NPS-AM-14-C11P07R02-037

PROCEEDINGS
OF THE
ELEVENTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

WEDNESDAY SESSIONS
VOLUME I

Analyzing Quality Attributes as a Means to Improve Acquisition Strategies

Lisa Brownsword, Carnegie Mellon University
Cecilia Albert, Carnegie Mellon University
Patrick Place, Carnegie Mellon University
David Carney, Carnegie Mellon University

Published April 30, 2014

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

Panel 7. Challenges of Software Development in an Open Architecture Environment

Wednesday, May 14, 2014	
1:45 p.m. – 3:15 p.m.	**Chair: TBD** ***Achieving Better Buying Power Through Cost-Sensitive Acquisition of Open Architecture Software Systems*** Walt Scacchi, University of California–Irvine Thomas Alspaugh, University of California–Irvine ***Analyzing Quality Attributes as a Means to Improve Acquisition Strategies*** Lisa Brownsword, Carnegie Mellon University Cecilia Albert, Carnegie Mellon University Patrick Place, Carnegie Mellon University David Carney, Carnegie Mellon University ***Combining Risk Analysis and Slicing for Test Reduction in Open Architecture*** Valdis Berzins, Naval Postgraduate School

Analyzing Quality Attributes as a Means to Improve Acquisition Strategies

Patrick Place—is a senior member of the technical staff at the Software Engineering Institute. Recent projects include developing practices for engineering in a system of systems context; the Service Migration and Reuse Technique; and the acquisition implications of adopting a service-oriented architecture (SOA) development strategy. He has participated in a number of independent technical assessments related to the adoption of SOA practices. Place has been an adjunct lecturer at Carnegie Mellon University, South Bank University, and Imperial College teaching various courses on the use of formal specification techniques. [prp@sei.cmu.edu]

Lisa Brownsword—is a senior member of the technical staff at the Software Engineering Institute. She manages and participates in customer engagements for major programs within the Department of Defense and federal agencies, providing pragmatic expertise in software engineering, systems of systems, commercial off-the-shelf–based systems, architecture and product lines, and iterative development. She is managing a research team to identify patterns of misalignment of acquisition strategies, architecture, and program business and mission goals that lead to program problems. [llb@sei.cmu.edu]

Cecilia Albert—is a senior member of the technical staff at the Software Engineering Institute, where she has managed strategic software improvement across Army programs and codeveloped a process for developing commercial off-the-shelf–based systems. Albert has more than 35 years of experience developing and acquiring software-reliant systems and holds a master's degree in computer science from Stanford University and a bachelor's degree from Sweet Briar College. [cca@sei.cmu.edu]

David Carney—is a retired member of the Software Engineering Institute (SEI), now working as a consultant on select assignments. He has 30 years' experience as a software engineer working at Intermetrics, the Institute for Defense Analyses, and the SEI. His fields of particular interest have included computer-aided software engineering tool environments, system of systems integration, issues relating to the use of commercial off-the-shelf products, and service-oriented architectures. [djc@sei.cmu.edu]

Abstract

In the acquisition of a software-intensive system, the relationship between the software architecture and the acquisition strategy is typically not specifically examined. The first phase of our research discovered an initial set of failure patterns that result when these two entities become misaligned. Programs with these failure patterns experienced reduced operational capabilities and effectiveness, cost overruns, and significant schedule slips. In other words, these programs resulted in systems failing to satisfy stakeholder needs.

This paper briefly describes the conceptual foundations for our project and summarizes the first phase as context for the second phase, which is the major thrust of this paper. The current research has centered on demonstrating the existence and utility of acquisition-related quality attributes, embodied in a program's business goals, which then drive the shape of the acquisition strategy. This is comparable to the relationship between mission goals, software-related quality attributes, and the software architecture. This paper describes the approach used to generate 75 acquisition-related quality attribute scenarios based on data derived from more than 23 large government programs spanning business, logistics, command and control, and satellite domains.

Introduction

Our project is focused on the relationships between software architecture[1] and acquisition strategy.[2] Although these entities might appear to be unrelated, there is a surprisingly deep connection between them. More specifically, we are concerned with their alignment or misalignment. By identifying and articulating how key entities that are critical to alignment or misalignment interact, we can provide a useful approach for organizations and project managers engaged in acquisition programs.

The key entities of interest are as follows: the architectures themselves, both software and system; the planned acquisition strategy; the quality attributes that drive those architectures and strategies; and the goals (both business and mission[3]) of all of the stakeholders. By examining these entities, we seek to pinpoint major sources that tend either to keep the software architecture and acquisition strategy in harmony or to pull them apart. By so doing, we intend to provide a method for organizations and project managers to avoid patterns of failures that we have discovered, which are summarized in the Phase One: Characterizing Failure Patterns section. We expect to validate the utility of this method through pilot applications on projects and programs outside the Software Engineering Institute (SEI).

