should be.

Step 4 – Develop and Refine Estimation Model

The outputs of steps 1 through 3 are relatively meaningless on their own. Only when they are combined as inputs to an estimation tool do they produce relevant, useful information. In this type of estimation exercise, leveraging a parametric tool to generate cost and schedule estimates is particularly important. Other estimation methodologies (analogy, wideband Delphi, cost estimating relationships, etc.) do not provide the flexibility needed to establish this proposal assessment framework. An estimation tool provides the flexibility to apply generalized assumptions where necessary and specific assumptions where appropriate. A trained, experienced user should be involved to make sure that the tool is used properly and the results are interpreted correctly. This methodology does not lend itself strictly to COTS acquisition, as most estimation tools are best suited for estimating effort on projects with custom development.

The model should be constructed with a work breakdown structure (WBS) that reflects the components of the software. For example, if the application will have a user interface with an Oracle back-end, then these two components should be modeled as separate WBS elements in the parametric tool. The WBS should also reflect any expected modular or incremental development strategies that might be proposed. Function points should then be allocated to the appropriate WBS elements or increments as accurately as possible.

Step 3 identified ranges of key project parameters. Applying these inputs to the estimation model will generate cost and schedule ranges with corresponding probabilities. Estimates in this form can be expressed as S graphs and are the linchpin of this evaluation structure. Using ranges in this way, as illustrated in Figures 2 and 3, provide the context and framework for evaluation of different bids from different vendors.

The estimates produced by the tool, however, should not just be accepted without consideration. Analysts should apply cross-checks, known analogies, or expert opinions to test the outputs for reasonability. The estimates should also be compared to the expected or known budget or schedule for the project. Ideally, those numbers will fall somewhere within the estimated ranges, preferably in the higher certainty levels. However, if the budget or schedule falls outside of the relevant range, the program office should review the model and key project parameters. Oftentimes this exercise can highlight
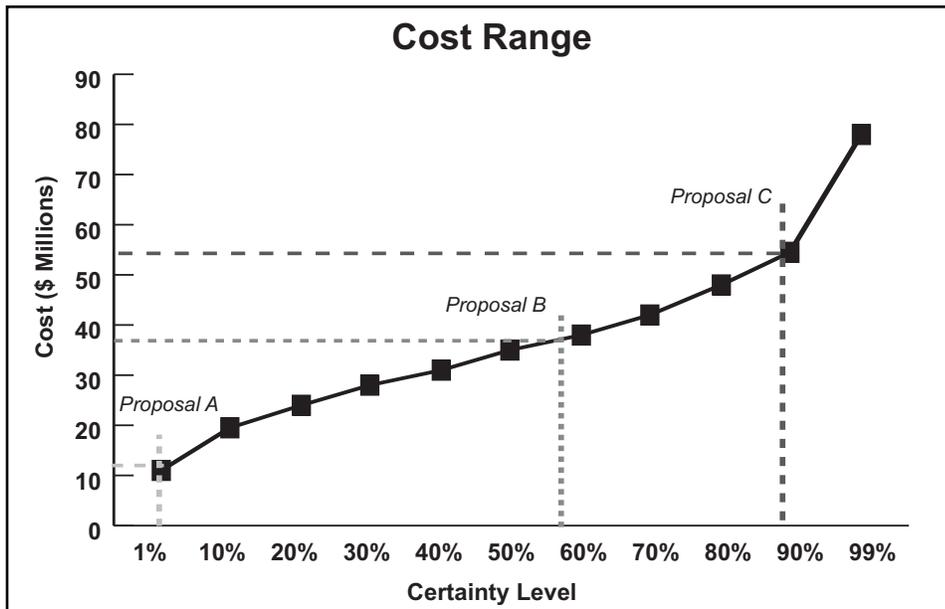
Figure 4: *Cost Evaluation Framework*

some assumptions that might be incorrect. For example, the initial model may have assumed the project would be staffed to minimize the development schedule. This staffing approach generally increases effort and cost (as well as the associated risk), so if the estimated range is too high for the known project budget, perhaps the project should be modeled to optimize the effort (resulting in fewer staff, lower cost, and longer schedule).

