Software Engineering Institute

A Method for Aligning Acquisition Strategies and Software Architectures

Lisa Brownsword
Cecilia Albert
David Carney
Patrick Place

October 2014

TECHNICAL NOTE
CMU/SEI-2014-TN-019

Client Technical Solutions Directorate

http://www.sei.cmu.edu

Carnegie Mellon University

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.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0001706

Table of Contents

List of Figures

List of Tables

Acknowledgments

Our research owes much to those who have gone before and to those who have helped make the current work possible. In particular, we would like to express our thanks to Paul Clements and Len Bass, whose work with eliciting architecturally significant business goals paved the way for our research and resulting alignment method.

We would also like to thank Chris Gunderson who provided us the opportunity to discuss the fundamental relationships between software architectures and acquisition strategies and further refine our concepts and method. In addition, our thanks go to Michael Phillips and Jeff Thieret, who reviewed our work and provided their insights. And finally, our special thanks to John Foreman for continuing to believe in this work.

Abstract

In the acquisition of a software-intensive system, the relationship between the software architecture and the acquisition strategy is typically not carefully examined. To remedy this lack, a research team at the Carnegie Mellon University Software Engineering Institute (SEI) has focused a multiyear effort to discover an initial set of failure patterns that result when these entities become misaligned and identify a set of desired relationships among the business and mission goals, system and software architectures, and the acquisition strategy. This report describes the result of the third year of the SEI's research, where the team defined a method that indicates such areas of misalignment (i.e., between a program's architecture and acquisition strategy). The alignment method is used as early in a program's lifetime as practical, ideally before the architecture or acquisition strategy has attained full definition. The authors illustrate the method by means of a case study, during which many of the key elements of the method were piloted.

1 Introduction

1.1 Premise of This Work

The premise of this work is that, by using a method such as the one we propose in this report, organizations, and especially program managers, could avoid some of the causes of failure in government acquisition that we have discovered. In previous work, we showed that many programs fail due to a lack of alignment between system and software architecture and the acquisition strategy [Brownsword 2013a]. Therefore, our research is focused on the relationships between software and system architecture and acquisition strategy; more specifically, we are concerned with their alignment or misalignment. By defining a method that explicitly addresses key entities that are critical to alignment or misalignment, we provide a useful approach for organizations and project managers engaged in acquisition programs to make informed decisions, thus increasing the likelihood of program success.

The key entities of interest are (1) the architectures themselves, both software and system; (2) the planned acquisition strategy; (3) the quality attributes[1] that drive those architectures and strategies; and (4) the goals (both business and mission) of all of the stakeholders. By defining a set of activities associated with these entities and their relationships, we seek to pinpoint major influences that tend either to keep the architecture and acquisition strategy in harmony or to pull them apart.

1.2 Definitions

The key terms used in this report have the following definitions:

- **Mission Goal**: an expression of some *operational* objective that is focused on *what the system should do* or *how it should behave*; sometimes called a "mission driver"
- **Business Goal**: an expression of some *organizational* objective that is focused on *what the acquisition (the business model for how the system will be procured) should do* or *how it should behave*; sometimes called a "business driver[2]"
- **Architecture**: the structure of components, their relationships, and the principles and guidelines governing their design evolution over time (see *IEEE Standard 610.12* and the *Department of Defense Architecture Framework*, or DoDAF)
- **Acquisition Strategy**: 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,

[1] We briefly define "quality attribute" in Section 1.2, and examine the concept in detail in Section 3.

[2] Clearly there are business goals that affect the behavior of the system. For example, system adaptability in response to anticipated changes to minimize long term sustainment costs. Similarly, there are mission needs that will affect the behavior of the acquisition. For example, schedule responsiveness to an urgent mission need for deployed capability.

and other activities essential for program success. The acquisition strategy is the basis for formulating functional plans and strategies (e.g., Test and Evaluation Master Plan (TEMP), Acquisition Plan (AP), competition, systems engineering) [DAU 2011].

- **Quality Attribute**: a measurable or testable property that is used to indicate how well the system, software, or program satisfies the needs of its stakeholders. Adapted from *Software Architecture in Practice*, 3rd edition [Bass 2012].

1.3 What Is Unique About This Approach?

We state at the outset that there are certain aspects of our work that will lead to unfamiliar recommendations about how acquisition programs proceed, particularly in their earliest phases. We fully realize that some of what we propose will differ from familiar acquisition practice.

In particular, we posit that there can be significant elements of an acquisition strategy in existence before there is a named program manager and that there can be significant elements of system and software architectures in existence before there is an architect. Further, we posit that enough is known about each of these artifacts that the relationships between them can be meaningfully analyzed as early as pre-Milestone A. We would argue that the acquisition strategy is formed far in advance of the formal decision to charter a program manager. Further, we assume for the purpose of our method, and as it is described in this report, that the person or team that generates or defends the acquisition strategy is the program manager or his or her proxy who is acting as the strategic decision making authority for a new program. Similarly, there is an architecture in the very earliest shaping of the program. This architecture describes, at a very high level, the elements of the system in a way that allows cost estimation and discussions about how to partition the work among contracts. For purposes of our method and its description in this report, we assume that the person or team that generates this high-level description is either the architect or his or her proxy.

1.3.1 Programmatic Issue

Our justification for the approach we take is based on firm ground. Harvard professor J. Ronald Fox,[3] at the request of the Center of Military History, published in 2011 a comprehensive report: *Defense Acquisition Reform, 1960–2009: An Elusive Goal*, which documents in detail most of the major attempts at acquisition reform that have been made. Fox observes

> *Most attempts to implement improvements in the management of the defense acquisition process during the past fifty years have fallen short of their objectives... There seems to be little hope of solving the chronic problems if the usual attempts at reform are tried once again.*

For that reason, in the approach we have been taking, we have consciously broken with traditional practice. In particular, we urge injection of technical analysis, particularly regarding software, in

[3] Fox served as assistant secretary of the U.S. Army and deputy assistant secretary of the U.S. Air Force. He received the Exceptional Civilian Service Award from the Air Force and the Distinguished Civilian Service Award, the highest decoration for public service awarded by the Department of the Army. He also chaired the Board of Visitors of the Defense Acquisition University, was trustee of the Logistics Management Institute and the Aerospace Corporation, and was director of the American Society for Macro Engineering. In 2004, he was named senior adviser on acquisitions to the Defense Acquisition History Project. In 2006, he was named to the Defense Acquisition University Hall of Fame.

the very early phases of an acquisition, before any commitments pertaining to acquisition strategy are made. Although our proposed method will likely lead to criticism such as “But we don’t do it that way!” and other comparable reactions, we believe that only through such unconventional approaches can the colossal logjam of current acquisition processes and failing programs (or programs that are to some degree over budget and years behind schedule) be alleviated, at least to some degree.

1.3.2 Significance for Systems Engineering

Our work places considerable emphasis on how an acquisition strategy must consider software at the outset. However, readers who have commented on earlier reports have criticized our approach on the grounds that, if the software could just be properly considered by the system engineer, then all would be well.

We concur that a good deal more needs to be done to bring software and system engineering into proper balance. However, today software is responsible for nearly 80 percent of a system’s operational performance [GAO 2011]. In the face of this reality, appropriate concern for software is often delayed until far too late in the system engineering processes, and we seek to partially remedy that. But we do not advocate consideration of the software without due consideration of the system and the other engineering disciplines that the system depends upon: both system and software engineering issues must be considered very early, and must contribute to the acquisition strategy’s definition. In other words, we believe that conflicts among the acquisition strategy and the system and software architectures can be resolved earlier in a program’s formation so that a sufficient degree of consistency among all three (which, in this report, we term “alignment”) can be achieved. As we previously learned, it can be catastrophic for a program if significant conflicts between and among these entities are not identified and resolved until after any of the three have attained maturity.

2 Background

The research that underlies our alignment method is completing its third year. Highlights from each year are discussed in the following sections. In addition, our research has relied on these foundations:

- our recognition and appreciation of the considerable body of knowledge that has emerged from the Software Engineering Institute's (SEI's) ongoing research in software architecture, particularly (1) the significance of an explicit description of a system's required properties, typically called "quality attributes" (e.g., performance), to support the tradeoffs the system and software architect will need to make; (2) a method for generating, documenting, and prioritizing these system properties, as in the Quality Attribute Workshop (QAW) [Barbacci 2003]; and (3) the formal statement of a more complete set of mission and business goals in the Pedigreed Attribute eLicitation Method (PALM) [Clements 2010]
- the ongoing efforts by the Department of Defense (DoD) to improve the acquisition process, particularly in that the business goals of the department have been clearly stated, and that they make the relationship between these business goals and the program's acquisition strategy more explicit
- the SEI's experience with more than 100 independent technical assessments (or ITAs, and often informally called "red teams"), particularly because our team's ITA experiences and those of our colleagues strongly corroborate the basic premise stated at the beginning of Section 1

2.1 Year 1 Research

During the first research phase, we discovered a set of acquisition "anti-patterns" [Brownsword 2013a]. We used the concept of an *anti-pattern*, derived from existing research in the area of design patterns, as a mechanism to characterize recurring modes of failing behavior [Brown 1998]. We studied a large number of troubled acquisition programs through the lens of the ITA experiences of our colleagues, and observed seven common anti-patterns that these programs manifested. Of particular interest were the failing patterns of undocumented business goals, and unresolved conflicts between goals.

We then analyzed the key entities and relationships that made up these anti-patterns. Based on that analysis, we developed Figure 1 as a representation, not of failure, but rather of the *desired* relationships between the key entities. In other words, the presence of one or more anti-patterns was an indication that one or more of the relationships in the diagram were either missing or weak. Central to our work was the realization that just as was true for the earlier research in software architecture, the concept of quality attribute of a *program* was (or should be) of fundamental importance in defining an acquisition strategy

Regarding Figure 1, we note that the lower half of the diagram (i.e., the flow from stakeholders through goals to architectures) is currently a well-defined area of engineering practice, although not well practiced within some organizations; we intend to include that practice in the method we

are defining. Hence, our research focus is on first bringing the top half of the diagram (i.e., the flow from stakeholders through goals to acquisition strategy) into maturity, and then dealing with the vertical relationships between the top and bottom portions of the diagram.

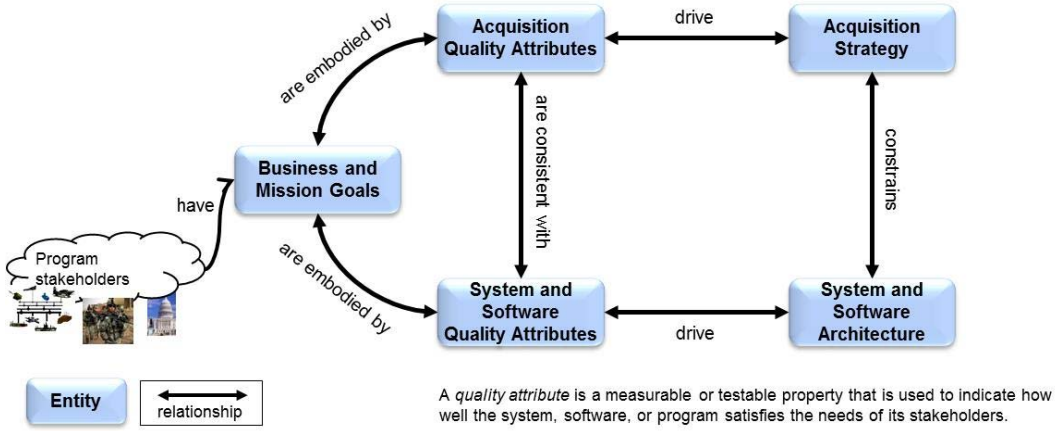

Figure 1: Desired Relationships Among the Principal Entities

We concluded, at the end of Year 1, that

1. There are stakeholder goals for a program (typically, business goals) that drive the acquisition strategy. These goals are a key entity, but these goals are often not sufficiently expressed or captured. One cause for this is that there is no formal process for expressing or capturing business goals, thus making it difficult to analyze these goals for conflicts with other mission or business goals.
2. The business and mission goals will imply quality attributes that have relevance for the acquisition strategy in much the same way that business and mission goals relate to the system and software architecture [Bass 2012].
3. These acquisition-related quality attributes are as important as those derived from the mission goals, and we refer to them as acquisition quality attributes.