This project is expected to take place over multiple phases. The remainder of this section describes the conceptual foundations for our project as a whole. The second section summarizes the first phase as context for the third section, which discusses the current work we completed in the second phase and is the major thrust of this paper. In the final section, we describe our plans for future phases.

Hypotheses

Our primary hypothesis is that a mismatch between acquisition strategy and software architecture contributes to significant problems in acquisition programs. If a program can avoid the patterns of failure (such as those we identified in phase one of our research), its acquisition strategy and software architecture can be aligned and the program can increase the likelihood of program success.

This hypothesis depends on two key premises. First, a software architecture and an acquisition strategy are necessarily related, as shown in the work of Conway (1968) and MacCormack (2011). These entities form two conceptual structures that are parallel, although in different spheres of the acquisition space (i.e., the software architecture, and the

[1] *Software architecture* is defined by Bass, Clements, and Kazman (2012) as “the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.”
[2] *Acquisition strategy* is defined by the Defense Acquisition University (2011) as “A business and technical management approach designed to achieve program objectives within the resource constraints imposed. It is the framework for planning, directing, contracting for, and managing a program. It provides a master schedule for research, development, test, production, fielding, modification, postproduction management, and other activities essential for program success.”
[3] A mission goal is an expression of an operational objective that could affect a user, focused on what the solution or product should do or how it should behave. A business goal is an expression of an organizational (e.g., Navy) objective, focused on what the acquisition (development or maintenance) organization should do or how it should behave.

mission users and goals, on one hand; the acquisition strategy, and business stakeholders and goals, on the other hand).

Second, the quality of the relationship between software architecture (and related mission goals and software and system quality attributes) and program acquisition strategy (and related business goals and acquisition quality attributes[4]) is of critical importance to the success of the program[5]. This relationship must be one wherein these two entities are both aligned and mutually constraining.

Foundations

This research builds on significant previous work. There are three foundations that frame our research. The first is our recognition and appreciation of the considerable body of knowledge that has emerged from the SEI's ongoing work in software architecture[6]. Two SEI methods are of particular relevance to our work:

- generating, documenting, and prioritizing a system's quality properties (e.g., performance, availability, interoperability): the quality attribute workshop (QAW; Barbacci et al., 2003) and
- eliciting and documenting high-priority business and mission goals and capturing the architectural implications of those goals: the Pedigreed Attribute eLicitation Method (PALM; Clements & Bass, 2010).

The second foundation of our research lies in the ongoing efforts of the Department of Defense (DoD) to improve the acquisition process. These efforts have had two positive effects on our research. First, the business goals of the DoD have been clearly stated; second, they make the relationship between these business goals and the program's acquisition strategy more explicit. With efforts such as Better Buying Power 2.0, described by the Office of the Under Secretary of Defense (2012), improvements are being sought in delivering better value to both the taxpayer and the warfighter.

A third foundation of our work is the SEI's experience with more than 100 independent technical assessments (or ITAs, and often informally called "red teams"). Such assessments are commissioned by the government to provide third-party analyses of a program's health, quality of progress, and similar conditions. Our team's ITA experiences and those of our colleagues strongly corroborate the observations noted in the previous paragraph.

Phase One: Characterizing Failure Patterns

As previously noted, developing a validated method that facilitates the alignment of a software architecture and the acquisition strategy within a program is a multiphase project. Our objective for phase one was to discover the potential causes of mismatch between the acquisition strategy and the software architecture that contribute to acquisition program

[4] Acquisition quality attributes are properties of the program—analogous to software or system quality attributes.
[5] Within the information systems arena, the relationship between business goals and information technology (IT) is termed *business-IT alignment* and is considered crucial to the success of an enterprise. For more information, readers can refer to Strassman (1998) or Henderson and Venkatraman (1993).
[6] Although we specifically call out SEI research in software architecture, we are not limited to this source in our research. We are leveraging other work in the broader architecture and requirements community, particularly as we move into phase three of this project.

problems. In this section, we summarize the activities and outcomes of the first phase of our project, principally covering the following: patterns of failure, or anti-patterns; entities and relations that pertain to the anti-patterns; and conclusions we drew from this phase of our research.

Entities and Relations That Pertain to Our Anti-Patterns

Our initial focus was on gathering data by conducting interviews with SEI personnel who had participated in major ITAs. In analyzing this data, we discovered several recurring patterns of mismatches between the acquisition strategy and the software architecture leading to programmatic failures. We based some of this analysis on an existing body of research on design patterns:

> [A pattern] describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice; in other words, a pattern is a template that can be used in a specific situations. (Alexander, Ishikawa, & Silverstein, 1977)

We transposed this description somewhat, since Alexander et al.'s description of a pattern included the presence of a solution to a problem, while we were describing only the problem element. This transposition is commonly called an "anti-pattern" within the software community, as exemplified by Brown, Malveau, McCormick, and Mowbray (1998).