If this review *still* results in an estimated range that does not include the necessary budget or schedule, then the project scope must be evaluated. Too often, a program office will set a project up for failure by demanding that the full set of software requirements be developed within a budget or period of performance that is simply unrealistic and unattainable. This methodology can help to avoid these situations by raising a red flag early in the process. Requirements should be evaluated and prioritized, and then the overall scope of the proposal should be reduced or phased in such a way to more likely fit the required budget and schedule. The powerful combination of function points (linked directly to requirements) and a parametric estimation tool make these *what if* scenarios possible.

Step 5 – Evaluate Proposals in Context of Estimates

This is the *payoff* step. Steps 1 through 4 prepare a program office for proposal evaluation. The cost and schedule estimates represented by the S curve charts

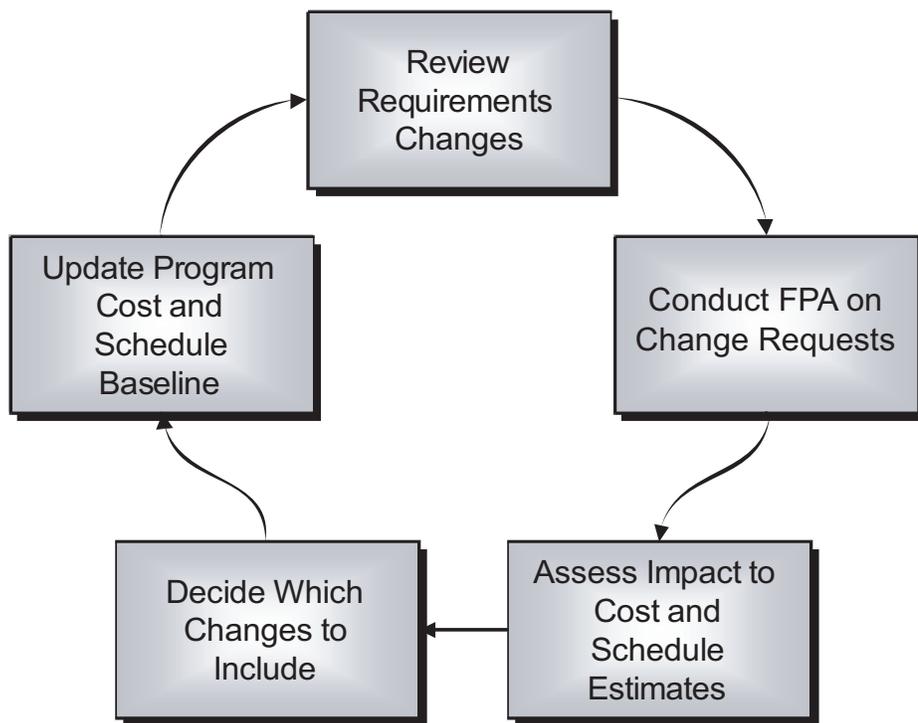Figure 5: *Schedule Evaluation Framework*
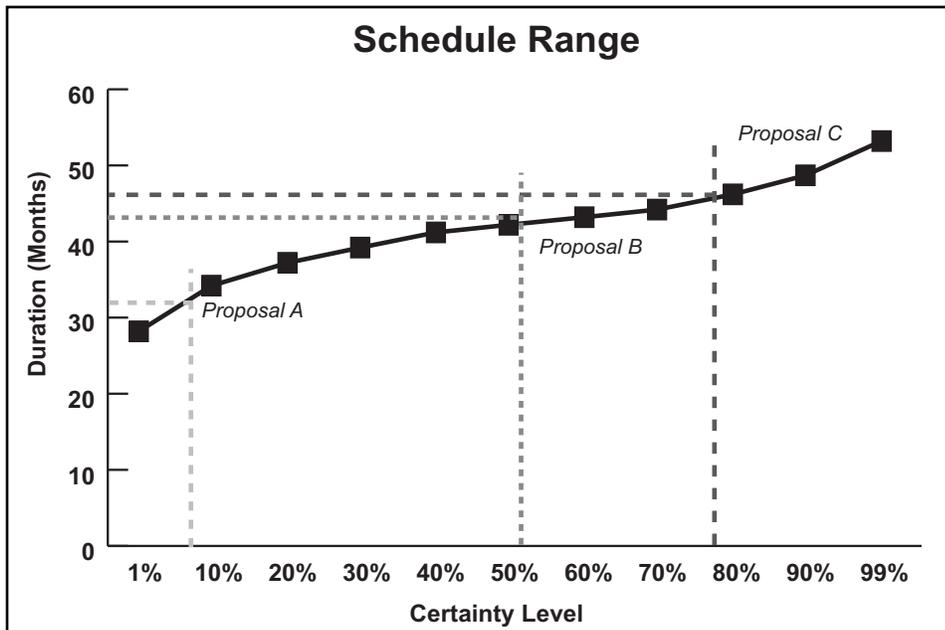
Figure 6: *Iterative Cost Estimation Process*