2.2 Year 2 Research

Our effort during research phase two was concentrated on the top portion of Figure 1, and was aimed at understanding and demonstrating the applicability of acquisition quality attributes, particularly by means of *acquisition-centric scenarios* [Brownsword 2013b]. Our focus was on the relationships between the business goals, acquisition quality attributes, and acquisition strategy.

We enumerated potential acquisition quality attributes following several approaches. We discovered that just as there are different ways that software quality attributes can be aggregated, so too is the case for the acquisition quality attributes. While it was apparent that acquisition quality attributes will vary in their breadth and generality, we speculated on a number of possible areas in which acquisition-related attributes would be found. For example

- contract issues (e.g., legality, contract manageability, comprehensiveness, appropriateness)
- program management issues (e.g., accountability, management visibility)

- program execution issues (e.g., promptness in reporting problems, responsiveness to government requests, responsiveness to budget changes)

We developed a list of roughly 30 acquisition quality attributes based largely from DoD and other government acquisition strategy guidance and instructions and then augmented by our research. These included acquisition quality attributes such as executability, flexibility, and realism. We decided to leave the possible acquisition quality attributes as an unordered list without concern to generality or specificity.

Following a similar path as that used with the original research on software quality attributes and as did the developers of QAW and PALM, we adopted the principle of using scenarios to give precise meaning to acquisition quality attributes. We explored the construction and use of program-specific scenarios, or small "stories," that specify some event (the "stimulus") that occurs under particular conditions (the "environment") with the desired behavior (the "response") of the program. An example of an acquisition quality attribute scenario 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

We generated 75 acquisition quality attribute scenarios from more than 23 programs. One viable approach we found was through adapting elements of the QAW, developed by the SEI to assist programs in forming their software architectures. We piloted a modified QAW to better understand two aspects: a program's drivers for its acquisition strategy and the applicability of the QAW elements for an acquisition strategy.

We experimented with a number of approaches for the analysis of the scenarios:

- demonstrating how different scenarios result in different acquisition strategies
- examining the role of acquisition tactics in forming aspects of an acquisition strategy, an analog to the architecture tactics defined in software architecture research

Our research also identified five gaps associated with the desired acquisition entity relationship model (Figure 1) that a method should address

1. salient stakeholders are not readily identified and involved, with the consequence that important goals are often overlooked
2. acquisition quality attributes derived from business goals are absent
3. software/system quality attributes are not routinely used
4. quality attributes (acquisition and software/system qualities) are not used to inform acquisition strategies
5. acquisition strategies and architectures typically are developed separately; thus, they are not typically aligned with each other. As we had concluded earlier, conflicts between and among all of these are therefore found too late, when removing the points of conflict is difficult, expensive, or impossible.

In year 2, we concluded that

- we could show a critical link between the acquisition quality attribute scenarios and an acquisition strategy
- by considering acquisition quality attribute scenarios and acquisition tactics together, we could bring to the surface additional issues and drive beneficial changes within a program's acquisition strategy, plans, and related artifacts
- incompatibilities between scenarios, whether acquisition, system, or software, can be distinguished and that the comparisons can reveal potential conflicts within a program
- we knew enough to define a method that would provide a viable path forward to aligning software and system architectures and acquisition strategies

2.3 Current Year Research

The focus was to develop an alignment method that seeks to remedy or at least lessen the impact of those gaps identified in year 2 of our research. To that end, we first performed work on individual topics that we expected would become significant elements or techniques of the alignment method:

- development of techniques for eliciting and structuring goals
- detailed analysis and characterization of various aspect of critical and prevalent acquisition quality attributes (e.g., flexibility, executability)
- examination and development of acquisition tactics
- development of techniques for comparing different acquisition quality attribute scenarios

We then aggregated these elements and techniques into the actual method for examining a specific program in terms of its acquisition strategy and its system and software architectures. In brief, the method begins by eliciting business and mission goals, using them to generate both quality attributes and scenarios that embody those attributes, and then analyzing all three (goals, attributes, and scenarios) for possible conflicts between the architecture and the acquisition strategy that might be implied.

3 Key Artifacts of the Method

There are a number of artifacts that are central to our alignment method: business goals and acquisition quality attributes. While many of these artifacts are strongly related, they are nonetheless conceptually distinct.

3.1 Business Goals

Given their role in our research, it is not surprising that business goals should be of prime importance. Just as we made significant use of previous SEI research in the domain of software architecture, we make extensive use of SEI research in the domain of business goals as they related to architecture. This work resulted in the Pedigreed Attribute eLicitation Method. The goal of this method is

> *… to empower architects to spot the likelihood of missing requirements by giving them a clear and full picture of the operative business goals [and]… to empower architects to be able to question difficult requirements that may not be necessary because they do not support any important business goal* [Clements 2010].

We discussed the PALM with its authors and analyzed many of its concepts and techniques. One essential distinction between our work and PALM was focus: our primary focus concerning business goals was how they would (and should) affect the acquisition strategy. For PALM, the primary focus was on how they would affect the software architecture. That said, many of the concepts and elements of PALM were highly relevant to our research.

3.1.1 Structure of a Business Goal

One essential element of PALM that is particularly relevant for our method is the formal mode of expressing a business goal. We adopted this formal mode for our step on eliciting business goals:

Goal-subject	the stakeholder who owns the goal, who wishes that it be met
Goal-object	the entity to which the goal applies
Goal	the goal itself; this may either be relatively general (“meet financial objectives”) or may be quite specific (“field system in less than six months”)
Environment	the context for the goal
Goal-measure	a measurement for determining whether the goal has been achieved
Pedigree and value	the source (e.g., the individual who stated the goal), the degree of confidence he has in it, and the goal‘s volatility and value

Sub-elements 1–5 can be combined into a sentence that reads

> *For the system being developed, <goal-subject> desires that <goal-object> benefit from <goal> in the context of <environment> and will be satisfied if <goal-measure>.*

The sentence can be augmented by the goal’s pedigree and value (sub-element 6) using such notions as “with a gain of return on investment (ROI) of 30 percent.”

This version of our alignment method is highly dependent on the first three sub-elements for expressing goals: goal-subject, goal-object, and goal. We envision the other three sub-elements playing a greater role in subsequent refinements of our method. Other than the business goal examples shown in this section, we will use the first three sub-elements in this report.

3.1.2 Examples of Business Goals from Government Programs

To illustrate the kinds of business goals that might be applicable to an acquisition, Table 1 lists what a representative set of stakeholders in an actual government program expressed as to what they cared about. Several of these areas of interest are then transformed into the business goal statement structure. The purpose of the program was the development of a large human resource (HR) system that would replace a collection of aging legacy systems.

Table 1: Examples of Stakeholder Interests

Stakeholder	Areas of Interest
Program Manager	• Wants the satisfaction of delivering a quality product that does something important for the department • Wants to showcase the capabilities of his program team • Wants to attract and keep high class talent in his organization • Wants to decrease program risk by avoiding requirements creep (so stay away from users)
System and Software Engineers	• Want assurance that the selected solution is buildable and technically correct
Commercial Off-The-Shelf (COTS) Vendors	• Want their product selected in a high-profile government program (implicitly, to assure themselves of future government work as the government starts to understand how linked their business processes really are) • Want to protect their product's reputation in the face of many horror stories of failed deployments
System Integrator Program Manager	• Wants the satisfaction of delivering a quality product • Wants to maximize the number of contracts they are awarded in a declining budget environment
System Verifiers and Validators	• Want to make sure the system meets its stated requirements • Want to be sure the system is fit for use by the warfighters • Want to make sure the correct system was fielded
System Sustainers	• Want to make sure the system is sufficiently documented • Want to be considered competent to maintain the system • Want to make sure the system can be readily evolved to use new technology
Members of the HR staff (supervisors and those who would use the system)	• Want to minimize changes to existing HR processes • Want to maintain the existing level of control over their personnel systems (implicitly wishing to avoid senior leadership imposing new and unfamiliar work processes)

The program manager's desire to decrease program risk by avoiding requirements creep might be transformed to

Goal-subject:	the acquisition program manager
Goal-object:	user community
Goal:	controlling requirements growth
Environment:	the execution phase of the program
Goal-measure:	costs for changes to requirements are less than xx percent of the total program budget
Pedigree and Value:	owner: acquisition program manager; the user community's mission seems to be changing and budgets are expected to be fixed

Written as a sentence, the goal might be expressed as

> For System HR, the acquisition program manager desires that the user community benefit from controlling requirements growth in the context of the execution phase of the program and will be satisfied if costs for changes to requirements are less than xx percent of the total program budget.

As a further example, the HR staff's desire to minimize changes to existing processes could be captured as

Goal-subject:	HR staff supervisor
Goal-object:	the current members of the HR staff
Goal:	retain the current HR business processes
Environment:	after the new system is fielded
Goal-measure:	user interface and business operation steps are the same as the legacy system
Pedigree and Value:	owner: HR staff supervisor; Joe and Pauline from the HR department as the current HR staff agree

The above goal might be expressed in sentence format as

> For System HR, the HR staff supervisor desires that current members of the HR staff benefit from retaining the current HR business processes in the context of fielding and operations of the new system and will be satisfied if user interface and business operation steps are the same as the legacy system.

3.1.3 Artifacts Used for Business Goal Elicitation

Two additional artifacts are used in our method when business goals are elicited from stakeholders: an abstract list of *goal categories* and an abstract list of *stakeholder categories*. Table 2 shows the abstract goal categories. These categories were adapted from the PALM for government programs. The abstract stakeholder categories are captured in Table 3 and are derived from our team's experience with government acquisition programs.

Table 2: Abstract Goal Categories

Alignment Method Goal Categories	DoD and Other Government Amplification
Managing Quality and Reputation of Products (Includes Meeting Responsibility to System of Systems/ Ecosystems Needs)	Technological maturity or risk, robust product; International interoperation and collaboration Interoperability Key Performance Parameter (KPP), Net-ready KPP; commonality (reuse) and standardization (architecture, data formats, interfaces)
Maintaining Growth and Continuity of Mission Organizations (Includes Improving Mission Process)	Development/growth of operational skills and capabilities Improving operational effectiveness (e.g., point on target)
Meeting Program Management Office Objectives (Includes Meeting Program Manager Objectives and Meeting Financial Objectives)	Get the product out Deliverable commitments, leadership, innovation Cost effective, reduced total lifecycle cost, reduced logistical footprint
Meeting Responsibility to Congress and OSD (Includes Meeting Responsibility to Country, Meeting Responsibility to Organic Institutional Organizations, Meeting Responsibility to Society)	Maintaining program support Military benefit, military worth (utility); political considerations for distribution of work Development/growth of organic capabilities (acquisition skills, logistics skills, test skills, etc.) Small business set-asides, Small Business Innovative Research Program (SBIR) technologies, “green”/ environmental impacts (safety, hazards)
Managing Industrial Base	Industrial base capability to design, develop, produce, support; planned capabilities from other programs

Table 3: Abstract Stakeholder Categories

Stakeholder Category	Example Stakeholders
Development Management and Oversight	• Program management offices • Oversight entities • Senior leadership • State or federal legislators
System and Software Development	• COTS/free and open source software (FOSS) product suppliers • System engineers • Software engineers • Program managers for a system integration company • System verifiers

Stakeholder Category	Example Stakeholders
Deployment	• Managers of hosting environment for the system • Owners of enterprise network systems (e.g., Defense Information Systems Agency, or DISA) • Certifiers and accreditors • Owners of training systems • Owners of interacting systems
Logistics and Sustainment	• Program managers for a maintenance organization • System maintainers • System verifiers
User Populations	• System administrators • Specific kinds of users and operators of the system

3.2 Software and System Quality Attributes

As defined in Section 1, a quality attribute is a "measurable or testable property that is used to indicate how well the system, software, or program satisfies the needs of its stakeholders." For software-reliant systems, software and system quality attributes are sometimes termed "non-functional" requirements or the "-ilities." Examples include performance, dependability, maintainability, or security.

