Progressive Acquisition and the RUP Part I: Defining the Problem and Common Terminology

by **R. Max Wideman**
Project Management Consultant
AEW Services

The November 2001 issue of The Rational Edge contains a perceptive article by Giles Pitette called "Progressive Acquisition and the RUP: Comparing and Combining Iterative Processes for Acquisition and Software Development." In this article, Pitette claims that

> Over the last twenty-five years, the software industry has devoted much effort to improving the way software developers construct information systems for their customers. In addition to considerable advances in domains such as programming languages and methods, improvements in the software engineering discipline have yielded widely accepted best practices and supporting tools. On the acquisition side, however, the advances have not been as striking.

But he then goes on to describe a new, progressive acquisition approach that is more compatible with modern, iterative software development practices. Based on my years of experience with acquisitions, and working closely with Rational's Mike Barnard, a Rational Unified Process® expert, I have formulated some basic tenets of a progressive acquisition approach that build upon Pitette's groundwork. In the next few issues of The Rational Edge, *I will share these ideas, trying to show how, at a high level, these tenets integrate with the software engineering methods and lifecycles of the RUP®. This first article lays the groundwork for the series by defining the problem space as well as basic terms and concepts that will enable common understanding of both the problem and possible solutions.*

Problems with the Traditional Acquisition Process

From a RUP perspective, the business of acquisition by contract (i.e., "contracting") introduces a whole new process unto itself, complete with roles and artifacts. This can be problematic for a typical software development organization because the traditional acquisition process is contract driven; and the specialists who direct it focus on activities, practices, and objectives that run counter to the best interests of successful software development.

What happens in a traditional acquisition process? Simply put, you figure out what you want, describe it in a Request for Proposal, solicit bids, pick a competent vendor with the lowest price and fastest delivery, enter into a standard legal contract, wait for the work to be finished, and come back when you can "pick up the keys to the front door." Unfortunately, as often as not, when you walk through that front door, the entrance hall is not what you expected, and it might lead directly to the back door!

Until recently, this was the process European countries[1] used to acquire large Defense Information Systems (DISs), and, as Pitette notes:

> ...there are many sad stories to tell about large DIS projects procured under classic, strictly sequential, "big-bang" model (also termed "grand-design" or "one-shot") -- stories about late deliveries, cost overruns, and failures to meet users' real needs. The problems stem from the big bang's inherent inability to deal with a few stark realities.

These "realities" include the following:

- **You can't express all your needs up front.** It is usually not feasible to define in detail (that is, before starting full-scale development) the operational capabilities and functional characteristics of the entire system.
- **Technology changes over time.** Acquisition lifecycles for very large systems (such as DISs) span a long period of time, during which significant technology shifts may occur.
- **Large systems are also complex systems.** This means it is difficult to cope with them adequately unless you have an approach for mastering complexity.

Actually, based on my experience in software development, the system does not have to be all that large, or the acquisition lifecycle all that long, to suffer from exactly the same difficulties. In fact, Pitette might have added:

- **The acquiring authority often fails to stay involved with the ongoing delivery of work.** Typically, this is because fixed pricing discourages them from doing so. Conversely, some purchasers breach their contracts by "interfering" too much.
- **The acquiring authority is not prepared** for the unwieldy changes that typify software projects.

- **The authority may fall short in managing and coordinating** large, parallel acquisition efforts involving multiple hardware/software suppliers.

Despite all of these shortcomings, the big-bang contract model remains very popular with executive and senior managers on the acquisition side. Why? Because they have a responsibility to maintain the financial viability of their organization. This applies whether the enterprise is government, private sector, or even non-profit. In assessing the needs of their organization, and in prioritizing opportunities (even if some system upgrades are mandated by legislation), managers must know "how much" in order to budget, estimate return on investment, and select among competing needs. And for commercial organizations, the question of "how soon" is usually more important than for those in the public sector.

Unfortunately, senior managers often have little understanding of how software development is best conducted, and (I hesitate to suggest), software developers often have little understanding of senior management's needs. So, there is potential for a serious communication gap between these two groups within the acquiring organization -- and often between managers in the acquiring organization and developers on the supply side as well.

Keeping It Simple

A modern, progressive acquisition process can help bridge these communication gaps. As I mentioned earlier, the purpose of this series of articles is to suggest basic tenets of progressive acquisition that integrate with the software engineering methods and lifecycles of the Rational Unified Process. Of course, neither suppliers nor acquirers are likely to take to these suggestions seriously at first, because they run counter to current practices. But if you keep on doing what you've been doing all along, then you'll keep getting the same kind of suboptimal results.