provide the framework against which actual proposals can be compared. When vendor responses are received, the cost and schedule proposed by each vendor can be mapped to a point on the independent cost and schedule curves (see Figures 4 and 5, page 11). Comparing certainty levels across the estimate continuum provides a more informed understanding of the relative value or risk of any given proposal.

Any given certainty level is interpreted as the likelihood that the project can be completed at or below that cost or schedule. The higher the certainty level, the more likely the project can be completed within that estimated cost and schedule. This comparative analysis can provide a

tremendous amount of information to a program office. It can help identify *price to win* proposals that are overly optimistic as well as more conservative proposals that might be overpriced. This framework allows the program to control the amount of cost and schedule risk it accepts when awarding the work by providing a context to the winning bid that is based on robust quantitative analysis.

Other Critical Considerations

When evaluating the submitted proposals within this framework, there are several items that absolutely must be kept in mind. These critical considerations can significantly impact the value of the analysis and

the quality of the information that results.

*   **Width of Cost and Schedule Ranges.** These should increase with more uncertainty in an acquisition. How much is *really* known about the project should be carefully evaluated at any given point.
*   **Contract Type.** The nature of the contract type may influence the proposals received in response to the RFP. Time and materials or cost plus award fee contracts are more likely to be priced aggressively (lower certainty levels) than fixed price proposals.
*   **Cost Versus Effort.** Parametric tools actually calculate effort and then multiply effort by labor rates to arrive at cost estimates. Evaluating proposals based on effort estimates normalizes for differences in labor rates and highlights which vendor is actually proposing the most efficient solution. Note, however, that highly skilled and experienced vendors will likely have higher hourly rates.
*   **COTS Usage/Software Reuse.** Some proposals may have assumed more software reuse or COTS applications than others. In this situation, the program office may want to run multiple scenarios with varying levels of COTS components or other software reuse assumptions. This produces multiple evaluation frameworks but allows for more appropriate apples-to-apples comparisons.

Post-Contract Award

The main purpose of this methodology is to enable more informed decisions when evaluating and awarding the initial development contract. The benefits, however, do not have to end there. The same estimation methodology can be applied in an iterative fashion throughout the entire contract life cycle to manage the project with quantitative data. The baseline FPA can serve as the basis of the initial contract, establishing threshold delivery rates or other relevant performance metrics. The contract could also implement a progressive fee structure for functionality added or changed in later development phases.