Significant work has been done within the software community to catalog and describe these quality attributes along with techniques for their expression, elicitation, and use in shaping software and system architectures.[4] Community experience has shown that the labels by themselves are not sufficiently detailed for design and design tradeoffs, let alone testing. As previous noted, our alignment method leverages the use of scenarios to provide a more useful meaning to the name or label of a quality attribute. The notation we use for these scenarios, or structured stories, derives from Barbacci where a simple three-part structure forms the basic organization [Barbacci 2003].

Stimulus	condition that affects the system (e.g., initiating or triggering event)
Environment	condition under which the stimulus occurred (e.g., preconditions)
Response	activity that results from the stimulus (e.g., desired system action or capability)

The parts of a three-part scenario can be combined as follows:

If <stimulus> during <environment>, then the system shall <response>.

[4] Readers unfamiliar with quality attributes may find the following items described in the reference section valuable: Barbacci 1995, Barbacci 2003, and Bass 2012.

For example, a software quality attribute scenario for a specific program's quality attribute of availability might be

> If half of the servers go down during normal operation, then the system shall maintain its overall availability.

Often greater information is needed to make effective architectural and design tradeoffs. Thus, for more detailed analysis of priority quality attributes, the three-part structure can be refined to a six-part structure.

Stimulus source	entity that generated the stimulus
Stimulus	condition that affects the system
Environment	condition under which the stimulus occurred
Artifact	artifact that was stimulated
Response	activity that results from the stimulus
Response measure	the measure by which the system's response will be evaluated

Below is an example software architecture scenario, borrowed from Bass [Bass 2012]. It is a scenario that illustrates how a hypothetical stakeholder for a specific program would define what he means when he indicates that he wishes a system to be modifiable (i.e., the quality attribute "modifiability"):

Stimulus source:	the system designer
Stimulus:	wishes to change the user interface
Environment:	during system development
Artifact:	the code
Response:	changes are made and unit tested
Response measure:	in three hours

3.3 Acquisition Quality Attributes

The concept of an *acquisition* quality attribute plays an equally critical part in our method. In this section, we elaborate on the concept, both as it is derived from the parallel concept in software architecture and also as it relates to acquisition strategy. We have adopted the extensive use of scenarios (in our case, termed "*Acquisition* Quality Attribute" scenarios) as a foundational practice. In addition, since we are postulating an unfamiliar concept, namely, the importance of *acquisition*-centric quality attributes, we provide in Appendix A an extended consideration of the two common acquisition quality attributes that emerged in our previous year's work: "flexibility" and "executability."

We have adopted the notion of a scenario as defined in the software architecture research but our purpose is to analyze and evaluate the quality of a *program* as it is characterized in its overarching

business plan, namely its acquisition strategy. Conceptually, acquisition quality attribute scenarios can be used to specify the quality requirements of a program's acquisition strategy.[5]

An acquisition-related scenario has a three-part structure (i.e., the stimulus, environment, and response sub-elements) as well as an expanded six-element structure:

Stimulus source	the person or organization that is the origin of the stimulus
Stimulus	an event that affects the program
Environment	the set of circumstances in which the scenario occurs; this is usually a temporal element
Artifact	the part of the program to which the stimulus applies
Response	the desired behavior of the program (via its participants)
Response measure	the means by which the response will be deemed satisfactory

As an example, we show an acquisition-related scenario that illustrates how a stakeholder for a specific program would define what he or she means, in the context of their program, by the acquisition quality attribute of "responsiveness:"

Stimulus source:	the stakeholders in the field
Stimulus:	request that, in the new iteration of the system, an action triggered from a menu of choices be split into two actions, each with a separate menu button
Environment:	during development of the third iteration of a system
Artifact:	the contract
Response:	the program manager determines from the contractor that new capability can be added and deployed with minimal delay (with no contract modification required)
Response measure:	the additional capability is added to the release within 10 days, and the schedule for the release is delayed no longer than 12 days

An additional aspect of software quality attributes is their relationship with various *architectural tactics* [Bass 2012]; we assert that that notion extends equally well to *acquisition* quality attributes being associated with *acquisition tactics* in forming acquisition strategies. Possible tactics for acquisition strategies are extensively discussed in Appendix A.

[5] Our year 2 research discussed extensively the value of acquisition quality attribute scenarios in helping to form acquisition strategies.

4 Acquisition Trade Space: Cost, Schedule, and Performance

Since we expect, when using our method, to influence how an acquisition strategy[6] is defined, we first must consider the major forces that govern any acquisition strategy; these are the familiar variables of cost, schedule, and performance. These three are often referred to as the "Project Management Triangle:"

> *Like any human undertaking, projects need to be performed and delivered under certain constraints. Traditionally, these constraints have been listed as "scope," "time," and "cost." These are also referred to as the "project management triangle," where each side represents a constraint. One side of the triangle cannot be changed without affecting the others. A further refinement of the constraints separates product "quality" or "performance" from scope, and turns quality into a fourth constraint.*
>
> *The time constraint refers to the amount of time available to complete a project. The cost constraint refers to the budgeted amount available for the project. The scope constraint refers to what must be done to produce the project's end result. These three constraints are often competing constraints: increased scope typically means increased time and increased cost, a tight time constraint could mean increased costs and reduced scope, and a tight budget could mean increased time and reduced scope.*[7]

DoD descriptions of cost, schedule, and performance are often extended to include the notion of risk, which is attached to each of the other three; the Defense Acquisition University (DAU) describes their overall significance in DoD acquisitions as follows:

> *Cost, Schedule, Performance, and Risk are the basic elements through which DoD acquisition professionals make tradeoffs and track program status. Risk cuts across the other three elements (Cost Risk, Schedule Risk, and Performance Risk). In DoDD 5000.01, paragraph 4.2. "The primary objective of Defense acquisition is to acquire quality products that satisfy user needs with measurable improvements to mission capability and operational support, in a timely manner, and at a fair and reasonable price.*[8]*"*

4.1 Definitions

Although what each of these terms means is intuitively obvious, it is worthwhile to cite their precise DoD definitions; the *DAU Glossary* defines schedule, and performance as follows:

> *Schedule: 1.) Series of things to be done in a specific sequence within a given period. 2.) A timetable. 3.) A listing of activities and events organized by time.*

---

[6] Where an actual program has not been formed, critical strategic decisions are made that will affect the acquisition strategy, software architecture, and system architecture. Even at this stage, cost, schedule and performance tradeoffs are made and would benefit from the considerations surfaced by the alignment method.

[7] http://en.wikipedia.org/wiki/Project_management_triangle

[8] https://dap.dau.mil/aap/pages/qdetails.aspx?cgiSubjectAreaID=9&cgiQuestionID=115982

> *Performance: Those operational and support characteristics of the system that allow it to effectively and efficiently perform its assigned mission over time.*

The *DAU Glossary* does not separately define Cost, but DoD 5000.4-M, *Cost Analysis Guidance and Procedures*, December 1992, defines seven standardized cost terms. These seven cost terms are development cost, flyaway cost, weapon system cost, procurement cost, program acquisition cost, operating support (O&S) cost, and lifecycle cost.[9]

And finally, and to the purposes of this report, the *Defense Acquisition Guidebook* notes that

> *The Acquisition Strategy is an appropriate place to discuss cost, schedule and performance implications or trades related to risks and risk mitigation.*[10]

4.2 How Cost, Schedule, and Performance Relate to Acquisition Scenarios

As is described in Section 5, we examine these three key values during the analytic phases of our method. What we seek to determine is the degree to which the scenarios indicate unanimity of purpose among the program's participants, since we believe that it is impractical for all of these three variables to be equal in importance.[11]

As an example, consider the following scenario:

Stimulus:	an independent team evaluates the system and finds that not all requirements are satisfied
Environment:	during developmental testing
Response:	the contractor must use internal funds to make sufficient fixes until all requirements are satisfied

This scenario indicates that the key values are cost and performance; schedule is obviously of lesser weight. If, however, the context of the program was one of great urgency, and the users in the field simply want *anything* that will work just well enough, then the scenario is at odds with that context, and its importance with respect to framing the acquisition strategy would be lowered appropriately.

As stated, however, the implication for the acquisition strategy would be the need to establish clear criteria for system acceptance; explicitly define threshold and objective requirements (and clarify that the threshold requirements must all be met); explicitly define quality (e.g., as measured by no class 1 or class 2 defects); make the key performance parameters sufficiently well-defined so they can be measured; and finally, in this particular case, use a fixed price contract.

---

[9] https://dap.dau.mil/aap/pages/qdetails.aspx?cgiSubjectAreaID=9&cgiQuestionID=115982

[10] *Defense Acquisition Guidebook*, September 16, 2013

[11] This issue is controversial, and although the common wisdom is "two but not three," there are proponents on both sides of the question. http://en.wikipedia.org/wiki/Daniel_Goldin shows one view, and http://www.dau.mil/pubscats/atl docs/mar-apr10/ward_mar-apr10.pdf shows the opposing view.

In the detailed description of the method in Section 5, we will describe other examples of how, just as architecture-based scenarios have implications for the software and system architecture, it is equally true that acquisition-centric scenarios generally have implications for the acquisition strategy. As we will also note, it is very often the case that an architecture scenario can also have implications on the acquisition strategy, and an acquisition scenario have implications for the architecture. This is a reflection of our foundational premise: those two entities are necessarily related, and the degree to which they are aligned and mutually constrained is of vital importance for the success of a program.

5 An Alignment Method

5.1 Pragmatic and Logistical Considerations

When to Use the Method

Given that the method is designed to lead to alignment between acquisition strategy and architecture, the best time to employ the method is prior to major acquisition-significant events. Therefore, we expect that the method could be used prior to Milestone A, or prior to Milestone B, or before significant acquisition events such as a major upgrade to an existing system. Note that, if the method is used before Milestone A, there is the possibility that a program office may not yet exist. But even though a program office (and by extension a program manager) does not yet exist, considerable work must be done to prepare for a Milestone A decision. Therefore, the person (or persons) who were primarily responsible for the Milestone A preparation work would be considered a proxy for the program manager in our method description. This reliance on a proxy also applies to key roles such as the system and software architects.

We note that the method, when used before Milestone A, will become a compact, focused encounter between the key decision-makers when a program is in its earliest days. We believe that, while this is not the most representative way to use the method, it is nonetheless perfectly reasonable. If properly executed, it can reveal, at a very early stage, potential conflicts among the program's foundational elements that might otherwise not appear until considerably later in program execution.[12] The case study that illustrates use of our method is in fact an instance of such usage.

What the Method Produces and How the Products Could Be Used

As shown in Figure 1 from Section 2.1, our ideal for the relationship among critical entities, we expect that the quality attribute scenarios, once they have been developed, can be used to inform the development or maturation of both the architectures and the acquisition strategy. These scenarios provide useful data on events (stimuli) to which the program and the system must respond and, also, the nature of the desired response. If and when the acquisition, system, and software quality attribute scenarios are aligned with each other, meaning that contradictions among them have been resolved, we believe they are more likely to lead to acquisition strategies and architectures that are in alignment.

Who Should Participate

This method works best when as many of the stakeholders as possible participate with the people responsible for the acquisition strategy and the system and software architectures. Depending on when the method is used, the persons named below may not have been identified, in which case, their proxies should participate. At the very least, the program manager and the system and software architects must participate in every phase. If the method is being used later in the program's

[12] A corollary is that demonstrable absence of conflicts is an indication that the program is feasible and should be of interest to decision makers.

lifecycle (e.g., before a major upgrade), other recommended participants could include the contracts manager, the senior test engineer, and one or more representatives of the user community.

Who Should Lead the Method

We expect that use of the method will be led by a team experienced in both acquisition and architecture and, of course, the method itself. Such a team could be constituted from the program office, but we recommend using an independent team unbiased by history in the program office. We expect that this method can be led by trained government personnel (or their consultants or contractors) though in early uses the team would comprise experienced persons from the SEI.

5.2 Introducing our Alignment Method

Our alignment method consists of these phases:

- Phase 1: Identify stakeholders and elicit business and mission goals
- Phase 2: Elicit quality attribute scenarios (both architecture- and acquisition-focused)
- Phase 3: Analyze goals, attributes and scenarios for misalignments
- Phase 4: Define/create remediation for misalignments (subject for future research)
- Phase 5: Validate that architecture and acquisition strategy are aligned (subject for future research)

Each of the following narratives for the phases includes a short description of the objectives and related information, the inputs and outputs, any assumptions, the primary participants, and the tasks used to achieve the objectives. These narratives reference Section 3 (method artifacts) and Appendix C (method templates).