While there are many patterns of failure in acquisitions, the analysis of our data identified a number of anti-patterns that were evident in the programs we studied. These were the following:

1. undocumented business goals,
2. unresolved conflicting goals,
3. failure to adapt,
4. turbulent acquisition environment,
5. poor consideration of software,
6. inappropriate acquisition strategies, and
7. overlooking quality attributes.

For each of these anti-patterns, we described the context in which the problem usually emerged, the specific nature of the problem, the observed response to the problem (NB: not a solution, but rather the observed response that failed to solve the problem), and examples of the consequences, both immediate and long-range. Brownsword et al. (2013) described these anti-patterns further.

Entities and Relations That Pertain to Anti-Patterns

Based on our analysis, we conjectured that there was a small number of critical entities involved in these anti-patterns and that they were related in significant ways. The entities are as follows:

- mission goals, and the (system and software) quality attributes implicit in those goals;
- business goals, and the (acquisition) quality attributes implicit in those goals;
- the acquisition strategy;
- the software and system architectures, which are closely related but separate; and

- the different sets of stakeholders who have expressed needs that are captured by the mission and business goals.

The set of entities and relations is shown in Figure 1 and is at the heart of our primary hypotheses: if these relationships between the main entities of an acquisition are strong, then there is a higher chance that the acquisition strategy and the software architecture are mutually constraining, and at least this cause of acquisition failure can be avoided. For example, by strengthening the relationship "stakeholders have business goals," such that these goals from the salient stakeholders are collected and exist in a coherent artifact, then Anti-pattern 1 (undocumented business goals) would not occur, or be substantially reduced. Brownsword et al. (2013) discussed further how the anti-patterns noted previously are affected by these relationships.

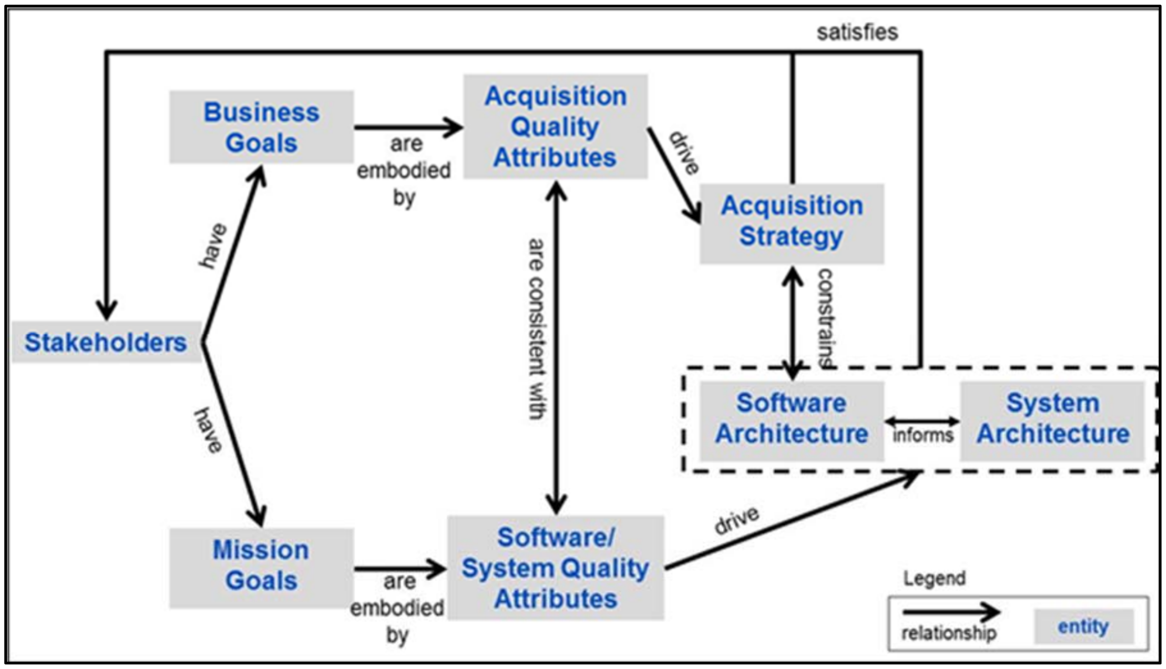

Figure 1. Desired Relationships Among the Principal Entities

Conclusions From Our Phase One Research

As shown in Figure 1, the business goals for a program are a key entity. Although our data showed that a number of important stakeholders have business goals, these goals are often not expressed or captured. Further, we observed that there was no process for doing so. Without such a process, it is difficult to analyze these goals for conflicts with other mission or business goals, let alone to analyze for the sufficiency of the acquisition strategy to accommodate the desired business goals.

Through our phase one research and analysis, we concluded that the business goals, similar to mission goals, will have quality attributes that should be the main drivers for the acquisition strategy. We assert that these acquisition strategy–related quality attributes are as important as those derived from the mission goals and refer to them as *acquisition quality attributes*. We posit that these acquisition quality attributes are a critical means for forming and analyzing the acquisition strategy for a particular program.