As part of a change control process, recurring FPA, coupled with updates to the estimation model, can evaluate the potential impact of proposed changes to functional or technical requirements on the project cost and schedule. The estimation process at this point should become cyclical in nature, reviewing change request, and revising size, cost, and schedule estimates based on the methodology

Figure 7: *Financial Management System Consolidation Cost Framework*
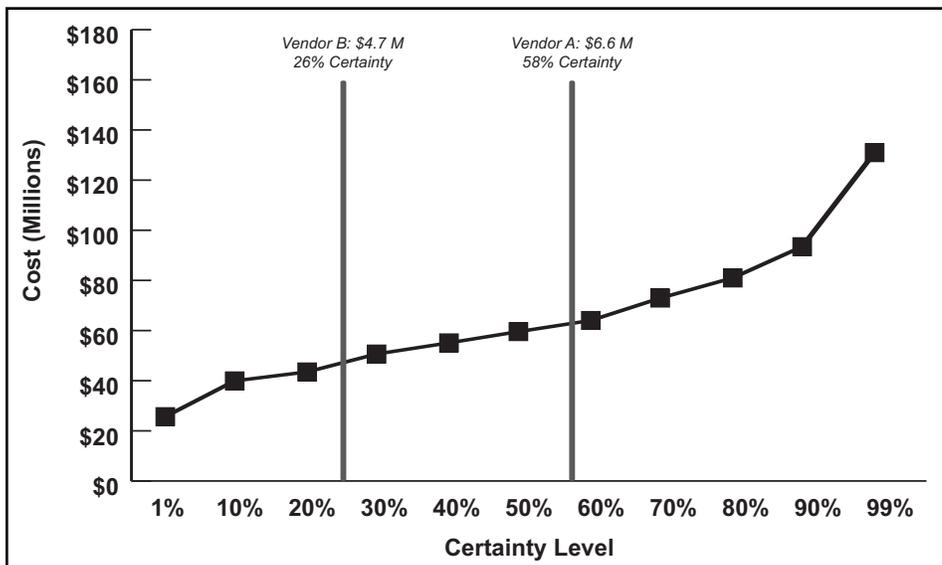
(see Figure 6). *Go/no go* decisions regarding these changes can be based on quantitative analysis instead of guesswork. This approach is one way to help keep requirements volatility and *scope creep* under control. Finally, function points and a robust estimation model can provide the data essential to earned value management by establishing well-documented baseline cost and schedule plans, providing the ability to update these plans when requirements change, and effectively assessing *percent complete* or the value that has been delivered at any point in time.

Example

A client organization desired to merge multiple financial management systems with overlapping functionality into a single, consolidated system. The business owners developed a master set of requirements that any solution would have to meet, then released an RFP for open bids. Two vendors responded, both of whom assumed they would be able to leverage some amount of functionality from existing proprietary systems. Vendor A bid a significantly higher price than Vendor B, but the contracting organization was unsure which would provide the best value and was wary of the risk of focusing on the low bid.

Consultants completed an FPA of the master requirements to establish the baseline size, then conducted an independent gap analysis of each vendor's existing systems. Then, following the methodology set forth in this article, the consultants set up the RFP analysis framework that enabled the client to put each bid into context.