In addition, as we describe the method, we will illustrate its use by means of a case study based on a real government program. Though our method was not codified, the program has relevance since we found that the interactions among the program participants were implicitly the actions and steps that have since become formalized in our method. The terminology we use in the case study reflects our method and is not necessarily how this program would have described these events.

This case study describes a program that was focused on a set of three systems, which for this report we term System A, System B, and System C. The three systems interoperate with each other and, collectively, provide mission-critical functionality. While all three systems were aging, two of them (B and C) were of particular concern: no maintenance contract existed for them; they were written in an obscure programming language; and the government office responsible for maintaining them had extremely limited funding. There was, in the user community, a huge desire to replace all three systems before they became unusable. The decision was made to create a new system, called System ABC that would replace Systems A, B, and C. The case study describes events that occurred in the early stages of that effort, before any major decisions had been made.

5.3 Phase 1: Identify Stakeholders and Elicit Business and Mission Goals

Description: During this initial phase, we perform a formal elicitation of goals, both business and mission. In the process of eliciting these goals, we use them as a mechanism to further identify relevant stakeholders and, therefore, additional business and mission goals. These goals will be used in subsequent method phases.

Inputs

- Identification of relevant stakeholders. At a minimum, the program manager, system architect, and software architects or their proxies[13]
- Sets of abstract goal categories and abstract stakeholder categories[14] to stimulate the considerations of persons interviewed

Outputs

- A set of business and mission goals that represent the foundational assumptions associated with the program (captured in Business/Mission Goal Template)
- A set of relevant stakeholders associated with the program (captured in Master Stakeholder Template)

General assumptions

- The process steps will be cyclic, recursive, and convergent; they will be carried out until sufficiency is achieved
- The method evaluation team will need either access to, or a surrogate for, every stakeholder who is believed to have significant business goals

Participants

- The program manager is the individual with the greatest need for the results of this phase and would be an ideal participant. Pragmatically, the pressures on program managers are such that their participation may need to be limited to the more critical steps.
- One or more program office staff
- Identified relevant stakeholders as the goal subjects who can be interviewed (in person or by phone) serially or in groups as dictated by their availability
- The evaluation team responsible for carrying out the method

[13] As noted above, if the method is to be used prior to Milestone A, and there is as yet no program office, we will need to identify the person or persons who created the materials and documents for the Milestone A decision.

[14] These abstract categories are discussed in Section 3.

Process tasks

❖ Task – Interview available stakeholders using the following task steps for each interviewee:

	Task Step	Activity
1	Introductions and Objectives	Introduce the purpose of the method and the objectives for this task. Record the name of the stakeholder and date of interview on a master list of stakeholders.
2	Identify Stakeholder Goals	Query the stakeholder about goals for the program and any pertinent knowledge about other relevant stakeholders.
3	Characterize and Record Goals	Record each stated goal on a master list of goals using the format of <goal subject> - <goal object> - <goal>.
4	Identify Additional Stakeholders	Record all newly identified <goal subjects> and <goal objects> that are persons as stakeholders on the master list of stakeholders including name, position, rank, and authority.
5	Identify and Characterize Additional Goals	Determine whether it is possible to interview that person. If so, schedule an interview. If not, query the stakeholder at hand as to his belief of what <object>'s goals would be and record those goals on the master list of goals.
6	Repeat Stakeholder and Goal Identification As Needed	If any of the goals elicited from the surrogate have <goal objects> that are persons, repeat steps 4 and 5, and continue recursively until sufficiency is achieved.

❖ Task – Determine sufficiency of goal and stakeholder information using the following criteria:

	Criteria
1	Every category in the abstract stakeholder categories has been considered, and
2	every category in the abstract goal categories has been considered for applicability, and
3	goals have been elicited from every named stakeholder on the master list of stakeholders, and
4	stakeholders on the master stakeholder list cover all of the applicable stakeholder categories, and
5	goals on the master goals list cover all of the applicable categories in the pre-existing goal categories, and
6	the program manager concurs that the master lists of goals and stakeholders are sufficient.

The issue of sufficiency is highly important. In general, having more of the goals is better as it reduces surprise later in the process. But practically speaking, it is likely that there will only be a subset of the goals uncovered. The point is that the more diligence spent at this point in the process, the better the program will be informed and the more likely it will achieve its goals.

Artifacts Applied to Phase 1 in the Case Study

During our case study, the participants surfaced many business and mission goals. These evolved into a large number of goal statements that were used in the analysis. To illustrate the alignment method in practice, we select a small number of these goals. These were chosen for two reasons: they illustrate (1) how goal analysis can indicate favorable elements of a program; and (2) how goal analysis can indicate conflict or misalignment in the same program.

Of the goals of interest, three were expressed as business goals, and the fourth as a mission goal:

- *Goal of representatives of the acquisition community*: wish to follow the DoD policy of avoiding vendor lock for all new acquisitions; this would definitely include System ABC

- *Goal of representatives of the end users*: want to replace System B and System C with new System ABC before B and C become unmaintainable
- *Goal of the chief financial officer of the organization acquiring System ABC*: wants to acquire System ABC for less than $10 million
- *Goal of the chief information officer of the organization acquiring System ABC*: wants System ABC to conform to an enterprise architecture with which all new DoD systems must be in conformance

If we express these goals in formal terms, they become[15]

Goal ID	Goal Statement
G1	The acquisition community <goal subject> has a policy <environment> to avoid vendor lock <goal> for all new acquisitions including System ABC <goal object>.
G2	The users <goal subject> want to replace Systems B and C <goal>with new system ABC <goal object> before B and C become unmaintainable <environment>.
G3	The chief financial officer <goal subject> has stated <environment> that system ABC <goal object> must be acquired for less than $10 million <goal>
G4	The chief information officer <goal subject> specified conformance to an enterprise architecture (<environment> to which all new systems <goal object> must conform <goal>.

5.4 Phase 2: Elicit Quality Attributes Scenarios

Description: Using the business and mission goals from Phase 1, this phase identifies and characterizes potential quality attributes (acquisition, software, and system) scenarios that if satisfied could achieve the goals. For this phase, an external team facilitates several structured discussions using a workshop setting. The expected participants include the program manager, system and software architects, and other available stakeholders (identified in phase 1) or their proxies. During the workshop the quality attributes (acquisition, software, and system) are identified and the scenarios representing these quality attributes are generated, prioritized, and refined.

The workshop consists of two parts: one where the emphasis is on acquisition quality attributes and a second where the emphasis is on system and software quality attributes. In considering the planning for the workshop, two issues arise:

- Will these two workshop parts be separate events, or will they be separate sections of a single event? Cost, logistics, and practicality are the drivers that will determine this. For simplicity, we describe them here as two separate events, but we note that it may be necessary to make them subcomponents on just one actual visitation. But whether one or two events, it is necessary that some of the participants are different. For this reason, holding two events may be preferable but isn't mandatory.
- Which part of the workshop should come first? Though there might be instances where one should precede the other, it is more likely that either can come first. In this report we will describe the part that focuses on the acquisition quality attributes first.

[15] For simplicity, we omit the goal-measure and pedigree, since neither affected the decisions and conclusions we reached in our analyses.

Inputs

- set of business and mission goals (from phase 1)
- set of stakeholders (from phase 1)
- at a minimum, a rudimentary form of the system and software architectures (may be only at the highest level of the presumed major components)
- at a minimum, a rudimentary form of the acquisition strategy (may be only at the highest level of the presumed elements and approaches)

Outputs

- matrix of goals with potential quality attributes (acquisition, software, system) that would achieve those goals (Goal-Attribute Matrix)
- acquisition quality attribute scenarios
  - set of prioritized scenarios (Candidate Scenarios Template)
  - set of refined high priority scenarios (Refined Scenario Templates, one for each scenario refined)
- software and system quality attribute scenarios
  - set of prioritized scenarios (Candidate Scenarios Template)
  - set of refined high priority scenarios (Refined Scenario Templates, one for each scenario refined)

General assumptions

- Logistics and planning for the workshop are satisfactorily completed such that the right participants are available and motivated and the facilitators are familiar with the proposed program.
- The steps of the process tasks for the two parts of the workshop are essentially the same. To avoid any confusion, both parts are described with any key differences noted.
- The steps of the process tasks are adapted from the mechanisms of the QAW [Barbacci 2003].

Participants

- key program office staff such as the program manager, system architect, and software architect
- evaluation team responsible for carrying out the method
- key stakeholders such as a chief engineer (from the program office and the contractor if one exists), a tester, the IA community, people representing other systems with which the system has to interact, and representatives of the user community. While none of these stakeholders are essential for the system and software and system quality attribute workshop, their presence would be helpful.
- key stakeholders such as a cost estimator, a contracts specialist, a budget specialist, and other representatives from the program office (including from a contractor program office if one exists). Again, the presence of these stakeholders will not be essential to the acquisition quality attribute workshop but their presence would be helpful.

Process tasks

❖ Task – Conduct quality attribute workshop part 1—**acquisition** quality attributes

Note that the task steps described below mirror those discussed with part 2 of the quality attribute workshop. The primary difference here is the focus on acquisition quality attributes and the acquisition strategy.

	Task Steps	**Activity**
1	Introductions and Workshop Presentation for Part 1	Workshop facilitators describe the motivation for the workshop as a whole, highlighting the relationship of the two parts along with the inputs and outputs. Each step of part 1 is explained along with relevant examples.
2	Review of Business/Mission Goals	Using the goals from phase 1 that were recorded in the Goal-Attribute Matrix, the facilitators and stakeholders review the goals, adding or modifying as appropriate. Any changes are captured in the Goal-Attribute Matrix.
3	Identification of Quality Attributes	For each of the goals in the Goal-Attribute Matrix, the facilitators and stakeholders identify which acquisition quality attributes would need to be satisfied to achieve the goal. The Goal-Attribute Matrix is updated accordingly.
4	Scenario Brainstorming	Stakeholders generate real-world scenarios for the program's acquisition strategy. Scenarios comprise a related stimulus, an environmental condition, and a response. The scenarios are captured in a Candidate Scenarios Template. Facilitators ensure that at least one scenario addresses each of key acquisition quality attributes identified on the Goal-Attribute Matrix.
5	Scenario Consolidation and Prioritization	Scenarios that are similar in content are consolidated. Stakeholders prioritize the scenarios through a voting process.
6	Scenario Refinement	The top four or five acquisition scenarios are further clarified and for each, the following information is captured in a Refined Scenario Template • goal(s) that are affected by the scenario • the relevant quality attribute(s) associated with the scenario • possible implications for the acquisition strategy including possible acquisition tactics • questions or issues that should be considered further and resolved

❖ Task – Conduct quality attribute workshop part 2—**software/system** quality attributes

Note that the task steps described below mirror those discussed with part 1 of the quality attribute workshop. The primary difference here is the focus on software and system quality attributes and the architectures.

	Task Steps	**Activity**
1	Introductions and Workshop Presentation for Part 2	Workshop facilitators describe the motivation for the workshop as a whole, highlighting the relationship of the two parts along with the inputs and outputs. Each step of part 2 is explained along with relevant examples.
2	Review of Business/Mission Goals	Using the goals from phase 1 that were recorded in the Goal-Attribute Matrix, the facilitators and stakeholders review the goals, adding or modifying as appropriate. Any changes are captured in the Goal-Attribute Matrix.

	Task Steps	Activity
3	Identification of Quality Attributes	For each of the goals in the Goal-Attribute Matrix, the facilitators and stakeholders identify which software/system quality attributes would need to be satisfied to achieve the goal. The Goal-Attribute Matrix is updated accordingly.
4	Scenario Brainstorming	Stakeholders generate real-world scenarios for the program's acquisition strategy. Scenarios comprise a related stimulus, an environmental condition, and a response. The scenarios are captured in a Candidate Scenarios Template. Facilitators ensure that at least one scenario addresses each of key software/system quality attributes identified on the Goal-Attribute Matrix.
5	Scenario Consolidation and Prioritization	Scenarios that are similar in content are consolidated. Stakeholders prioritize the scenarios through a voting process.
6	Scenario Refinement	The top four or five software/system scenarios are further clarified and for each, the following information is captured in a Refined Scenario Template: • goal(s) that are affected by the scenario • the relevant quality attribute(s) associated with the scenario • possible implications for the software and system architectures including possible architectural tactics • questions or issues that should be considered further and resolved

Artifact Applied in Phase 2 in the Case Study

In our case study the business goal of low cost for replacing two of the existing three systems (G3) was linked[16] to an affordability acquisition quality attribute. A scenario exemplifying this acquisition quality attribute was

Scenario ID	Scenario Statement
S1	At least two of the systems must be replaced during times of limited available budget for one-tenth of predicted cost to replace their functionality if developed from new.