How are these acquisition quality attributes best elicited and captured? Can they be used to surface potential conflicts among other business goals? Can they show possible impacts on an acquisition strategy? Exploring these questions became the basis for phase two of our project.

Phase Two: Exploring Acquisition Quality Attributes

The focus for phase two of our project was to demonstrate the applicability of acquisition quality attributes. Our premise was twofold: (1) there is a set of program-specific acquisition quality attributes that can be derived from a program's business goals that drive its acquisition strategy and (2) acquisition quality attributes can be expressed in a way that allows them to be analyzed and evaluated. Our research methodology for this phase consisted of the following activities:

- form a list of potential acquisition quality attributes,
- define an approach for expressing program-specific acquisition quality attributes that allows them to be effectively reasoned about,
- elicit and capture acquisition quality attribute scenarios,
- build and validate a prototype workshop to elicit acquisition quality attribute scenarios, and
- analyze the acquisition quality attribute scenarios.

Much of this work followed a similar path as that used with the original research on software quality attributes. In particular, similar to the developers of QAW and PALM, we adopted the principle of using scenarios to give precise meaning to acquisition quality attributes.

Potential Acquisition Quality Attributes

There are many different ways that attributes—whether the software quality attributes of software architecture or the acquisition quality attributes we are presently focusing on—can be aggregated. We decided initially to simply create a list, unordered and without concern for generality or specificity, and use the scenarios to give us insight as to what a reasonable taxonomy might be. The initial list was derived from a combination of reviews of DoD acquisition strategy guidance and discussion with acquisition professionals, colleagues, and several brainstorming sessions within our team.

However, as we reflected on the collection of acquisition quality attribute scenarios generated through our research, we saw emerging themes that may provide the basis for a possible taxonomy of acquisition quality attributes in the future. We observed the following:

- *Executability* tends to occur when program cost, schedule, and performance are in balance and can therefore be further decomposed into *affordability, schedulability,* and *performability*.
- *Flexibility* tends to occur when a program can respond appropriately to changes and could therefore also embrace the attribute of *innovativeness*.
- *Program survivability* tends to occur to the extent that a program can defend against external pressure; this attribute could also embrace the attribute of *staffability*.
- *Realism* tends to occur when stakeholder expectations are compatible with the program.
- *Transparency* tends to occur when information about cost, schedule, and performance is available.

These initial observations are oriented exclusively to acquisition and programmatic factors and do not attempt to account for software architecture decisions. Further work during phase three of this project may refine and extend these early observations.

Expressing Program-Specific Acquisition Quality Attribute Scenarios

We next considered how program-specific scenarios might be constructed. Once again, the example from the work in software architecture–based scenarios was invaluable. In a software architecture QAW, end users are encouraged to create small "stories" that specify some event (the "stimulus") that occurs under particular conditions (the "environment") and then the desired behavior (the "response") of the system.

An example of such a scenario from the domain of software architecture (Software Engineering Institute, n.d.) might be the following:

Stimulus:	An internal component fails
Environment:	During normal operation
Response:	The system is able to recognize a failure of an internal component and has strategies to compensate for the fault

A parallel example from the domain of acquisition might be the following:

Stimulus:	An unexpected budget cut
Environment:	For a multi-segment system
Response:	The program is able to move work between major segments to speed up or slow down separate segments within the available funding

Subsequently, a program would expand these three-part scenarios to six parts: the original three parts; who generates the stimulus (the "source"); the artifact that the stimulus most strongly affects, and the measure(s) by which the success of the response will be evaluated. In practice, this expansion and refinement takes considerable effort. We investigated this refinement and expansion for many of the acquisition quality attribute scenarios created in this phase, and we expect to continue our investigations in the following phase of our work. For simplicity of presentation in this paper, we use the three-part scenario form.

Elicit and Capture Acquisition Quality Attribute Scenarios

A major component of phase two was the task of collecting and describing a large number of scenarios that would provide us with the necessary basis for analyzing alignment and incompatibilities. To accomplish this task, we needed additional data in the form of actual acquisition situations, events that occurred (whether beneficial or otherwise), the types of conditions in which the programs unfolded, the kinds of authority structures and strictures that were present, and related kinds of information. To this end, we gathered a large body of actual acquisition experiences through interviews from a variety of acquisition professionals, and then we refined that experience into a collection of acquisition quality attribute scenarios. The aggregate data covered 23 government programs gathered through interviews with former program management office personnel and ITA members.

The data we collected from the interviews described actual acquisition experiences, each one concerning events with significant effect on the success of a given program. For each of the descriptions, we isolated the event that occurred: this formed the *stimulus* of the scenario. The kinds of stimuli that we noted included the discovery that a contractor is nonperforming due to a lack of capability on staff, and the need to react quickly, and there are only a limited number of contractors able do the work

We then noted the conditions that were present when that stimulus occurred. By "conditions that were present," we refer to a variety of things that might provide the *environment* for the scenario. Examples included a program where the work is classified and it takes a long time to get people cleared and where warfighters have urgent operational needs and there is a limited number of contractors able do the work.