As the client suspected, Vendor B did offer a rather aggressive price bid, driven lower by an assumption that significant legacy reuse would be possible (see Figures 7 and 8). The probability of delivery for the bid amount was lower than desired (~26 percent certainty). Vendor B's schedule estimate was actually more conservative, but the client identified the lower cost estimate and the high reuse assumption as a potential risk for the project. Vendor A provided an estimate that fit into the evaluation framework at a more conservative level (~58 percent). Vendor A's schedule estimate was more aggressive than Vendor B's, but it was still within a reasonable range in the framework. Cost risk was more critical to the client organization than schedule risk. This approach allowed the client to make an informed decision to award the contract to Vendor A based on quantitative analysis. The client could justify the higher price bid while understanding where critical risk
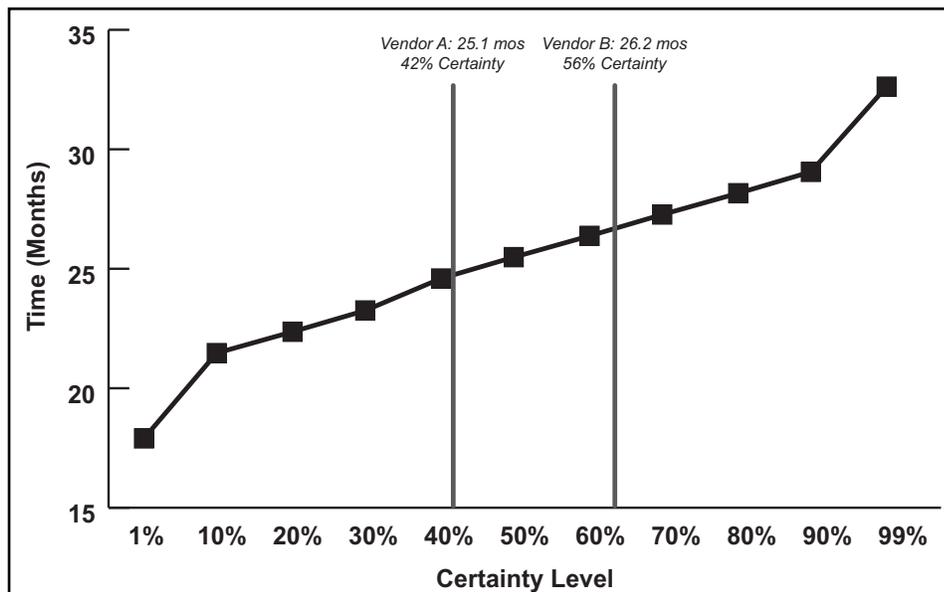

Figure 8: *Financial Management System Consolidation Schedule Framework*

areas were in the acquisition strategy.

Summary

The more information a program office has during the software acquisition process, the better the chances are for having the answers to the questions that the beginning of this article laid out. The reasonableness of the proposed price will be known, thanks to the context provided by the independent estimate. Proposals that are intentionally low-bid in order to win the contract can be identified and filtered prior to award. Conducting an FPA will help ensure completeness of requirements to improve the probability of delivery of full functionality. The methodology also allows a program office to conduct a *self-examination* to make sure that the planned budget and duration are reasonable and do not doom the project for cost and schedule overruns before it even starts. This methodology enables the contracting program office to more effectively control the balance of information and, in turn, produce acquisition results that benefit all stakeholders – program office, user community, and vendor alike.◆

References
1. Schofield, Joe. "The Statistically Unreliable Nature of Lines of Code." CROSSTALK Apr. 2005 <www.stsc.hill.af.mil/crosstalk/2005/04.html>.
2. Dekkers, Carol, and Mauricio Aguiar. "Applying Function Point Analysis to Requirements Completeness." CROSSTALK Feb. 2001 <www.stsc.hill.af.mil/crosstalk/2001/02.html>.

Additional Reading
1. Boehm, Barry W., Ellis Horowitz, Ray Madachy, and Donald Reifer. Software Cost Estimation With COCOMO II. Prentice Hall, 2000.
2. Galorath, Daniel D., and Michael W. Evans. Software Sizing, Estimation, and Risk Management. Auerbach, 2006.
3. IFPUG. IT Measurement: Practical Advice from the Experts. Addison-Wesley Professional, 2002.
4. Jones, Capers. Estimating Software Costs. McGraw-Hill, 1998.
5. McConnell, Steve. Software Estimation: Demystifying the Black Art. Microsoft Press, 2006.

About the Author

Ian Brown is a senior associate with Booz Allen Hamilton. With 10 years of experience in software measurement and analysis, he leads the firm's Quantitative Software Analysis capability. Brown is a member of the board of directors of the IFPUG, has spoken at several international software conferences, and is a Certified Function Point Specialist. Brown earned a bachelor's degree from Cornell University and a master's degree in public policy from Harvard University.

Booz Allen Hamilton
8283 Greensboro DR
Booz 2036
McLean, VA 22102
Phone: (703) 902-4971
Fax: (703) 902-3634
E-mail: brown_ian@bah.com