Similarly, the business goal for rapidly replacing the two systems most at risk (G2) was linked to an executability acquisition quality attribute; a scenario exemplifying this acquisition quality attribute was

Scenario ID	Scenario Statement
S1	A change is required for the failing systems when no one is able to make that change and the systems are replaced by a new system with equivalent functionality.

Regarding both of these scenarios, the workshop participants made note of an existing system with similar functionality to that needed by System ABC and hence the possibility that this existing system might, in some way, be reused to form a part of ABC. Thus, when it came to prioritizing and refining scenarios, S1 and S2 were grouped together.

[16] We are not ascribing any special meaning to the term "linked"—it simply represents the association in the minds of the participants between the goal of achieving a low cost replacement and the affordability quality attribute.

The business goal to avoid vendor lock (G1) was linked to the flexibility attribute; one of the related scenarios was

Scenario ID	Scenario Statement
S3	A contractor developing the new system raises its rates significantly when the budget is already stretched; the program office replaces the contractor with a different one and stays within budget and schedule.

Scenarios S1, S2, and S3 were all developed in the acquisition quality attribute workshop. The software quality attribute workshop developed many scenarios; the one derived from the goal regarding the enterprise architecture (G4) exemplified the software/system quality attribute of performance:

Scenario ID	Scenario Statement
S4	The CIO office performs an architectural assessment of all new programs with respect to conformance to the enterprise architecture and System ABC is found to be appropriately componentized.

One important aspect of this final scenario is that the need to conform to the enterprise architecture was binding on all new DoD systems.

5.5 Phase 3: Analyze Scenarios for Misalignments

This phase is the heart of the method. As we discuss it, we note that one very significant issue about this phase concerns the intellectual basis on which this part of the method rests. In short, regardless of the precision of the methodological framework that we describe below, it cannot be denied that a good deal of our proposed analysis depends on expert judgment. In stating this, we note that other comparable analysis methods[17] rested on a similar intellectual basis. We do not believe that this reliance invalidates our approach: many individuals, from our own institute and other comparable organizations, are often called on to exercise expert judgment. And in the context of the overall method we propose, we are positioning this expert judgment in a formal, well-defined framework. Hence we feel confident that the approach is reasonable.

Description: The experts analyze, both individually and collectively, a selected number of scenarios, both architecture- and acquisition-centric. The analysis will seek to reveal whether any inconsistencies between multiple scenarios implied misalignment between the architecture and the acquisition strategy.

Inputs

- the full set of scenarios from the Phase 2 workshop (part 1 and part 2), which has been prioritized as one of the final workshop activities; the goal(s) and attributes from which each scenario has been derived

[17] See *Identifying Commercial Off-the-Shelf Product Risks: The COTS Usage Risk Evaluation* (CMU/SEI-2003-TR-023) [Carney 2003]; *Software Risk Evaluation (SRE) Method Description* (Version 2.0), (CMU/SEI-99-TR-029) [Williams 1999].

Outputs

- a list, each element of which is a grouping of two or more scenarios that appear to be in conflict in some manner (scenarios can appear in multiple groupings)

General assumptions

- Although we do not so indicate in the process tasks described below, as the analysis progresses, early analyses will be reexamined in light of later analyses.
- Misalignment or conflict may be present either between acquisition strategy and architecture, or purely within any of these entities (e.g., the strategy may be in conflict with itself).

Participants

- program manager
- system architect and software architect
- evaluation team responsible for carrying out the method

Process tasks

❖ Task – Determine appropriateness of the set of high priority scenarios:

For this task, each quality attribute workshop (i.e., part 1 and part 2) will have reduced its set of scenarios to a smaller number of the highest priority scenarios. At this point, the method evaluation team must determine whether the aggregate set of scenarios is tractable, or must be further reduced. This decision will likely be made based on practical criteria such as the amount of time available for analysis.

❖ Task – Perform a pair-wise comparison of the scenarios. For each scenario pair:

	Task Step	Activity
1	Identify External Factors	Examine the scenarios in terms of any conflicting relevant external factors (see further discussion below and Appendix B)
2	Identify Potential Conflicts	Examine the scenarios in terms of conflicting quality attribute tactics implied by the scenarios
3	Record Potential Conflicts	Record the potential conflict (e.g., on an “Potential Conflict List”)

Pair-Wise Scenario Comparison

This task involves comparing multiple scenarios, seeking to determine whether any conflicts exist between them. One useful mechanism for comparing two scenarios can be exemplified by the familiar friction between security and performance. For instance, if, for some program, there is a scenario that depends on some high rate of performance, and another scenario that depends on some high degree of security, we will generally make an immediate prediction that these scenarios are likely to collide. The basis of this prediction is that most experienced software engineers actually have some historical experience to rely on about the friction between the two (i.e., high security typically involves complex authentication and authorization processes that can require a number of accesses to remote databases requiring significant overhead time; a performance scenario typically calls for a high number of transactions in a very small time span.) In short, we are exam-

ining both of these scenarios with regard to some additional factor, in this case, our knowledge about time required for multiple security-related accesses.

Therefore, in analyzing scenarios for conflicts, we often rely on such additional factors (e.g., the time implied by a given scenario) to illuminate the existence of potential collisions between different scenarios. In all cases, we are making conjectures, but ones that are based on historical knowledge. Some of these factors might be

- **Time**: Can we conjecture the speed at which both scenarios will likely unfold?
- **Authority**: Can we conjecture about any applicable decision-making authority that is required to execute both scenarios?
- **Capacity**: Can we conjecture any spatial implications in the scenarios that are in conflict?
- **Cost**: Can we conjecture the actual cost of executing both scenarios that might make it difficult or even impossible to accomplish both?
- **Practicality**: Can we conjecture whether it will be realistically possible for both scenarios to successfully execute?

Another useful technique for scenario comparison relies on considering the quality attributes that generated them, and in particular, the most likely tactics implied by those attributes. Using this approach, we seek to determine whether or not the tactics suggest (either directly or through the stimulation of expert knowledge) conflicts within the scenarios.

Phase 2 has produced some number of scenarios, each of which is associated with one or more quality attributes. These attributes can now be used to consider the range of possible tactics and then choosing those tactics which are most likely to be adopted. Note that if a given tactic applies to many scenarios (i.e., makes many scenarios possible), then it is more likely to be adopted by the program than a tactic that only makes one scenario possible or, worse, makes many other scenarios impossible.

Artifacts Applied in Phase 3 in the Case Study

We illustrate these techniques by describing their application in our case study. First, we recall the scenarios from the Quality Attribute Workshops that were of particular interest. Three were developed in the acquisition quality attribute workshop and a fourth scenario (S4) was developed during the software quality attribute workshop.

Scenario ID	Scenario Statement
S1	At least two of the systems must be replaced during times of limited available budget for one-tenth of predicted cost to replace their functionality if developed from new.
S2	A change is required for the failing systems when no one is able to make that change and the systems are replaced by a new system with equivalent functionality.
S3	A contractor developing the new system raises its rates significantly when the budget is already stretched; the program office replaces the contractor with a different one and stays within budget and schedule.
S4	The CIO office performs an architectural assessment of all new programs with respect to conformance to the enterprise architecture and the ABC system is found to be appropriately componentized.

We first made a pair-wise consideration against such external elements as time, authority, and so forth. Since we are restricting this illustration to only four scenarios, only the issue of practicality was the external factor seemed especially significant, e.g., the likelihood of S1 and S2 both being successfully achieved seemed highly impractical. Managing a cost reduction of 90 percent at the same time as fielding a system that is a perfect match for the users' needs is not commonly observed in acquisition practice. However, this issue did not appear (as yet) to be an out-and-out conflict.

But when we then examined the scenarios together with the likely tactics associated with their respective quality attributes, two important issues arose:

1. We noted one very significant area of harmony. Scenario S1 was derived from the business goal (of the CFO) of keeping costs low; this is associated with the acquisition quality attribute of affordability. Similarly, S2 was derived from the business goal (of the end users) for rapidly replacing the two systems most at risk; that is associated with an acquisition quality attribute of executability. An obvious tactic related to both affordability and executability is to reuse an existing system as the basis for the new system. This type of concord was a hopeful sign for the program.
2. We noted a significant misalignment. As we have already stated, discussion with the participants revealed that they were already aware of an existing COTS-based system with functionality very similar to that needed by System ABC; the possibility of reusing it had already been discussed, and it was expected that this system could be reused to form a major part of System ABC. In fact, it had become an explicit assumption that this would prove the strategy for the program, hence these two scenarios had already been conceptually grouped.

The evaluation team then considered the other two scenarios. Scenario S4 was essentially a binding requirement that the architecture of System ABC would have to be componentized; scenario S3 implied that the program office would adopt the tactic of awarding contracts for different components to different contractors. At first glance, these two tactics are not in conflict with each other. At this point, however, one of the participants voiced a question as to whether the system expected to be reused as the basis for System ABC (i.e., the system on which S1 and S2 focused) could be cost-effectively split into multiple pieces (as the mandated enterprise architecture demanded), or even split sufficiently that it could at least partially provide the desired new components.

This provoked taking the step of performing an architectural assessment of the system in question; this was done with the participation of the COTS vendor. The assessment showed unequivocally that splitting the system into pieces was infeasible without great expense and further, that it did not provide all of the necessary functionality to sufficiently support all of System ABC. And finally, even if it had all of the functionality of ABC, the goal of avoiding vendor lock would still not have been satisfied.

As a result, the program manager opted not to pursue the hopeless quest of using the system in question, and chose to seek out some other reusable components. Although there was urgency, notably by the user community, to replace individual System A, System B, and System C, the program manager made the command decision to avoid starting an infeasible program, and in-

stead continued the search for a reusable candidate that truly could be the basis for the desired system.

6 Conclusions and Future Work

In Section 1, we stated our initial premise:

> The premise of this work is that, by using a method such as the one we propose in this report, organizations, and especially program managers, could avoid some of the causes of failure that we have discovered.

Having observed the method exercised, at least partially, in the case study, we conclude that the premise is sound: the method works to uncover misalignments that exist. This in itself is, we believe, a valuable contribution to acquisition programs. While we make no claims that it would indicate every possible misalignment, nor every other kind of programmatic defect, we feel that, with further refinement, it could be a beneficial initial step for government programs to take.

But additionally, we have also observed that when the elements of a program are exposed in the manner we recommend (e.g., the full slate of goals from stakeholders at all levels are exposed for all to see), there are valuable side effects as well. One example is seen in the traditional division of a program into the acquisition and the technical spheres. It is common that the personnel who write the acquisition strategy have little or no contact with personnel who define the system architecture. It is obvious that given this isolation, it is hardly surprising that differences both of opinion and understanding can exist, and that those differences can later show up as misalignments.

Hence, the method has the valuable side effect that people who traditionally are isolated into one or the other side can more easily become aware of each other's goals and the wide array of qualities that different people expect both the program and the system to exhibit.

The method also adds to breadth of understanding on the part of the program manager. Since he or she participates in every step of the method, he or she learns that some high-level goals may not be shared by stakeholders at all levels. The program manager learns that conflicting goals will very possibly exist, and which could at some future period threaten the stability of the program. And it adds to increased understanding on his or her part of what the implications of a given action might be. This last should prove even more valuable as a program evolves, since over a multiyear time span, many elements of a program's inception are forgotten. Given that one output of the method includes extensive documentation of every goal, scenario, attribute, and tactic that the method unveils, there can be a valuable record that will prove a useful reminder when years have passed and memories have dimmed.