And finally, we considered the behavior, i.e., the *response* to the event. At this point, our focus became divided. On one hand, some of our data (generally drawn from ITA experiences) indicated what a program had actually done, which was, in retrospect, failing behavior. By contrast, we also examined data where a program had planned well in its earliest days, and when some unforeseen event occurred, the program responded in a beneficial manner.

Comparing these different programs was at the heart of our work during this phase. We realized that developing acquisition quality attribute scenarios for a real program would have acquisition-focused program participants try to forecast unexpected events, and the persons defining the acquisition strategy must design the strategy to permit an appropriate and beneficial response. We therefore cast each scenario in "beneficial" terms.

The following are examples of the scenarios we constructed. We added an element to show possible acquisition strategy tactics—that is, how the scenario response could be incorporated into an acquisition strategy.

Acquisition Quality Attribute Scenario A:

Stimulus	One associate contractor refuses to share information with other contractors
Environment	Associate contractors are competing on other customer work
Response	Use management structures and incentives to force collaboration
Potential Acquisition Tactic	Create contract requirements so government can monitor collaboration

Acquisition Quality Attribute Scenario B:

Stimulus	A new need arises when we want to react quickly
Environment	There are only a limited number of contractors able to do the work
Response	Work to satisfy the need is added to an existing contract
Potential Acquisition Tactic	Award indefinite delivery/indefinite quantity contracts to multiple (perhaps eight or so) vendors, and issue task orders in a round-robin fashion

Build Prototype Workshop to Elicit Acquisition Quality Attribute Scenarios

The investigations in scenario elicitation through interviews were focused largely on how to form viable acquisition quality attribute scenarios from the interview data. This was a necessary step toward developing a technique that we could use with a program that was in the process of forming its acquisition strategy. We again leveraged the work in the software architecture community in eliciting software quality attributes—namely, the QAW. Where necessary, we made some modifications, but in essence, the prototype acquisition quality attribute workshop (AQAW) paralleled the QAW closely. The shape of a QAW is as follows:

- Opening presentations define the QAW process, describe the program's business and mission drivers, and outline the plan for the system architecture.
- Scenario brainstorming takes place in a round-robin fashion where each workshop participant is, in turn, asked to provide a scenario or pass for the round. Scenarios are provided in a three-part format of stimulus, environment, and response.
- The last steps of a QAW relate to analysis of the generated scenarios. The scenarios are consolidated so that duplicates are removed and the remaining scenarios are prioritized.

We adapted the QAW to form an AQAW primarily by placing more emphasis on the business presentation and replacing the architecture presentation with one on the program's acquisition strategy plans. Prior to scenario brainstorming, we modified the identification of architectural drivers step to focus on acquisition strategy drivers instead of the architecture.

We conducted a prototype of the AQAW as a test to determine whether our QAW variant could, indeed, elicit acquisition quality attribute scenarios. The prototype was conducted on a real program using SEI staff who supported the program in place of members of the program office. We asked the SEI team members to role-play the actual stakeholders associated with the program, identifying which role they were playing.

The prototype AQAW generated 20 acquisition quality attribute scenarios. While only a single instance, the prototype has successfully demonstrated that an AQAW is a plausible approach for capturing acquisition quality attribute scenarios.

Analyze Acquisition Quality Attribute Scenarios

Our interviews generated 55 acquisition quality attribute scenarios in addition to 20 acquisition quality attribute scenarios captured in the prototype AQAW. Using this data, we looked for general themes or trends.

The scenarios generated from interviews were developed by asking our interviewees to identify memorable negative and positive events that occurred in the programs they were associated with (i.e., we were identifying possible scenarios *after* the fact). Thus, they gave us scenarios that largely represented dominant problems encountered in their programs. Given the large number of programs represented, there were repeating, or at least similar, program events. Similar scenarios were grouped together, forming five categories as shown in Table 1.

Table 1. Distribution of Scenarios Derived From Interviews

Classification	Number
Contractor Capability	10
Program Office Capability	16
Sharing Across Programs	12
Innovative Solution/Technology	8
Other Software Life Cycle	9

The two most common themes occurring in the scenarios relate to personnel and requirements as part of either the stimulus or the environment: 11 (20%) of the scenarios reference lack of skilled personnel in either the program office or the contractor and seven (13%) of the scenarios reference the reality of changing or urgent requirements.

As part of capturing each scenario, we associated it with the acquisition quality attribute it defines. Table 2 shows the frequency of the acquisition quality attributes.

Table 2. Frequency Count of Acquisition Quality Attribute Scenarios

Acquisition Quality Attribute	Frequency
Flexibility	23
Performability	15
Realism	14
Affordability	10
Survivability	6
Executability	5
Responsiveness	4
Programmatic Transparency	2
Innovativeness	1
Schedulability	1

Analysis of Scenario Content

Just as for software quality attributes, the general form of an acquisition quality attribute scenario can be expressed as “if this event occurs (stimulus) when we are in this state (environment), then we want to be able to do this (response).” However, if we examine the acquisition quality attribute scenarios we have collected and focus on the stimulus, we see that the majority of these stimuli follow a slightly different form. Specifically, the stimulus itself is in two parts, where the first part reveals an issue with one of the three major programmatic controls (cost, schedule, and performance) and the second part defines the reason for that issue.