In laying out these tenets of progressive acquisition in future articles, we will assume an uncomplicated scenario that meets the following conditions:

- The system is "complex"; in other words, it can be developed progressively and begin delivering value early on.
- Only one supplier is involved in delivering the complete system.
- Both the acquirer and the software system supplier are using the RUP.
- The main issue is how best to set up a contract that meets the

The Vocabulary of Acquistion

One effective way to bridge communication gaps is to establish a common vocabulary among all the players. Interpreting terminology is often a problem within a business unit whose members are spread far apart, but it can become a serious obstacle when you begin to mix people with totally different business backgrounds. Not only do some terms mean different things to different people, but

needs of both parties.

Yes, I know, there will be a howl from those who must deal with multiple suppliers. That certainly adds risk and complexity, and it is frequently a source of conflict and grief. However, from a project management perspective, what that boils down to is simply assigning responsibility for coordinating, integrating, and configuring the various parts of the system in a legal and competent way.

In this first article, we'll define common terms and concepts that enable better communication for all parties involved in acquisition. Then, future articles will go on to define an effective core process for acquisition as well as likely deviations from it. My hope is that this series will help organizations to integrate their software development and acquisition processes more effectively -- and also help suppliers who market to software organizations.

What Does Contracting Involve?

As we have noted, in a traditional acquisition process, the legal terms of a contract drives the activity in a fairly rigid way. So typically, a lot of energy goes into reaching agreement on the terms of the contract. This presents another communication challenge, as contracts themselves are very flexible. They can be devised to reflect any number of variables, such as:

- The degree of definition for the thing to be delivered (i.e., the scope).
- The type of product, whether tangible as in a good, or intangible, as in a service or a product such as software.
- Safety and liability considerations for the product in

also the same meaning may be expressed in entirely different terms. Because I hope that both developers and managers -- and both acquirers and suppliers of software development resources -- will read this article, it is important to establish shared definitions for the following key terms.

Acquisition -- The process of obtaining a system, software product, or software service through contract.[2] Also known as procurement. Although not specifically stated in the ISO/IEC 12207 Standard, the term refers to purchases made through legal contract. Sometimes acquisition is applied in a narrower sense, to refer to buying an off-the-shelf, or pre-existing, system or software, with or without some degree of customization. While this is not correct usage, Figure 1 below shows "off the shelf software" at the far end of the customization spectrum as an example of zero-customization.

Acquirer -- An organization that acquires or procures a system, software, or software service from a supplier. The acquirer may also be referred to as a purchaser, buyer, customer, or owner. The acquirer may or may not also be the "user." Generally, users are a subgroup interested primarily in the software's capability and ease of use, whereas the acquirer is more concerned with cost and delivery schedule, given agreement on the functionality.

Contract -- A binding

use.

- The urgency or specific timetable for product delivery.
- The price to be paid upon delivery of the product, as well as financial incentives or penalties tied to specific benchmarks.
- The degree of control to be exercised by either party to the contract.
- The degree of tolerable risk, internal and external, and who should assume which risks.

Part III of this series, "Working with Traditional Contracting Practices," will explore these variables in greater detail.

Modern contracts for software acquisition should reflect the notion that initial plans for a software system are not monolithic and ironclad. The RUP focuses on an iterative approach and delivery of value as the most effective, efficient, and least risky means of developing software. Similarly, the PA processes developed and articulated by European governments have shown that iteratively acquiring and implementing system functionality is a much more effective approach than purchasing a complete system all at once, as in the "big-bang" model.

Software acquisition contracts should also recognize the high degree of variability in software "products." In fact, the actual software deliverable and its corresponding contract can fall anywhere along a continuum of a number of semi-interdependent variables. Figure 1 displays three major variables that I will discuss, along with several others, in a later article.

agreement between two parties, especially enforceable by law, for the supply of software service or the supply, development, production, operation, or maintenance of a software product.[3] This is a binding agreement that establishes the requirements for the products and services to be acquired.[4]

The ISO/IEC 12207 Standard definition also suggests that a contract may be "a similar agreement wholly within an organization." Generally, no form of agreement is enforceable by law unless the parties are operating "at arm's length" -- in other words, they are entirely independent of one another. However, large corporations may wish to establish internal agreements similar to legal ones as a matter of operational policy, and the extent to which they are enforceable by law depends on the relationship between the parties. Note that contract law labels the parties to a contract as *buyer* and *seller*.

Progressive Acquisition (PA) -- A strategy to acquire a large and complex system that is expected to change over its lifecycle. The objective of PA is to minimize many of the risks for both parties associated with the length and size of software projects. The final system is obtained by upgrades of the system capability through a series of evolutionary, operational increments.[5]

Subcontractor -- A second and distinct party to which a primary contractor passes some portion of work

described in the contract. This term is sometimes used incorrectly to describe the awarding of several contracts by the acquiring organization.

Supplier -- Any organization that supplies services or goods to the customer. Also known as a *contractor*, *seller*, *subcontractor*, or *vendor*.