Further Definition of Phases for the Method

Although much of the work on defining and describing an alignment method was completed, formulization of the latter two phases remains to be done:

- Phase 4: define/create remediation for misalignments
- Phase 5: validate that architecture and acquisition strategy are aligned

Phase 3 results in a collection of misalignments between goals or quality attribute scenarios (acquisition, system, and software). Phase 4 is envisioned to analyze each of the identified misalignments to identify the stakeholders affected and appropriate courses of action to resolve those misalignments. This phase would include characterizing potential tradeoffs and their impact.

Phase 5 is then envisioned to provide techniques to validate the alignment of the architectures and the acquisition strategy. Further research is needed in this area, although we believe that our research and results in pair-wise comparison of quality attribute scenarios and acquisition tactics provides a solid starting base.

In sum, we believe that acquisition programs can derive significant value from this initial version of the alignment method in that phases 1 through 3 bring to the surface critical issues and information and foster greater cross-discipline and cross-stakeholder communication.

Appendix A: Extended Descriptions of Critical Acquisition Quality Attributes

The acquisition quality attributes of flexibility and executability emerged as the most prevalent and important acquisition quality attributes both from our prototype acquisition quality attribute workshop as well as the scenarios from ITAs that we examined. We reiterate, however, the caution we mentioned earlier (i.e., the possibility for disagreement about which quality attribute a particular concern belongs to).The boundaries between quality attributes are not well-defined, and what one person perceives as flexibility is often what another person perceives as executability, and vice versa. In the discussions below, therefore, we are merely examining some very general concepts, first through the lens of one acquisition quality attribute and then the other. But only when an attribute is embodied by program-specific scenarios will it have any objective meaning.

Flexibility

Regardless of the circumstances that attend the start of a given acquisition program, it is inevitable that those circumstances will change, often radically. A program that begins at a time of fiscal plenty can be affected by a nationwide housing crash that brings on a recession with associated budget cuts. A program that begins with a vision of methodical research is victim to a sudden crisis where operational capabilities are needed yesterday. A program that begins with a specifically Air Force mission meets an overpowering political demand to become a joint program, with particular special requirements for every branch of service. A program's driving objective shifts from providing a tactical advantage to delivery of increased reliability.

Flexibility Defined

In brief, flexibility is about reacting to the need for change, whatever its source and nature, and reacting in a favorable manner to that need. Among the definitions in *The Free Dictionary*[18] the one most appropriate for our purposes is: "Responsive to change; adaptable: a flexible schedule."

More importantly, however, in the context of an acquisition program, a brief definition of flexibility might be

> *The ability of the program to respond favorably to unplanned or unanticipated events or circumstances that are likely to occur during the program's lifetime.*

In truth, the program as a whole is not the entity that responds; instead, one or more of its constituent entities does the actual "flexing." If it is more than one entity, they may need to flex in some coordinated fashion.

[18] http://www.thefreedictionary.com/flexibility

In order to factor flexibility into an acquisition strategy, we must consider several questions:

- ***What is the cause?*** What condition or circumstance might demand the need for flexibility? It could be simply related to large-scale issues of budget or schedule, or may be more subtle, such as a changing political landscape, or a change in command personnel that shifts from a methodical approach to one of immediate fielding of capability. In all cases, it will be critical to fully understand the cause of the need for flexibility in order to properly respond to it.
- ***What artifact is involved?*** If flexibility is needed, it is inevitable that there will be one or more artifacts associated with an acquisition strategy or plans that will have to respond to the needed change. This could be the contract(s), but the change can also be entirely programmatic, such as a reorganization of management structure within the program office, addition of a new oversight agency, or a contract mechanism.
- ***What is the desired response?*** This is the person or persons who must take some unplanned or unanticipated action; possible persons include: program manager, contractor, end users, sponsoring agent, or oversight/authority agent. In the widest sense, the kind of actions that will be taken usually fall into the budget/schedule/performance categories (e.g., the planned budget for the program is reduced, or the schedule tightened). However, circumstances are often more subtle. For instance, for some political purpose, a program manager might be strongly (and quietly) urged, just as a request for proposal (RFP) is about to be released, to avoid using a particular contractor. The simplest response would be to revise the RFP away from that particular contractor's strong points. (Note that while this might not initially have a significant effect on budget or schedules, the decision to revise the RFP such that the unwanted contractor has little chance of winning might later have negative impact on budget, schedule, or both.)
- ***How will the response be measured?*** Making any significant course change in a program will have an impact on both budget and schedule, and depending on the change, performance as well; the key question is how much the impact will be. To answer that question is extremely difficult, since these are all interrelated: the schedule may be sufficiently flexible to accept delay, but the delay means additional cost; the budget may be sufficiently flexible to accept new requirements, but the technical complexity of the system adds additional risk that the quality of the end product will suffer, and so forth.
- ***What are the side effects?*** Aside from any impact on budget and schedule, it is also true that making a change anywhere in a complex entity like an acquisition can have a ripple effect outside the program. Possible side effects could include: another system that depends on the present system may have a forced delay in its schedule; introduction of a new oversight agent brings the need to adhere to different security requirements; and so forth.

Evidence of flexibility

Some brief indications that a program exhibits flexibility might be

- When presented with a solution that addresses only some of the user's needs, the user responds: "OK, it's not quite what we wanted, but we'll make do…"
- When presented with a slate of new and unexpected requirements, the contractor says: "OK, we'll manage to bang these in without too much trouble and with minimal additional cost…"

- When given some set of new priorities by DoD management, the program manager says: "OK; I didn't expect this, but we'll manage to retask some money…"

Anticipating the Need for Flexibility

Even if the actual stimulus for flexibility is unknown in advance, there may still be an anticipation that *some* kind of flexibility will be required *somewhere* down the line (e.g., a program may be just getting underway during a highly unstable budgetary environment such as the recent sequester/shutdown mode of funding). By contrast, the need for flexibility may arise purely unexpectedly (e.g., a sudden new condition in a warfighting context may impose significant requirements turbulence, such as the rapid growth in the number of improvised explosive devices in Iraq and Afghanistan).

A concrete flexibility scenario would be as follows: Contractor personnel have been staffing most of the program functions. The program office receives a DoD-wide mandate to replace those by personnel from an 8A/Veteran-owned company; the existing contract with the current contractor is terminated cost effectively and the current contractor personnel transfer to the 8A company, which wins the replacement contract. The personnel changes occur within four months.

Tactics for Flexibility

Tactics to allow flexibility have the objective of ensuring the possibility of actually making the requisite changes, as well as controlling the time and cost to make them. There are a minimum of five areas in an acquisition program where the need for flexibility is most apparent. These are

Tactic Category	Possible Tactics
Contract	Appropriateness of contract structure Potential for unanticipated terminations Defining qualities as requirements Specifying bidding constraints
Schedule	Build flexibility in Support of key goals
Performance	Avoiding certain contract types Dynamic requirements Use of iteration Appropriate fallbacks
Relationships with Contractors	Clarifying expectations and responsibilities Parallel prototypes
Relationships with Users	Keep in touch with users

Note that these categories are not entirely discrete, and that flexibility tactics often tend to fall into two categories simultaneously. Each of these possible tactics is further discussed below.

Flexibility Tactics for Contracts

- *Use an appropriate contract structure* that not merely is defined for the best pricing structure, but also anticipates such things as unexpected changes in user demand. As an example, if for a given program it is known that there are only a few capable contractors, a contract structure that permits awarding indefinite delivery/indefinite quantity (IDIQ) contracts to multiple ven-

dors might provide the needed flexibility when such a sudden surge in needed capability appears.

- *Build potential for unanticipated terminations and restarts into the contract*. It can never be predicted when events might demand that an existing contract be terminated. However, the mechanism for cost-effective termination can certainly be built into a contract as a precaution.
- *Define commonly needed qualities (e.g., portability, interoperability) as requirements*. This anticipates such unforeseen events as directives to move government-provided platforms, or to achieve additional interoperation with some other system. Also define clearly how these qualities will be tested, evaluated, and verified.
- *Clearly specify all bidding constraints*, using language in the RFP to anticipate and prevent surprises during source selection. This can mitigate, for example, the sudden appearance of other government organizations that claim some authority for parts of the acquisition (e.g., a government agency that claims that it should perform the system integration tasks).

Flexibility Tactics for Schedule

- *Incorporate schedule flexibility* by avoiding specific dates. Instead, use date ranges; define delivery dates as "X months after previous delivery," etc.
- *Focus on activities that support the key goals* since these should take priority. Thus, let the schedule be prioritized around whatever activities are most needed for those key program goals. For example, if the goal is to develop a product line, ensure that the schedule clearly supports the relevant activities that most support that goal.

Flexibility Tactics for Performance

- *Handle requirements change properly*. As an example, understand when you should avoid forcing the contractor to bid to the requirements. This situation can occur when it is known, just before release of the RFP, that the requirements will soon undergo significant revision. In such a circumstance, avoid a completion or firm-fixed-price contract; instead, emphasize the contractor processes for managing requirements and responding to change (e.g., more like a services contract that procures program management, engineering, and development services rather than the finished product).
- *Plan in advance for dynamic requirements as the mission changes.* Do this by levying firm constraints on which requirements are accepted and which release will satisfy them and establish a single, empowered authority that can accept or reject proposed requirements
- *Encourage iterative development*, and actively monitor and participate in relevant market segments. Additionally, there must be an explicit willingness to revise architecture as needed in midstream.
- *Use a strategy with appropriate fallbacks*. For instance, a program may have a product line as the goal, but it could become apparent that it cannot be achieved at present. In such an event, have a strategy that accepts that even if two variants cannot become a product line now, there is a fallback strategy where they can begin sharing later (e.g., by sharing something [platform, architecture, components] now, and planning for flexibility to accommodate cost and schedule impacts and define how developers would be incentivized to follow a product line ap-

proach). In other words, the goal of achieving a product line must be flexible (i.e., build flexibility of the goal by explicitly defining those characteristics for which you are willing to trade for other characteristics, etc.).

Flexibility Tactics for Contractor Relationships

- *Clarify all expectations, responsibilities, and obligations for contractors*. Be sure that the RFP (and the subsequent contract) explains in detail everything that the contractor is expected to do, including all rights and responsibilities. Also define whatever mediation mechanisms and penalties are in place.
- *Use parallel prototypes*. Keep the maximum number of independent and innovative contractor solutions in play until the program can understand their feasibility. An additional factor is that it must be easy to drop a contractor, so an additional tactic is that the contract has to make this clear and easy.

Flexibility Tactics for User Relationships

- *Keep in touch with users*. Maintain a continuing relationship with the stakeholders to allow for adjustments to performance requirements.

Executability

At the very foundation of any acquisition exists a critical question, one that is too often left unasked for too long. When someone proposes a new system, or a grand vision of some novel capability, or some groundbreaking advance in technology, someone else should loudly raise the question: "Do we realistically think that this is possible?"[19] Ideally, that would provoke a lengthy argument that touched on such issues as practicality, realism, the current state of political favor, the current state of technology, availability of resources, and other similar topics, all of which would shed light on the issue of whether the proposed acquisition—and, in particular, the strategy that will guide that acquisition—are really executable.

Executability Defined

Though the term is not particularly ambiguous, "executability" is not a common word. Most definitions of "executability" tend to be circular, as for instance, the following:

> *Executability: capability of being executed (e.g., "the job is executable for two million dollars.")*[20]

Somewhat more helpful is a related definition of "executable," as, for instance

> *Executable: Capable of being produced in accordance with a plan or design (e.g., "the plan is executable, though only if all of the preconditions are met.")*[21]

---

19 Informally, we are dealing in this section with the question of "*Can* the system be built?" Note that this is quite different from "*Should* the system be built?"

20 www.thefreedictionary.com/executability

And finally, in the specific context of acquisition strategy, the quality attribute "executability" connotes one or both of the following:

- Whether or not the acquisition strategy is "performable" (i.e., is it possible to carry out the strategy). In other words, a strategy that is executable evidences practicality and realism: the system can be technically built, and the scope of what is planned is commensurate for some combination of resources: cost, schedule, and people. This in turn presupposes that there are capabilities (e.g., technology, skills, tools) existing in the industrial base to build the system. Using a musical metaphor, "executability" would mean that for some piece of music, it is possible to play the piece, given performers of sufficient skill.
- Whether or not the resources available to the program are sufficient to perform the strategy (i.e., right budget, right schedule, right staff). A significant contributor to this sense of "executability" is related to political wherewithal: the powers that be want the system built and have allocated (or will allocate) appropriate resources. Using another musical metaphor: the people who have the right skills are available and ready to play the piece of music.