For example, one scenario from our data has the stimulus “The schedule is not being met because of poor planning by a subcontractor.” We can see that the former part is that the schedule is not being met, and then a reason is given as the latter part. It is logical that most scenarios will have a stimulus of this form, since cost, schedule, and performance are key indicators of the acquisition’s progress and are the areas on which a program reports. However, if cost, schedule, and performance were the only pieces of the stimulus, it would be impossible to fashion a detailed response. Thus, a reason for the perturbation is also a necessary part of the stimulus. The response can then be crafted to mitigate the reason; in the case of this scenario, one of the responses was “The prime contractor trains the subcontractor in project management.”

Just as occurs in a QAW, different acquisition quality attribute scenarios from a single program can be variants of each other. In the case of software quality attributes, these variants are frequently based on the same stimulus in different environments that could lead to a different response. We have found in the case of acquisition quality attributes that these variants are more likely to be based on different responses to the same stimulus and the same environment.

In hindsight, the different responses in acquisition quality attribute scenarios are a reflection of the nature of acquisition. Acquisition is about people, not software, making decisions; and, frequently, these decisions are strongly influenced by factors outside the control of the program. System and software responses are more deterministic.

The external environmental influences on acquisition decision are very difficult—maybe even impossible—to explicitly delineate. The number of factors that could influence the decisions that a program manager makes are numerous and can range from obvious factors, such as the effect of an unforeseen budget cut or new direction on the schedule driven by operational crisis, to hidden or subtle factors, such as the relationship between the program manager and the customer organization.

Different Scenarios Result in Different Acquisition Strategies

If acquisition quality attribute scenarios are truly analogous to software quality attribute scenarios, then we should be able to anticipate the influence from the acquisition quality attribute scenarios on the "goodness" of the acquisition strategy analogous to the way software quality attribute scenarios influence the "goodness" of the software architecture. Even if applied after the acquisition strategy has been developed, we should be able to use the acquisition quality attribute scenarios to distinguish between acquisition strategies or to determine the appropriateness[7] of the acquisition strategy with respect to any given scenario. Since these two ways to use an acquisition quality attribute scenario are simply a matter of timing, we focus on the first use with the knowledge that if we can demonstrate that a scenario might influence the acquisition strategy, then we can also use a scenario to test a strategy.

For an acquisition quality attribute scenario to have an influence on the acquisition, there must be some element of the scenario that leads the program office to make some kind of choice between one strategy and another. Examining the relationship, we see that the acquisition strategy should be such that the program office can make the response specified in the acquisition quality attribute scenario. Thus, if there are to be different scenarios, it is reasonable to see that we have either two different scenarios with different responses or a single scenario that leads to two different responses. In either case, we show that different scenarios lead to different acquisition strategies.

Examining the scenarios we collected, we found a number relating to new technology and the issues that arise if the chosen innovative technology fails to deliver on its promises. From the collected scenarios, we may posit a single scenario with two variant responses, where the variation depends on the environment component (indicated by italics) of the scenario:

1. A new technology that the program office expects to use is found to be unsuitable *where schedule is of prime importance*; the program office switches to an alternative that is also currently under development and is evaluated to be suitable.
2. A new technology that the program office expects to use is found to be unsuitable *where costs must be kept as low as possible*; the program office instructs the contractor to restart but using an alternative technology.

[7] We avoid the judgmental terms *good* and *bad* since most strategies will be "good" with respect to some scenarios and "bad" for others. A "good" strategy is one that is appropriate for the crucial scenarios.

We can see from these two scenarios that the stimulus is the same but the environment changes; in the first case, schedule is more important than cost, and the second case reverses their relative importance. In the first case, an acquisition strategy starting multiple developments simultaneously with a requirement for some kind of decision between the alternatives would be appropriate. In the second case, a strategy starting a single development and continuing with that until such time as it was found to be infeasible and then switching to an alternative would be appropriate.

As simple as this example is, it demonstrates that different acquisition quality attribute scenarios can lead to different acquisition strategies. This strengthens our contention that our use of acquisition quality attributes and acquisition quality attribute scenarios is, indeed, analogous to the use of software quality attributes and software quality attribute scenarios and that we may continue to rely on methods and mechanisms developed for that purpose to assist with the creation of sound acquisition strategies.

Identifying Incompatibilities Between Scenarios

In software, we frequently find that two or more software quality attributes are incompatible with each other (e.g., performance attributes are often in conflict with security attributes) and thus become the subject of architectural trade-offs. We, therefore, examined the acquisition quality attribute scenarios to determine the possible kinds of incompatibilities between different scenarios. We first considered incompatibilities that could occur between different acquisition-related scenarios and then considered incompatibilities that could occur between an acquisition-related scenario and an architecture-related scenario.

Incompatible Acquisition Scenarios

Conflicts between scenarios are not always obvious and may not become apparent immediately. In the following example, for instance, the conflict is quite subtle without some analysis. Organization ABC has deployed a large, complex legacy system in multiple operational locations, where each location installed its own local variant of the system. Over time, these variants diverged in response to differing requirements of the local users. The various operational locations identified a need to share data in a more integrated way. A new program was initiated to acquire one replacement capability that would support all of the differing needs across the multiple fielded locations. The program decided to implement an incremental approach to replacing the legacy system so they could respond to budgetary constraints and uncertainties.

The operational processes and need vary between the current fielded locations. Understandably, the user requirements for the new capability also differ across the various operational sites. As the program attempts to define an agreed-upon set of requirements, the user representatives change their requirements. In addition, an influential stakeholder has advocated the use of a new commercial off-the-shelf (COTS) product as the solution approach.

As might be expected, there are incompatible scenarios that the new program would need to surface and explicitly address if it is to meet its various stakeholders' expectations. The first scenario reflects the expectation of one influential stakeholder who advocated the use of a COTS product that had been successfully used at one of the operational installations:

Stimulus	There is a desire to replace a complex component of a large legacy system with a COTS package
Environment	Within an established enterprise architecture with many local variations implemented that are largely different from each other
Response	The program runs a contest with a big prize to evaluate COTS packages for an enterprise-wide solution.

The second set of stakeholders, reflecting the operational users, is counting on the new system to quickly address their current needs. Understandably, these needs vary among the current fielded locations. During the time it takes the program to define an agreed-upon set of requirements for each increment, the user representatives from the various fielded locations change their requirements. This leads to the second acquisition scenario for this program:

Stimulus	Requirements for the next release keep changing
Environment	For a program with a fixed budget that must be carefully managed
Response	The program accepts the new requirements

Both of these scenarios are related to an acquisition quality attribute of flexibility. They describe how the program would accommodate different stakeholder needs. Unfortunately, the two scenarios are potentially incompatible with respect to designing the acquisition strategy. The first scenario is centered around the implementation of a common COTS product across all locations. This could provide sizeable value in terms of moving to one capability that is distributed across all fielded locations, but it may not meet what the current users consider urgent needs.

Implied in these two scenarios is a third set of stakeholders, the enterprise system engineers, who are advocating the implementation of an enterprise architecture that extends across all of the local fielded implementations. This enterprise architecture could be incompatible with both of the preceding scenarios: Each COTS product, by definition, is built to an architecture and a set of requirements that ABC has no control over. Further, the demands for local fielded implementations compete with architectural changes within a constrained budget.

Competition Between Acquisition and Architecture Scenarios

A different kind of incompatibility, and one less likely to be recognized, can occur between an acquisition-focused scenario and an architecture-focused one. One cause of this is that different communities (i.e., acquisition personnel and software personnel) and different sets of goals (i.e., business and mission) are involved in creating the scenarios.

In one program, for instance, organization XYZ had been under significant criticism for delay in responding to users in the field. A new director had been appointed with a mandate to remove bottlenecks and reduce the time between program start and initial operational capability.

The acquisition strategy therefore emphasized agility, responsiveness, and other such attributes. Among the elements of the strategy were several goals that (had the AQAW been available) could have led to appropriate acquisition quality attribute scenarios. For example, the goal of responsiveness led to a strategy of maximizing the use of open-source software. If we were to couch that goal in terms of an acquisition quality attribute scenario, it might take the following form:

Stimulus	Users request significant new functionality to be delivered rapidly
Environment	During the program's development phase
Response	Create the functionality rapidly by reusing open-source software from other projects to provide much of the capability

At the same time, however, the software architects had been warned that the situation in which the system was to be used made it necessary that the system was to be safety-critical and hard, real-time. Stringent certification standards would also apply to the system. For that reason, certification of the system would depend on removal of unreachable code from any reused or open-source software. During a subsequent QAW, therefore, one of the key scenarios was aimed at a system/software quality attribute of certifiability:

Stimulus	A new requirement to adhere to a rigorous safety standard is applied to the system
Environment	During the program's development phase
Response	Remove all unreachable code to insure that the system will pass stringent new certification standards

As in the previous example, both of these scenarios were well-intentioned, but they ultimately collided. Because, as the program unfolded, the open source that was most appropriate for the system had a considerable amount of unreachable code, the development underwent very large delays since the unreachable code was extensive and was pervasive in all of the reused modules. The result was that the system was fielded almost three years late, since certification could not be done until the developers were convinced that all of the dead code was removed. By common agreement, the program office believed that while the open-source software provided benefits, they were not as significant as expected.