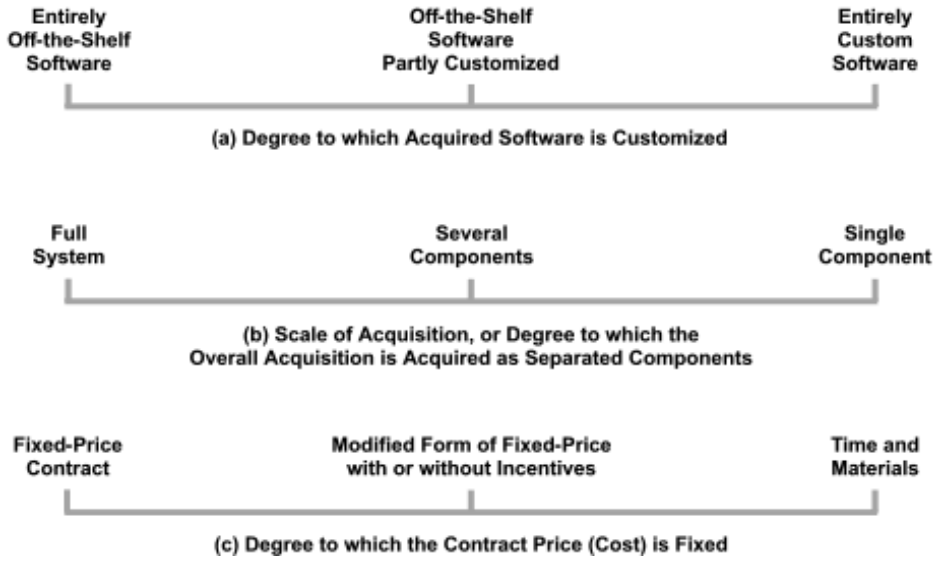

Figure 1: Major Variables Affecting Software Acquisition

Given the highly variable nature of these deliverables, determining an appropriate form of compensation is a major factor in formulating the contract -- and hence the relationship between the parties.

The Project Management Institute's PMBOK® Guide describes the following traditional contract/compensation options.[7]

- **Fixed Price (or lump-sum) contracts**. This category of contract involves a fixed total price for a well-defined product. Fixed-price contracts may also include incentives for meeting or exceeding selected project objectives, such as schedule targets.
- **Cost-reimbursable contracts.** This category of contract involves payment (reimbursement) to the contractor for its actual costs. Costs are usually classified as direct costs (costs incurred directly by the project, such as wages for members of the project team) and indirect costs (costs allocated to the project by the performing organization as a cost of doing business, such as salaries for corporate executives). Indirect costs are usually calculated as a

percentage of direct costs. Cost reimbursable contracts often include incentives for meeting or exceeding selected project objectives, such as schedule targets or total cost.

- **Time and material contracts.** Time and material contracts are a hybrid type of contractual arrangement that contains aspects of both cost-reimbursable and fixed-price-type arrangements. Time and material contracts resemble cost-type arrangements in that they are open ended, because the full value of the arrangement is not defined at the time of award. Thus, time and material contracts can grow in contract value as if they were cost-reimbursable type arrangements. Conversely, time and material arrangements can also resemble fixed-unit arrangements when, for example, the units rates are preset by the buyer and seller, as when both parties agree on the rates for the category of "senior engineers."

The supporting text in the PMBOK Guide also describes six major procurement management processes: Project Procurement Management, Procurement Planning, Solicitation Planning, Solicitation, Source Selection, Contract Administration, and Contract Closeout.

However, these classifications are too limited; they reflect a traditional procurement paradigm, not a progressive acquisition approach. The challenge for RUP users is to devise a new approach that speaks to all players in terms they can understand, while at the same time remaining consistent with the RUP philosophy and methodology.

Next month, in Part II of this series, we will take a look at an actual acquisition process and discuss how to make it work in a way that is compatible with RUP recommendations for software development.

Notes

[1] ISO/IEC 12207 International Standard, Section 3: "Definitions."

[2] "Software Acquisition Capability Maturity Model," Appendix B: Glossary of Terms. Software Engineering Institute, 1999.

[3] ISO/IEC 12207 International Standard, Section 3, "Definitions."

[4] "Software Acquisition Capability Maturity Model," *Op.Cit*.

[5] Giles Pitette, "Progressive Acquisition and the RUP: Comparing and Combining Iterative Process for Acquisition and Software Development," *The Rational Edge*, November 2001.

[6] "An Abridged Glossary of Project Management Terms" (Rev.4) in the *Association of Project Management (UK) APMP Syllabus*, second edition, 2000.

[7] *A Guide to the Project Management Body of Knowledge*, 2000 edition. Project Management Institute (USA).

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!