In considering acquisition strategy in light of the first of these meanings, we must answer the following questions:

- ***Is the planned strategy practical?*** There are many other ways that a strategy can lack practicality. For instance, the planned strategy may make naïve assumptions about technological breakthroughs. Another example would be a strategy that depended on numerous physical constraints being overcome, yet with little or no empirical evidence that this can in fact occur. A further example might involve the assumption that there will be changes in laws, rules, or regulations. By contrast, an indicator of practicality might be a strategy that takes explicit cognizance of precisely which skillsets were needed for a given project.
- ***Is the planned strategy realistic?*** One example of lack of realism would be an assumption that multiple contractors will freely share proprietary processes or data. The more likely circumstance is that, even with contractual language to that end, contractors will seek to minimize or even sidestep required pooling of information. In particular, making any such assumptions must take into account the other projects that the contractors are involved in, since they may be cooperating on the program at hand, but might be in fierce competition for bidding on upcoming projects. In such a case, a strategy that assumes extensive sharing between contractors is not based on realism about typical contractor behavior.

Evidence of Executability

The second of the above meanings (i.e., resources) is related to the familiar questions of cost, schedule, and personnel. Indications that a program exhibits executability would be found in the answers to the following questions:

---

[21] http://dictionary.reference.com/browse/executable. NB: the above is a slight paraphrase of the web page referenced; the actual definition given there was of the verb "execute" rather than the noun "executability." Further, the example has been changed in the above citation.

- Does the strategy show evidence of "affordability?" In other words, is there sufficient funding available to execute the strategy? Having sufficient funds implies that not only that costs are appropriate to achieve the desired capabilities, but also that they are sufficient to achieve them on schedule.
- Does the strategy show evidence of "schedulability?" In other words, is the schedule appropriate for the executing the program using the planned budget?
- Does the strategy show evidence of "staffability?" In other words, can the program find and retain the right people to execute and manage the program, given the budget and schedule? Staffability is of two kinds:
  - Staffability of the **contractor**: the contractor has allocated the right number of people with the right skills to do the planned work. This has to do with the availability of the right skills in the industrial base, the selection of the contractor who has or can attract those skills, and the incentives applied through the contract to use the right people to do the work (the "A" team).
  - Staffability of the **program office**: the program office has the right number of people with the right skills to oversee the contractor's execution of the planned work. This has to do with the availability of the right skills (or the ability to contract for the right skills) on the program office team. In some cases, it will depend on the ability of the program manager to even recognize the need for those skills.

Tactics for Executability

Tactics to promote executability have the goal of ensuring that the acquisition strategy is truly executable, and that the necessary resources are in place to support that execution. Some of these tactics apply to the manner in which the program is initially defined, and others apply as the program progresses (i.e., static and dynamic). There are a minimum of five areas in an acquisition program where the need for executability is apparent:

Tactic Category	Possible Tactics
Contract	Be predictive Make expectations explicit Prepare for change Reflect the vision
Schedule	Factor in the end game Create a pessimistic schedule
Budget	Create budgetary safeguards Use parallel approaches Consider using government labs
Program Management Office Structure	Create a staffing plan Maintain the staffing plan Define standardized processes Optimize new personnel
Relationships with Contractors	Maintain awareness Pay attention to workforce Liberal incentives Pay attention to subcontractors Monitor contractor collaboration

As was true in the description of flexibility, these categories are not entirely discrete, and executability tactics often tend to fall into one or more categories simultaneously. Each of the possible executability tactics is further described below.

Executability Tactics for Contracts

- *Be as predictive as possible* when writing the contract. For instance, contractors have been known to bid using one staffing profile (e.g., recognized experts) and then replace them soon after contract award. To prepare for this, put restrictions in the contract (e.g., a key personnel clause) to ensure that the team that created the bid is the team that does the work.
- *Make expectations explicit,* for instance with regard to data rights (e.g., by making them part of the selection criteria) and intellectual property (IP) rights (e.g., ensure that there are agreements for the transfer of IP).
- *Prepare for change* as the program gets underway. For instance, if it is likely that the program will have some volatility, explicitly include the possibility of task reallocation in all contracts. As another example, put in place a no-penalty clause that permits some variation of some system features if the cost is prohibitive.
- *Have the contract reflect the vision* for the program. Ensure that the contract requires alignment to the vision and puts in place frequent reviews to ensure that no divergence occurs.

Executability Tactics for Contractor Relationships

- *Maintain awareness of contractor personnel* not only to guarantee that the team that was bid actually performs the work, but also to guard against top-down government interference. For instance, directives occasionally appear that can demand workforce changes (e.g., start using 8/A workers). So plan to track the personnel on the contract in order to be able to demonstrate the skills that will be lost in such an event.
- *Pay attention to every aspect of workforce qualification,* for instance, necessary clearances. Provide an opportunity for competing contractors to increase their pool of cleared people in advance of contract award. It is also possible to require that contractors demonstrate that they have the cleared people necessary on their staff.
- *Be liberal with incentives* by creating performance measures of and incentives for desired behavior. These do not simply have to be focused on development; it is also possible to find ways to incentivize the development contractor to care about long term maintenance costs. It is also possible to build a suitable reward structure as an incentive for the contractor to be prepared to increase the size of the workforce.
- *Pay attention to subcontractors* by enforcing the flow down of critical process maturity requirements from the prime contractor to ensure the subcontractors have the planning skills they need; set aside funds to train/mitigate risks of gaps.
- *Create contract requirements* that allow the government to monitor collaboration between contractors.

Executability Tactics for Structuring the PMO

- *Create a detailed PMO staffing plan.* Above all, have a staffing plan in place that includes specific required skills mapped to program activities and milestones.

- *Maintain the PMO staffing plan* and be sure it shows the skills required to oversee all aspects of the contract. Include a plan to hire or develop the needed skills—with an understanding of the impact of not being fully staffed. And finally, as a precaution, accompany this plan with the impact on the program should the skills be unavailable.
- *Define standardized processes* to handle routine work so that scarce program office resources can be allocated to the most pressing needs.
- *Optimize new PMO personnel* by making them effective as quickly as possible. Have in place an organized way to transmit program knowledge that trades the overhead of documentation for the learning curve.

Executability Tactics for Budget

- *Create budgetary safeguards* that will clarify the specific cause for any increased costs. Require appropriate controls and metrics to manage cost increases, and ensure that the metrics include risk mitigation.
- *Use parallel approaches whenever possible* since these are generally cost savers in the long run. If possible, keep the maximum number of independent and innovative contractor solutions in play until the program team can understand their feasibility.
- *Consider using government labs* to produce the initial product when building an unprecedented capability, or a system with significant technical risk and unknown requirements. If the engineering phases are tightly controlled, a contractor can later be required to design their implementation based on an approved prototype.

Executability Tactics for Schedule

- *Factor the end game* into the initial schedule by considering all of the tasks that will be required to field the system. For example, getting information assurance (IA) certification will inevitably become an issue at some point. Plan for this initially by integrating IA personnel and IA considerations into the development process. Additionally, provide early deliverables to IA personnel.
- *Create a pessimistic schedule* and resist pressure to change it. Schedule slips happen one day at a time. Make **any** slip—even a single day—a major item, such that the program status is labeled “Behind Schedule.” Prevent contractors from hiding **any** schedule delays in the small print at the bottom of slides.

Appendix B: Examples of Conflicting Scenario Pairs

Example 1

In carrying out the analyses in our method, we note that the types of conflict observed will sometimes be fairly obvious, but in other cases they will be more subtle, particularly when comparing an architecture scenario and an acquisition scenario. In the former case, the conflict will be readily apparent. One example of an obvious conflict could be the following:

Program Executive Office (PEO) Goal: wants the program to be open to requests for additional capability in ongoing incremental deployments (related to acquisition quality attribute of *responsiveness*)

Stimulus:	the stakeholders in the field request {some additional capability} in delivery of the third iteration
Environment:	during development of the third iteration of a system
Response:	the program manager determines from the contractor that new capability can be added and deployed
Response measure:	the additional capability is added to the release within {x} days, and the schedule for the release is delayed no longer then {y} days

Elements of analysis for the scenario include

- Primary acquisition trade space value: performance and schedule
- Implication for acquisition strategy: The strategy must clarify that the program will depend on planning that allows new requirements to be inserted in each iteration; the new requirement does not go at the bottom (as in standard Agile development).

Thus, a perfectly reasonable goal leads to a perfectly reasonable scenario. But in the same program, the end users may have voiced the following goal and scenario, also perfectly reasonable:

System Users' Goal: want the system to be extremely robust, and recover from faults rapidly so as to require down time of no more than {x seconds} for a simple crash and {y minutes} for a serious crash (related to software/system quality attribute of *availability*)

Stimulus:	incorrect user-input data crashes the system/network; crash kills the system
Environment:	during operation
Response:	The users fulfil mission operations with no significant degradation of operations caused by system crashes. "Significant degradation of operations" is understood as no external entities that interact with system users perceive unacceptable delays.
Response measure:	the system's recovery to full capability occurs within {x} seconds/within {y} minutes

Elements of analysis for the scenario include

- Primary acquisition trade-space value: performance
- Implication for architecture**:** Fully define all recovery tactics for all versions of the system before finalizing the system architecture.
- Implication for acquisition strategy: It will be necessary to ensure that cost and schedule are sufficient to allow performance to dominate.

Comparing these two scenarios, it is unlikely that any system could support both. If two releases of the system are already deployed, and unexpected capability must be added to the third release, then the scenario of defining all recovery tactics before solidifying the architecture is probably impossible.

Example 2

An example where the conflict is less obvious is illustrated by a program in which a large and complex commercial product was to be acquired for joint use by two different organizations within a single service. The COTS product would first need to be customized, a process which is often long and difficult. The principal expected benefit was that several duplicative processes currently used by both organizations would be eliminated, since they were all present in an integrated manner in the COTS product. Implicit in this plan was the expectation that the two organizations would undergo a large amount of business process modernization to accommodate the integrated processes.

Service Secretary Goal: wants the service to integrate process areas across two large and critical organizations, thus eliminating both wasteful duplication and complex and inefficient workarounds (related to acquisition quality attribute of *flexibility*)

Stimulus:	spiraling cost of both redundant processes and time required to accomplish cross-organizational missions
Environment:	before program inception
Response:	orders that a program be started to implement an integrated system to be shared by both organizations
Response measure:	the system should reach initial operational capability (IOC) within six years

Elements of analysis for the scenario include

- Primary acquisition trade space value: cost
- Implication for acquisition strategy: The acquisition strategy should stress the need for real cooperation across the organizations, and should incorporate incentives for cooperation and penalties for failure to accomplish integrated processes This goal was extremely important, since the budgets for each of the services in this business area was growing at a very large rate.

However, in light of a recent report on the rate of failure of “big bang” acquisitions, the size and expected cost led to the decision that there should be two separate acquisitions, one for each of the main functional areas of the organizations. The strategy was that, since these functional areas

were already present in an integrated manner in the COTS product, each of the two programs could develop their needed capabilities separately, and they could then be merged later into the desired integrated system. The scenario from the program executive officer's (PEO's) perspective was therefore

PEO Goal: wants the program to avoid a "big bang" acquisition since these have often led to failing programs (related to acquisition quality attribute of *program survivability*)

Stimulus:	the PEO is made aware of the rate of failure of "big bang" acquisitions
Environment:	before program inception for a system based on a very large and complex COTS product
Response:	the PEO orders that acquisition of the system be split into two separate programs, each with its own funding stream, and the two programs focused on the individual functional areas of the two sponsoring organizations
Response measure:	none of the bad results that plague "big bang" acquisitions occur in this program

Elements of analysis for the scenario include

- Primary acquisition trade-space value: cost
- Implication for acquisition strategy: There will now need to be two separate acquisition strategies, and each will focus on different particulars, account for the different funding streams and different development organizations. Each of the separate acquisition strategies must put in an explicit description and plan for how to link the two separate systems so that they get the benefits of the original product.