In analyzing the 55 acquisition quality attribute scenarios generated from our interviews, we identified 24 scenarios as having a probable impact on the software architecture. We would, therefore, expect to have one or more software quality attribute scenarios for each of the acquisition quality attribute scenarios that would need to be elicited, captured, and analyzed for potential incompatibilities. This will be an area of emphasis in the future phases of our project.

Summary and Proposed Future Steps

We are on a journey to provide a better approach to identify, understand, and reason about key drivers of a program's acquisition strategy and software architecture. Our research shows either that these drivers—in the form of business and mission goals—are implicit or that conflicts exist among the goals that are not resolved. Such misalignment can lead to reduced operational capabilities and effectiveness, cost overruns, and severe schedule slips, eventually resulting in systems failing to satisfy stakeholder needs or, still worse, leading to program cancellations.

In making this pervasive problem tractable, we first created a model of the desired relationships among key entities—stakeholders, business goals, mission goals, acquisition strategy, software and system architecture, acquisition quality attributes, and software/system quality attributes. Forming the model then allowed us to (1) determine that there is no process for eliciting, capturing, and adjudicating the business goals of a program's stakeholders comparable to a process such as the DoD's Joint Capabilities

Integration Development System process and (2) guide our research to prove our hypothesis on the existence and utility of acquisition quality attributes, embodied in the business goals, that drive the shape of the acquisition strategy—comparable to the relationship between mission goals, quality attributes, and the software architecture.

Our research in phase two was on item 2 mentioned previously (acquisition quality attributes) and is the focus of this report. We patterned our approach from that used by the SEI as it identified and codified key concepts and techniques around the relationship between software/system quality attributes and the software architecture. For the acquisition domain, we adapted the role of scenarios to create and demonstrate a viable way to express acquisition quality attributes specific to a particular program. We modified the original scenario elements in some ways: the acquisition strategy for the software architecture, the program for the system, and the program manager for the architect.

Underlying our work is the assertion that eliciting quality attributes so they can be analyzed is as critical for a sound acquisition strategy as it is for software/system architectures. We conducted various investigations in this regard, gaining experience within the acquisition domain and capturing numerous example scenarios. These investigations gave us the confidence that modifying the SEI QAW could be a viable starting point to elicit, capture, and analyze acquisition quality attributes and begin identifying potential impacts on an acquisition strategy. Our use of the prototype AQAW also indicated that it is important to explore a more deterministic approach for eliciting acquisition quality attribute scenarios that cover the breadth of acquisition strategy drivers and is potentially less dependent on the particular participants attending a workshop

Our research to date has given us strong confirmation that our initial suppositions were sound and that the method we will now develop will make a strong contribution to the acquisition community. In phase one, we saw ample evidence that, among the many pitfalls that plague acquisition programs, the lack of alignment between acquisition strategy and architectures ranked high on the scale of problems. During phase two, the gradual maturing of our concept of the acquisition quality attribute and the value of acquisition-related scenarios has taught us many considerable lessons in the complex and subtle ways that acquisition strategy and architecture have mutual influence.

References

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A pattern language: Towns, buildings, construction*. Oxford, England: Oxford University Press.

Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., & Wood, W. (2003). *Quality attribute workshops (QAWs)* (3rd ed.) (CMU/SEI-2003-TR-016). Retrieved from Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6687

Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice* (3rd ed.). Addison-Wesley.

Brown, W. J., Malveau, R. C., McCormick, H. W., & Mowbray, T. (1998). *Antipatterns: Refactoring software, architecture, and projects in crisis.* Hoboken, NJ: John Wiley & Sons.

Brownsword, L., Albert, C., Carney, D., Place, P., Hammons, C., & Hudak, J. (2013). *Isolating patterns of failure in Department of Defense acquisition* (CMU/SEI-2013-TN-014). Retrieved from Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=53252

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-oriented software architecture volume 1: A system of patterns*. Hoboken, NJ: John Wiley & Sons.

Clements, P., & Bass, L. (2010). *Relating business goals to architecturally significant requirements for software systems* (CMU/SEI-2010-TN-018). Retrieved from Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9347

Conway, M. E. (1968). How do committees invent? *Datamation*, 14(5), 28–31.

Defense Acquisition University. (2011). Glossary of defense acquisition acronyms and terms (14th ed.). Retrieved from https://dap.dau.mil/glossary/Pages/Default.aspx

Henderson, J. C., & Venkatraman, N. (1993). Strategic alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal,* 32, 1.

Office of the Under Secretary of Defense (OUSD). (2012, November 13). Better Buying Power 2.0: Continuing the pursuit for greater efficiency and productivity in defense spending [Memorandum].

Software Engineering Institute. (n.d.). Reasoning about software quality attributes. Retrieved from http://www.sei.cmu.edu/architecture/start/reasoning.cfm

Acknowledgments

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

No warranty. This Carnegie Mellon University and Software Engineering Institute material is furnished on an “as-is” basis. Carnegie Mellon University makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Carnegie Mellon University does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

This material has been approved for public release and unlimited distribution.

DM-0001076