The result was that by splitting the program (and, by implication, the large and complex COTS product) into two parts, the door was opened to several unhappy conflicts between the two programs. First and foremost, the goal of the service secretary was never embodied in any realistic authority strictures (i.e., the incentives and penalties for cooperation across organizations); neither program devised an appropriate acquisition strategy. The integrated processes that consisted of cross-organizational flows were never created nor implemented. One of the two programs suffered a long and costly protest, thus separating the scheduled IOC of each part. And finally, the difficulty of reintegrating the complex COTS product had been severely underestimated, and was never begun.

Example 3

It is possible for a conflict to arise when one element (e.g., the acquisition strategy) undergoes significant change, but the original conception of the architectures is still in place. In one case, a program had begun with a very large research basis and an extended deployment strategy, but upon getting a new PEO, the program shifted to a focus of rapid deployment of capability. The acquisition strategy thus underwent a radical change, but did so without consideration of the architecture, particularly of the time required for it to mature.

(New acquisition scenario) PEO Goal: wants the program to deploy capability to the field as quickly as possible (related to an acquisition quality attribute of *responsiveness*)

Stimulus:	the PEO demands releases of capability as quickly as possible
Environment:	during early development, where a program is developing a considerable amount of unprecedented technology
Response:	the program speeds up development and makes an immediate release
Response measure:	system begins to be deployed to the field within two years

Elements of analysis for the scenario include

- Primary acquisition trade space value: schedule
- Implication for acquisition strategy: The strategy must emphasize a development model that permits sudden change of focus from research to development

The conflict results from the fact that though the acquisition strategy is now radically different, the architectures are still based on the original, research-based concepts that implied a lengthy time period for technology maturation. Thus, the key scenario that had originally governed the architectures was

(Old architecture scenario) Individual Service Chiefs' Goal: want the system to have an architecture that supports all service-specific variants, but also permits interoperation between and among all variants (related to a software/system quality attribute of *usability*)

Stimulus:	individual service requirements are incrementally clarified and matured
Environment:	during early development of an unprecedented architecture (both system and software)
Response:	the architecture is developed incrementally, and total architecture will be fully defined when all requirements are known to be recorded
Response measure:	the separate components built using these technologies operate in the desired manner in both standalone mode and in interoperation

Elements of analysis for the scenario include

- Primary acquisition trade-space value: performance
- Implication for acquisition strategy: The acquisition strategy must specify a careful, methodical, step-by-step approach for maturing a cross-system architecture and the unprecedented technology, and must define how cross-service cooperation will be effected. The consequence if the response (i.e., the mitigation of the risk) does not occur is that the program will fail, which is precisely what occurred.

The acquisition trade space values of the two scenarios are completely at odds. The change of approach to immediate deployment occurred at a time when the architecture was barely defined, and the attempts to quickly develop applications ended in failure. The time needed to finalize a viable architecture was far greater than the aggressive new schedule that was ordered. The temporal aspect of response shown above in the new acquisition scenario—the program speeds up development and makes an immediate release—was fundamentally in conflict with the temporal aspect of the state of developing the unprecedented architecture.

Appendix C: Materials Used in Alignment Method Execution

This appendix shows several templates that are useful in various phases of the alignment method. These include

- Master List of Program Stakeholders
- Business/Mission Goal Template
- Goal-Attribute Matrix
- Candidate Scenarios Template
- Refined Scenario Template

Master List of Program Stakeholders

This template is used in the following phase of the alignment method:

- Phase 1: Identify stakeholders and elicit business goals

This template is used to record all potential stakeholders, their contact information, and when they were interviewed. This master list is helpful when planning the method events such as in Phase 1 as well as later as when the program is attempting to resolve potential goal or scenario conflicts. "Representative Stakeholder Category" refers to the abstract list of stakeholder categories shown in Table 3 in Section 3.1.3.

Stakeholder Name	**Stakeholder Contact Details**		**Representative Stakeholder Category**	**Interview Date**
	Organization			
	Position			
	Rank			
	Phone			
	Email			
	Organization			
	Position			
	Rank			
	Phone			
	Email			
	Organization			
	Position			
	Rank			
	Phone			
	Email			
	Organization			
	Position			
	Rank			
	Phone			
	Email			

Business/Mission Goals Template

This template is used in the following phase of the alignment method:

- Phase 1: Identify stakeholders and elicit business goals

Goal ID	Goal Title	Goal-Subject	Goal-Object	Goal

Goal ID is a unique identifier for each business or mission goal generated and used as part of the alignment method. Simple alphanumerical values such as G1, G2, etc. are usually sufficient.

Goal Title is an optional field but many find it a helpful reference during discussions and analysis. A short three- to five-word phrase that captures the intent of the goal tends to work best.

Goal-Subject is the stakeholder who owns the goal, who wishes that it be met.

Goal-Object is the entity to which the goal applies.

Goal is the goal itself; this may either be relatively general ("meet financial objectives") or may be quite specific ("field system in less than six months").

Goal-Attribute Matrix (Adapted from PALM)

This template is used in the following phases of the alignment method:

- Phase 1: Identify stakeholders and elicit business goals
- Phase 2: Elicit quality attributes scenarios
- Phase 3: Analyze scenarios for misalignments

This template works well in a spreadsheet tool. It can either be displayed and used directly during the workshop to record goals or used solely to record what is recorded on flip charts as the participants brainstorm their goals.

Goal ID	Goal Statement	AQA - <QA label>	AQA - <QA label>	…	S/S QA - <QA label>	S/S QA - <QA label>	…

Goal ID is a unique identifier for each business or mission goal generated and used as part of the alignment method. The Goal ID should correspond to the ID from the Business/Mission Goal Template.

Goal Statement is a sentence form of the goal sub-element such as "For the system being developed, <goal-subject> desires that <goal-object> benefit from <goal>."

AQA <QA label> is one column for each acquisition quality attribute identified (e.g., executability). Collocating all acquisition quality attributes columns will streamline various analysis activities.

S/S QA <QA label> is one column for each software or system quality attribute identified (e.g., performance, maintainability). Collocating all software and system quality attributes columns will streamline various analysis activities.

Candidate Scenarios Template

This template is used in the following phase of the alignment method:

- Phase 2: Elicit quality attributes scenarios

This template works well in a spreadsheet tool. It can either be displayed and used directly during the workshop to record scenarios or used solely to record what is recorded on flip charts as the participants brainstorm their scenarios.

Note that often a response measure is not known at this point in forming an acquisition strategy or architecture. If such information is known and is central to the scenario, it is good to capture it.

Scenario ID	Scenario Title	Stimulus	Environment	Response/Response Measure

Scenario ID is a unique identifier for each acquisition and software/system quality attribute scenario generated and used as part of the alignment method. Simple alphanumerical values such as S1, S2, etc. are usually sufficient.

Scenario Title is an optional field but many find it a helpful reference during discussions and analysis. A short three- to five-word phrase that captures the intent of the scenario tends to work best.

Stimulus is the condition that affects the system (e.g., initiating or triggering event).

Environment is the condition under which the stimulus occurred (e.g., preconditions).

Response is the activity that results from the stimulus (e.g., desired system action or capability) and if known, **Response Measure** is the measure by which the system's response will be evaluated.

Refined Scenario Template

This template is used in the following phase of the alignment method:

- Phase 2: Elicit quality attributes scenarios (step 6, scenario refinement)

Scenario ID		
Scenario	Cut and paste the scenario here.	
Business/Mission Goals	List those goals that this scenario affects.	
Quality Attributes	Quality attributes (acquisition, system, software) addressed	
Scenario Refinement	**Stimulus**	Condition affecting the system/artifact
	Stimulus Source	Entity generating stimulus
	Environment	Conditions under which the stimulus occurred
	Artifact	Software/system QA: Part of system stimulated (can be whole system) Acquisition QA: Part of program stimulated
	Response	Activity undertaken after arrival of stimulus
	Response Measure	Measure of the activity taken after the arrival of the stimulus
Acquisition Strategy Decisions and Reasoning	List the acquisition strategy decisions relevant to this scenario that affect the quality attribute response and briefly explain the qualitative and/or quantitative rationale for why the acquisition strategy decisions contribute to achieving the quality attribute response requirement.	
Architectural Decisions and Reasoning	List the architectural decisions relevant to this scenario that affect the quality attribute response and briefly explain the qualitative and/or quantitative rationale for why the architectural decisions contribute to achieving the quality attribute response requirement.	
Risks	List any discovered risks. Note what decision was made (or not made) to cause the risk, the context in which it occurs, and its consequence. Note if it is system, software or both.	
Sensitivities	List any discovered sensitivities.	
Tradeoffs	List any discovered tradeoffs.	
Other Issues	List any other discovered issues.	

References

URLs are valid as of the publication date of this document.

[Barbacci 2003]
Barbacci, Mario, Ellison, Robert, Lattanze, Anthony, Stafford, Judith, Weinstock, Charles, & Wood, William. *Quality Attribute Workshops* (QAWs), *Third Edition* (CMU/SEI-2003-TR-016). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6687

[Bass 2012]
Bass, Len, Clements, Paul, & Kazman, Rick. *Software Architecture in Practice*, 3rd edition. Addison-Wesley, 2012. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=30264

[Brown 1998]
Brown, W.J. *Antipatterns: Refactoring Software, Architecture, and Projects in Crisis*. Wiley and Sons, 1998.

[Brownsword 2013a]
Brownsword, Lisa, Albert, Cecilia, Carney, David, Place, Patrick, Hammons, Charles (Bud), & Hudak, John. *Isolating Patterns of Failure in Department of Defense Acquisition* (CMU/SEI-2013-TN-014). Software Engineering Institute, Carnegie Mellon University, 2013. http://www.sei.cmu.edu/library/abstracts/reports/13tn014.cfm

[Brownsword 2013b]
Brownsword, Lisa; Albert, Cecilia; Carney, David; & Place, Patrick. *Results in Relating Quality Attributes to Acquisition Strategies* (CMU/SEI-2013-TN-026). Software Engineering Institute, Carnegie Mellon University, 2013. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=78312

[Carney 2003]
Carney, David; Morris, Edwin; & Place, Patrick. *Identifying Commercial Off-the-Shelf (COTS) Product Risks: The COTS Usage Risk Evaluation* (CMU/SEI-2003-TR-023). Software Engineering Institute, Carnegie Mellon University, 2003. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=6767

[Clements 2010]
Clements, Paul & Bass, Len. *Relating Business Goals to Architecturally Significant Requirements for Software Systems* (CMU/SEI-2010-TN-018). Software Engineering Institute, Carnegie Mellon University, 2010. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9347

[DAU 2011]
Defense Acquisition University. *Glossary of Defense Acquisition Acronyms and Terms*. 14th Edition, 2011.

[GAO 2011]
United States Government Accountability Office. Report to Congressional Committees: Defense Acquisitions; Assessments of Selected Weapon Programs (GAO-11-233SP). March 2011.

[Williams 1999]
Williams, Ray; Behrens, Sandra; & Pandelios, George. *SRE Method Description (Version 2.0) & SRE Team Members Notebook (Version 2.0)* (CMU/SEI-99-TR-029). Software Engineering Institute, Carnegie Mellon University, 1999. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13557

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2014	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
A Method for Aligning Acquisition Strategies and Software Architectures	FA8721-05-C-0003

6. AUTHOR(S)

Lisa Brownsword

Cecilia Albert

David Carney

Patrick Place

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2014-TN-019

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFLCMC/PZE/Hanscom Enterprise Acquisition Division 20 Schilling Circle Building 1305 Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

In the acquisition of a software-intensive system, the relationship between the software architecture and the acquisition strategy is typically not carefully examined. To remedy this lack, a research team at the Carnegie Mellon University Software Engineering Institute (SEI) has focused a multiyear effort to discover an initial set of failure patterns that result when these entities become misaligned and identify a set of desired relationships among the business and mission goals, system and software architectures, and the acquisition strategy. This report describes the result of the third year of the SEI's research, where the team defined a method that indicates such areas of misalignment (i.e., between a program's architecture and acquisition strategy). The alignment method is used as early in a program's lifetime as practical, ideally before the architecture or acquisition strategy has attained full definition. The authors illustrate the method by means of a case study, during which many of the key elements of the method were piloted.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Acquisition, software architecture, quality attributes, business goals	64

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL