MP110369

MITRE PRODUCT

Patterns of Success in Systems Engineering
Acquisition of IT-Intensive Government Systems

Sponsor: MITRE
Dept. No.: A225/E500
Contract No.: FA8721-11-C-0001
Project No.: 01AAV700-M1

Bedford, MA

George Rebovich, Jr.
Joseph K. DeRosa

November 2011

Abstract

The objective of this effort was to discover patterns of success in the systems engineering of information technology (IT)-intensive systems in a government acquisition environment using the method of positive deviance.

Thirty government programs were identified, each with some notable success in the acquisition of IT-intensive capabilities. Twelve were selected for extensive follow-up and analysis, including detailed interviews with front-line practitioners who cope with the demands of the government acquisition system and are in a position to influence or observe positive deviance in their environment.

This report describes two large-scale success patterns that were observed, each with several recurring sub-patterns. “Balancing the Supply Web” addresses “social” interdependencies among enterprise stakeholders who have different equities in the capability being developed. “Harnessing Technical Complexity” addresses the technical interdependencies among system components that together deliver an operational capability for the enterprise. The large-scale patterns are two sides of the same coin. The programs studied achieved success because of the way they each navigated through these dual interdependencies.

Keywords: acquisition, complex systems, design patterns, information technology, positive deviance, systems engineering

This page intentionally left blank.

Executive Summary

The objective of this study is to discover and document patterns of success in the systems engineering of IT-intensive systems within the government acquisition environment. Rather than focus on failures and their remedies, the effort focuses on successful practices and their patterns. It favors practice over theory. Results come from practitioners, not advisors.

The results indicated two large-scale patterns of success. Like two sides of the same coin, one represents social aspects of successful systems engineering, the other technical. The study also uncovered fifteen sub-patterns of success, roughly organized along social and technical lines.

Aspect	Social	Technical
Large-scale Patterns	***Balancing the Supply Web***	***Harnessing Technical Complexity***
Small-scale Patterns	• ***Up Close and Personal*** • ***Close, But Not Too Close*** • ***Divide and Conquer*** • ***Circle of Trust*** • ***Role and Responsibility Subnets*** • ***Seek Secondary Sources*** • ***Network Beats the Node*** • ***Top Cover***	• ***Seeing Is Believing*** • ***Riding on the Infrastructure*** • ***Loose Couplers*** • ***Social and Technical Alignment*** • ***Plan to Re-plan*** • ***Technology Surfing*** • ***Architect.org***

The implications are far reaching. The systems engineering and acquisition process in this environment has been broken and resistant to change for many years now. Numerous attempts at reform have failed to take hold. True acquisition reform can come about by searching out and applying the wisdom that already exists inside the systems engineering community engaged in government acquisitions. Organizing successful practices along social and technical lines in two large-scale patterns—like two sides of a coin—makes remembering and communicating them simple. A handful of sub-patterns, roughly organized along the same lines, capture many accumulated years of successful experience. These patterns succeed where so many others have failed. It remains now to bring these results back to the community from which they were uncovered.

Acknowledgments

This work was jointly sponsored by MITRE’s Office of the Corporate Chief Engineer and by the General Manager of the Command and Control Center under the auspices of the Advanced Systems Engineering Center. We thank Dr. Louis Metzger, corporate chief engineer, and Mr. David Lehman, senior vice president and chief operations officer and former general manager of the Command and Control Center, for their support and encouragement for this project. We especially thank the anonymous but participating MITRE project leaders and systems engineers for cogent and candid assessments of their programs.

Table of Contents

1 Introduction ... 1
1.1 The Current Situation ... 1
1.2 Fundamentals of Positive Deviance ... 3
1.2.1 Why Positive Deviance Works ... 3
1.2.2 Examples of Positive Deviance ... 3
1.3 Fundamentals of Patterns ... 4
1.3.1 Definition of Patterns ... 5
1.3.2 Examples of Patterns ... 6
1.3.3 Template for Patterns ... 6
2 Identifying Patterns of Positive Deviance ... 8
2.1 Defining Success ... 9
2.2 Identifying Successes ... 9
2.3 Interviewing the Teams ... 10
2.4 Capturing Their Experience ... 12
2.5 Communicating the Results ... 12
3 Large-Scale Enterprise Patterns ... 14
3.1 Balancing the Supply Web ... 14
3.1.1 Context of Balancing the Supply Web Pattern ... 14
3.1.2 Problem ... 16
3.1.3 Forces at Work ... 17
3.1.4 Solution ... 18
3.1.5 Examples ... 18
3.1.6 Outcome ... 19
3.2 Harnessing Technical Complexity ... 19
3.2.1 Context of Harnessing Technical Complexity Pattern ... 19
3.2.2 Problem ... 20
3.2.3 Forces at Work ... 21
3.2.4 Solution ... 21
3.2.5 Examples ... 21
3.2.6 Outcome ... 22
4 Supply Web Patterns ... 23
4.1 Up Close and Personal ... 24
4.1.1 Context of Up Close and Personal Pattern ... 24

4.1.2 Problem 25
4.1.3 Forces at Work 25
4.1.4 Solution 26
4.1.5 Examples 26
4.1.6 Outcome 27
4.2 Close, But Not Too Close 27
4.2.1 Context of Close, But Not Too Close Pattern 28
4.2.2 Problem 28
4.2.3 Forces at Work 28
4.2.4 Solution 29
4.2.5 Examples 30
4.2.6 Outcome 31
4.3 Divide and Conquer 31
4.3.1 Context of Divide and Conquer Pattern 32
4.3.2 Problem 33
4.3.3 Forces at Work 33
4.3.4 Solution 33
4.3.5 Examples 35
4.3.6 Outcome 36
4.4 Circle of Trust 36
4.4.1 Context of Circle of Trust Pattern 36
4.4.2 Problem 37
4.4.3 Forces at Work 37
4.4.4 Solution 37
4.4.5 Examples 38
4.4.6 Outcome 39
4.5 Role and Responsibility Subnets 39
4.5.1 Context of Role and Responsibility Subnets Pattern 39
4.5.2 Problem 39
4.5.3 Forces at Work 40
4.5.4 Solution 40
4.5.5 Examples 41
4.5.6 Outcome 42
4.6 Seek Secondary Sources 42
4.6.1 Context of Seek Secondary Sources Pattern 42

4.6.2 Problem 44
4.6.3 Forces at Work 44
4.6.4 Solution 44
4.6.5 Examples 44
4.6.6 Outcome 46
4.7 Network Beats the Node 46
4.7.1 Context of Network Beats the Node Pattern 46
4.7.2 Problem 46
4.7.3 Forces at Work 46
4.7.4 Solution 47
4.7.5 Examples 48
4.7.6 Outcome 48
4.8 Top Cover 48
4.8.1 Context of Top Cover Pattern 48
4.8.2 Problem 48
4.8.3 Forces at Work 49
4.8.4 Solution 50
4.8.5 Examples 51
4.8.6 Outcome 51
5 Harnessing Technical Complexity Sub-Patterns 52
5.1 Seeing Is Believing 53
5.1.1 Context of Seeing Is Believing Pattern 53
5.1.2 Problem 53
5.1.3 Forces at Work 54
5.1.4 Solution 55
5.1.5 Examples 55
5.1.6 Outcome 55
5.2 Riding on the Infrastructure 56
5.2.1 Context of Riding on the Infrastructure Pattern 56
5.2.2 Problem 56
5.2.3 Forces at Work 56
5.2.4 Solution 57
5.2.5 Examples 58
5.2.6 Outcome 58
5.3 Loose Couplers 59

5.3.1 Context of Loose Couplers Pattern 59
5.3.2 Problem 59
5.3.3 Forces at Work 61
5.3.4 Solution 61
5.3.5 Examples 61
5.3.6 Outcome 61
5.4 Social and Technical Alignment 61
5.4.1 Context of Social and Technical Alignment Pattern 62
5.4.2 Problem 62
5.4.3 Forces at Work 62
5.4.4 Solution 63
5.4.5 Examples 64
5.4.6 Outcome 64
5.5 Plan to Re-plan 64
5.5.1 Context of Plan to Re-plan Pattern 65
5.5.2 Problem 65
5.5.3 Forces at Work 65
5.5.4 Solution 66
5.5.5 Examples 67
5.5.6 Outcome 68
5.6 Technology Surfing 68
5.6.1 Context of Technology Surfing Pattern 68
5.6.2 Problem 68
5.6.3 Forces at Work 68
5.6.4 Solution 69
5.6.5 Examples 69
5.6.6 Outcome 71
5.7 Architect.org 71
5.7.1 Context of Architect.org Pattern 72
5.7.2 Problem 72
5.7.3 Forces at Work 72
5.7.4 Solution 73
5.7.5 Examples 74
5.7.6 Outcome 74
6 Summary 75

Appendix A Study Methodology 76
Appendix B Case Overviews 78

List of Figures

Figure 2-1. SE Profiles of the 12 Programs (2 per chart) 11
Figure 3-1. Balancing the Supply Web Pattern 15
Figure 3-2. Harnessing Technical Complexity Pattern 19
Figure 4-1. Up Close and Personal Pattern 24
Figure 4-2. Close, But Not Too Close Pattern 28
Figure 4-3. Divide and Conquer Pattern 31
Figure 4-4. Circle of Trust Pattern 37
Figure 4-5. Role and Responsibility Subnets Pattern 40
Figure 4-6 (a) Seeking Secondary Sources 43
Figure 4-6 (b) Operating With and Without Secondary Sources 43
Figure 4-8. Top Cover Pattern 49
Figure 5.1. Seeing Is Believing Pattern 53
Figure 5-3. Loose Couplers Pattern 59
Figure 5-4. Social and Technical Alignment Pattern 62
Figure 5-6. Technology Surfing Pattern 70
Figure 5-7. Architect.org Pattern 72
Figure A-1. Flowchart of Methodology 76

List of Tables

Table 1-1. Template of Design Patterns 7
Table 3-1. Balancing the Supply Web Pattern 16
Table 3-2. Harnessing Technical Complexity Pattern 20
Table 4-1. Up Close and Personal Pattern 25
Table 4-2. Close, But Not Too Close Pattern 29
Table 4-3. Divide and Conquer Pattern 32
Table 4-4. Circle of Trust Pattern 38
Table 4-5. Role and Responsibility Subnets Pattern 41
Table 4-6. Seek Secondary Sources Pattern 45
Table 4-7. Network Beats the Node Pattern 47
Table 4-8. Top Cover Pattern 50
Table 5-1. Seeing Is Believing Pattern 54
Table 5-2. Riding on the Infrastructure Pattern 57
Table 5-3. Loose Couplers Pattern 60
Table 5-4. Social and Technical Alignment Pattern 63
Table 5-5. Plan to Re-plan Pattern 66
Table 5-6. Technology Surfing Pattern 71
Table 5-7. Architect.org Pattern 73

This page intentionally left blank.

1 Introduction

Every Sector Has Problems

The federal government is struggling to modernize its IT-intensive systems. Problems with the cost, schedule, and performance of acquired systems plague the Department of Defense (DoD), the Federal Aviation Administration (FAA), and the Internal Revenue Service (IRS), to name a few. Indeed no sector of government seems to be immune. Clearly it is a difficult task to engineer and acquire a new system or system component that must fit into the larger, complex information enterprise of any sector of government. The new complexity that the government faces in its systems and enterprises is a consequence of the interdependencies that arise when many systems are networked to achieve a collaborative advantage. When networked systems are individually adapting to rapid technology and mission changes, the environment becomes unpredictable. Systems engineering success depends on the ability to adapt not only the individual systems, but the network of constantly changing systems. It is not surprising that poor performance would be the norm. This report follows a methodology that focuses on patterns of practice that successfully deviate from that norm.

1.1 The Current Situation

Many Organizations and Studies Have Tried

Although a considerable storehouse of knowledge exists regarding how to perform systems engineering (SE) and acquisition, problems persist. Professional organizations like INCOSE and IEEE have defined standard SE processes, for example, requirements and test. The government as well as these same professional organizations has developed standard models for the acquisition of systems, for example, waterfall and evolutionary acquisition.[1], [2] Innumerable studies have been conducted on how to improve both the SE and acquisition processes.

Failures Persist

Despite all of these studies, the government has experienced many failures.[3] The failures range from cancellation of the program or non-delivery of the system to the system's arriving too late or not adequately performing. Over the years, several acquisition reform efforts have been aimed at fixing the causes of these failures, and yet they still seem to plague the SE and acquisition communities.

The Promise of IT Is Bogged Down

SE and acquisition of IT-intensive government systems started out with the promise of delivering markedly improved capabilities at substantially reduced costs. There have indeed been successes. The government, like the private sector, has become an information enterprise. The question arises as to why delivering significant new capability through the information enterprise is so slow and expensive while the technology advances so quickly and is so cheap. Why are SE and acquisition "bogged down"? People are quick to point out valid reasons for the current situation:

[1] DoD, The Defense Acquisition Systems, Directive 5000.01, Certified as Current as of Nov. 29, 2007, http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf (Viewed Sept. 2010).
[2] DoD, Operation of the Defense Acquisition System, Instruction 5000.02, Dec. 8, 2008, http://www.dtic.mil/whs/directives/corres/pdf/500002p.pdf , (viewed Sept. 2010).
[3] B. E. White, *Complex Adaptive Systems*, IEEE International Systems Conference, Montreal, Canada, Apr. 2008.

- Excessive government regulation
- The interconnectedness and interdependency of the systems
- The fast-changing nature of government needs (e.g., military threats, changes in IRS tax code, and health and human services laws)
- The rising expectations of the IT-literate populace
- Uncertainty in economic conditions supporting the programs

The list could go on, and for each item there has likely been a study and a recommended remedy. Sometimes these remedies work for a while. Sometimes they work in an isolated case. Sometimes they don't work at all, but rather simply add to the burden the bureaucracy levies on the government program office.

Improvements that "Stick"

This study attempts to highlight improvements that work and will endure. It makes no attempt to change the context in which government acquisition of IT-intensive systems takes place, but rather focuses on improving activities so that they may be done better in the current environment. It uses two proven methods to accomplish this:

- Positive Deviance
- Patterns

Positive Deviance: A Search for What Works

When faced with the question of how to improve the practice of systems engineering and acquisition within a government department or agency, a common approach is to put a spotlight on what has gone wrong in recent, high-profile programs and to posit specific ways to fix the problem. Often the proposed fixes are variations or adaptations of approaches that have worked well in environments different from that of the department or agency, such as commercial business sector practices or those that come from government "skunk works."

The difficulty in transferring approaches that work in one environment to another is in inferring exactly which patterns of an approach must be copied intact, which should be modified, and which may be ignored, to duplicate success in the new environment. In complex social systems many results—both positive and negative—derive from interactions among multiple patterns that are not always fully understood. Attribution of credit in complex systems is fraught with difficulties, and this is why solution approaches transplanted from other environments often fail.

Positive deviance is an approach to improvement based on the idea that every community performing an activity has certain individuals or teams whose attitudes, practices, strategies, or behaviors enable them to function more effectively than others with the same resources and environmental conditions. It is a search for what works. Because positive deviants are embedded in the same environment as the rest of the community, the problems associated with attribution of credit are less severe than when transferring solution approaches across environments. Because many communities are reluctant to change fundamental beliefs based on outsider say-so, positive deviant ideas are more likely to be accepted by their community.

Rather than focus on fixing failures by instituting more regulation and control on the government program offices from outside those offices, positive deviance focuses on successes achieved from inside those offices. It makes no attempt to change government regulations or SE standards.

It does attempt to isolate the strategies and behaviors of those practicing systems engineers and managers who seem to function more effectively than their peers—inside the existing government acquisition environment. It then looks for ways to explain and amplify success.

Positive Deviance Communicates Through Patterns
A key element to amplifying any success is effectively communicating its essence to other practitioners. This report boils down successful learning into a handful of patterns that are easy to both remember and apply. That is, it seeks to improve SE through learning and adapting. Positive deviance and patterns are discussed in this section as generalized tools. Their specific application to the acquisition of IT-intensive systems is discussed in Section 2.

1.2 Fundamentals of Positive Deviance

It Started in 1990
The positive deviance approach was pioneered by Jerry and Monique Sternin around 1990 in fighting malnutrition of Vietnamese children.[4] It was soon after applied to management problems by Seidman and McCauley.[5] It was popularized in 2007 by Gawande in his bestselling book, *Better*.[6]

1.2.1 Why Positive Deviance Works

Social as well as Scientific Dimensions
The core tenet of positive deviance is that for problems that require the cooperative behavior of large groups of people, the top-down directed reforms do not generally work. They don't excite the people carrying out the day-to-day work and they don't end up changing the culture. Gawande states, "The social dimension turns out to be as essential as the scientific."

It Mines Internal Wisdom
What is frequently observed is that restricted attitudes, beliefs, and behaviors don't allow individuals in a community to solve seemingly intractable problems, even though they may have the necessary knowledge to do so. However, when solutions do emerge from within a community, they have the potential to transform its culture. Positive deviance helps participants discover what has been working in their environment and then disseminate that wisdom throughout the community. Rather than imposing solutions from outside experts, positive deviance discovers and amplifies internal solutions.

1.2.2 Examples of Positive Deviance

This approach has been successful in several instances:

1. *Save the Children Vietnamese Anti-Starvation Program*

   In the impoverished conditions in Vietnam, the Sternins found a few families who succeeded in raising healthy children. When asked what they did, the response was simple things: washing hands more often, cooking food differently, and consuming crops that were not

[4] Richard Pascale, Jerry Sternin, and Monique Sternin, *The Power of Positive Deviance: How Unlikely Innovators Solve the World's Toughest Problems*, Boston: Harvard Business Press, 2010.
[5] William Seidman and Michael McCauley, Harvesting the experts' "secret sauce" to close the performance gap, *Performance Improvement Journal*, Jan. 2003, vol. 42, no. 1, pp. 32–39.
[6] A. Gawande, *Better, A Surgeon's Notes on Performance*, New York: Henry Holt, 2007.

relished by the community. When these behaviors were advertised and measured, they became the basis for long-standing change in the community.

2. *Combating Hospital Infections*

   In combating bacterial infections (such as staphylococcus and vancomycin-resistant enterococci) in hospitals, Gawande observed a similar scenario. There were those who succeeded in keeping infections low where others did not. When the successful methods and attitudes were examined, advertised, and measured, the hospital community in general improved its performance. Cultural change from within succeeded where top-down directives to wash hands and clean instruments more often had failed.

3. *AT&L System of Systems (SoS) Engineering Guide*

   An example from the world of SE is exemplified in version 0.9 of a System-of-Systems (SoS) Engineering Guide created for the Office of the Undersecretary of Defense for Acquisition, Technology and Logistics (AT&L). The team working that project was charged with extending traditional SE to SoS engineering. They logically "extended" each of the 16 technical and technical management processes in the Defense Acquisition Guidebook (DAG), leaving the basic construct intact.

   That version of the guide was presented to active SoS engineering practitioners working on programs that had demonstrated some measure of success. They were asked to comment on whether that view coincided with their experience and to describe how it differed where it did. These practitioners reported that things were indeed different. For example, the decision analysis process for SoS engineering is more complex than it is for conventional SE because it needs to consider potential ripple effects on constituent systems for each major SoS decision. Also, the resolution of any contentions involves a greater number of stakeholders than a single system would. The simple, process-by-process "extensions" quickly became burdensome and intractable.

   They also reported that they did use the 16 DAG processes, but the nature of the SoS environment required that they be used in a different way. Rather than use them as a linear, sequential progression of processes, they were used as components or tools that needed to be configured in different ways to address SoS engineering issues as they arose. They tended to be assembled into core elements that were not consecutively executed, but rather were executed continuously and contemporaneously. The interactions among the core elements were more complex, as well. The guide was rewritten to reflect the view of these practicing SoS systems engineers. The rewritten SoS Engineering Guide has been played back to a larger audience of practicing system engineers, and the response has been positive across both government and industry. The important point for this discussion is not the details of what was discovered, but rather, the way in which it was discovered—not by an extension of conventional wisdom, but from successful practitioners.

1.3 Fundamentals of Patterns

Pattern History

Architect Christopher Alexander introduced "patterns" as a way of organizing design in the context of constructing buildings and cities.[7] Patterns are frequently used to capture the essence of design solutions for recurring problems. They have also been successfully applied to software

[7] C. Alexander, *The Timeless Way of Building,* Oxford U. Press (1979).

design[8] and system architecture.[9] They are relatively new to systems engineering, having been first proposed in an INCOSE conference in 1998.[10] They were subsequently used by Haskins to deal with some social aspects of systems engineering[11]. Simpson[12] and Cloutier[13] have made some first attempts at systems engineering patterns. This report uses the concept of patterns to capture and communicate positive deviants observed in the study of SE and acquisition of IT-intensive government systems. Christopher Alexander aptly describes the use of patterns:

> *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core solution, in such a way that you can use this solution a million times over, without ever doing it the same way twice.*[14]

1.3.1 Definition of Patterns

Humans Perceive Patterns
Generally, a pattern is a "rule of thumb" for designing things. Specifically it has three parts:

1. *Context*: Defines the larger system that preceded and constrains the current design.
2. *Forces at work*: Expresses the relations among elements within that given context.
3. *Solution*: Presents a design or configuration of elements that successfully resolves those forces.

Because it is a "rule of thumb," a pattern captures only the essential nature of a design, and it can be reused again and again to serve a wide variety of situations and local needs. Patterns by which people build things are usually the result of accumulated knowledge and successful experiences of many people over time. As in positive deviance, patterns are based in practice.

Pattern Principles
Embodied in the definition of patterns are three key principles:

- *Tested Methods*. Because successful patterns are steeped in actual practice, they provide tested methods. For that reason, proper use of a pattern can speed up the development process and mitigate the risk of hidden problems surfacing later.
- *Latitude for Innovation*. Because a pattern provides only the core or essence of a solution, there is latitude to tailor it to the particular situation at hand.
- *Adaptation and Learning*. Once a pattern is described and named, it provides a mechanism to communicate among peers, and therefore, to improve it. In discussing complex systems, Holland[15] speaks of rule discovery, that is, the generation of plausible hypotheses, which centers on the use of tested building blocks. In this way,

[8] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, New York: Addison Wesley, 1994.
[9] Martin Fowler, *Patterns of Enterprise Application Architecture New York*, Addison Wesley, 2002..
[10] Robert H. Barter, A systems engineering pattern language. *Proceedings of the 8th Annual International Symposium of the International Council on Systems Engineering,* Vancouver, BC, July 1998.
[11] Cecilia Haskins, Using patterns to transition systems engineering from a technological to social context, *Systems Engineering*, Vol. 11, Issue 2, Summer 2008, pp. 147–155.
[12] J. Simpson and M. Simpson, Foundational systems engineering patterns for a SE pattern language, *Proc. 16th annual INCOSE Symposium*, Orlando, Fla., July 2006.
[13] R. Cloutier and D. Verma, Applying pattern concepts to enterprise architecture, *J. Enterprise Architecture* 2(2), May 2006, pp. 34–50.
[14] Alexander, p. 66.
[15] John H. Holland, *Hidden Order: How Adaptation Builds Complexity*, Reading, Mass.: Helix Books,1996.

past experience is directly incorporated, and innovation is allowed through wide latitude in application of the patterns. Identification of patterns provides both guidance and learning.

1.3.2 Examples of Patterns

Patterns Are Common

Patterns are routinely used in several areas:

- *Construction.* The construction industry relies heavily on patterns, from the arrangement of houses in a development to the way in which walls are built with studded ribs and top and sole plates. Likewise there are well-used patterns of electrical and plumbing distribution. Although construction is based on these recurring patterns, no two developments or buildings are ever the same.
- *Software*. Object-oriented programming relies heavily on patterns for such things as delegation (where one object defers a task to another object) and aggregation (where simple objects are combined into more complex ones). Patterns also exist at the architectural level, for example, Model-View-Controller pattern. These patterns have evolved over time through the concerted efforts of many designers and have endured due to their relative success in use.
- *The World Wide Web (WWW).* The World Wide Web is a good example of a large IT-intensive enterprise using patterns to build system components— "hypertext linking" and "browser rendering of web pages," to name two. Easy access to knowledge for collaboration with a never-ending mix of partners confirmed these as patterns of success. Yet, the variety of new technologies used to implement patterns of the WWW, as well as the variety of collaborations that take place, seems almost infinite.

1.3.3 Template for Patterns

The Template We Used

There is no single template for capturing patterns. However, a template should clearly state the problem, its context, and the forces at work. The essential elements of the solution should be described along with clarifying examples and sketches. To be widely used, each pattern should have a meaningful name to facilitate communication and learning. The easier to remember and the more descriptive it is, the better. The template we used to capture the patterns we found is shown in Table 1-1.

Table 1-1. Template of Design Patterns

Name	***Descriptive Name for Pattern***
Summary	• Sums up the pattern in an easy to remember sentence or two.
Context	• Defines those parts of the problem that precede the system to be developed. • Lays out the background and context of the problem that best motivates this pattern.
Problem	• Describes the essence of the specific problem to be solved. • Defines any variations in the problem, symptoms, or context that have a bearing on the solution.
Figure	• Provided separately in the accompanying text.
Forces	• Factors that must be weighed to reach a solution. • Explains why this is a hard problem to solve. • Entices the reader to read on to find out how it is solved.
Solution	• Describe the essential elements of how and when to implement this solution. • Relates those elements to one another. • Describes how they work together to reinforce and/or oppose one another. • The specific implementation is less important than the essential elements. • When not to use it may imply another pattern.
Examples	• Provided separately in the accompanying text.
Outcome or Resulting Context	• Describe the result of applying the pattern. • What constituted success. • It gives both the cost and benefit, so potential users of the pattern can judge whether or not to apply it to their particular situation.
References	• Further reading or other related patterns presented in accompanying text.

2 Identifying Patterns of Positive Deviance

Government Acquisition Environment
Across government sectors, the environment in which a program office acquires IT-intensive systems is constrained by the context of the existing acquisition community and information enterprise. The interdependencies of the diverse stakeholders and IT components result in a complex environment. As stakeholders in the acquisition process attempt in earnest to improve it, the environment changes and seems to thwart their efforts. At times, delivering a needed operational capability within fiscal, regulatory, and schedule boundaries seems like an over-constrained problem—a "Mission Impossible."

Needs versus Resources
A number of forces are at work. First, an acquisition program begins with a user who has a compelling need for some capability. That need is articulated, and it competes for the limited financial and personnel resources within its own organization. Likewise, the leaders of that government organization compete all the way up their management chain to Congress through the Federal Planning and Budgeting process. Eventually a program office that is usually separate from the user is designated and given resources and responsibility to develop and deliver the capability. As with IT developers in the private sector, the program office finds itself trying to resolve the conflict between the forces of needs and resources available.

Other Forces at Work
The government sector has additional complications. It is ruled by a large body of law and regulations that control government acquisition, whereas systems engineering is rooted in a set of well-defined and expected processes. As the program office tries to find its way through this web of competing forces, other government entities are similarly acting on their own behalf, changing the landscape of the overall enterprise, and in particular, the environment for any one program office.

Flourish or Merely Survive
As a result, it is common for a program office to follow the practice of merely surviving, with the result that it contributes to keeping the government acquisition of IT-intensive systems trapped in poor performance. However, within any community, some do more than survive. They flourish because they exhibit patterns of positive deviation in their practice.

Seeking Positive Deviance
Thus our approach is to look at what people and organizations actually do in successful situations. On the surface it may appear that different people find solutions that are never the same as others, but at a deeper level there can be invariant elements of those solutions. We look for those essential elements of repeated pathways to success. The resulting patterns of positive deviance do not define the final solution, but they do give a program office and its systems engineers a starting point and a direction. Thus acquirers of new systems can use tried-and-true patterns to guide them through the myriad decisions ahead of them.

Summary of What We Did
In conducting this study, we followed the methodology in Appendix A and captured our results in several well-defined patterns. We focused on IT-intensive systems across a wide spectrum of government sectors. We went to the practicing systems engineers and managers who actually

provide SE and acquisition support to government programs offices and asked them to identify successes. We then interviewed the project teams that had those successes and asked them what they had done. It was from these interviews that we constructed the patterns of positive deviance reported in Sections 3, 4, and 5.

Yet To Be Done
What remains to be done is for the patterns of positive deviance to be communicated back to the SE and acquisition community from which the results were obtained. This was partially accomplished. As the study progressed, some of these patterns became evident. At the end of the interviews, we were able to share our preliminary insights with the teams, comparing and contrasting their experience with other programs. This report captures the complete set of insights, and it is our intention to make them available to the practitioners whose experiences were the basis for our conclusions.
.

2.1 Defining Success

Success Is in the Eyes of the Beholders
Defining a practice as successful depends on the criteria for success. In Alexander’s world of building buildings and cities, his criterion was somewhat ephemeral: he wanted the design to make people “feel more alive.” His test was that people continued to use the designs over and over again in the enrichment of their daily lives. When designs made people feel more alive, it reinforced a continual cycle of practitioner reuse.

Success in Government Acquisition
For the acquisition of IT-intensive systems, the success criteria are less ephemeral, but no less difficult to define. Stakeholders in the government acquisition systems are looking for ways to provide utility to users of the systems, to provide it in a timely way, and to satisfy the host of government entities funding and regulating the set of systems. Our success criterion was broad. We sought programs that deviated in a positive way from the norm in delivering something tangible and useful to their users without any offsetting degradation. That deviation could be a marked improvement in performance, cost, or schedule. We deliberately left the definition of success open in seeking recommendations for programs to interview and during the interviews.

2.2 Identifying Successes

Success Defined from Within
Initially we met or corresponded with about a dozen senior systems engineering leaders from across MITRE’s several FFRDCs (federally funded research and development centers). They represented a cross-section of government sectors with hundreds of programs in Defense, Intelligence, Aviation, Enterprise Modernization, and Homeland Security. We asked them to identify IT-intensive programs that had demonstrated one or two notable successes without really defining what we meant by success. Many of them had immediate answers for successes in their programs, and others went to their project leaders for information. This activity resulted in the identification of about 30 programs.

Choose a Representative Cross-Section
We next chose a dozen programs that represented a diverse cross-section of government IT-intensive programs. They were U.S.-only developments, but several had coalition or

international partners. Six of them had multi-organization users, and the other six were either single-organization or a single class of users. To assess the complexity of those programs, we used an Enterprise Systems Engineering Profiler tool developed by Stevens.[16] It represents the complexity of a systems development effort across eight dimensions:

- Mission Environment
- Scope of Effort
- Scale of Effort
- Acquisition Environment
- Stakeholder Involvement
- Stakeholder Relationships
- Desired Outcome
- System Behavior

These dimensions are plotted on a "spider chart" in which the farther out the assessment is on each axis, the more interdependent, uncertain, or variable the program is in that dimension. The more the overall locus of points is confined to the interior of the chart, the more the system development follows a conventional acquisition. The more it is on the periphery, the more it represents development of a complex system. Each of the programs chosen for the study had moderate or high complexity in several of the dimensions. The results are shown in Figure 2-1.

2.3 Interviewing the Teams

Wisdom from the Workers

The teams we interviewed consisted of a project leader (PL) and key project staff. Generally the program manager in the government program office uses the SE and acquisition expertise of the PL, and often the PL serves as either the designated or de facto chief engineer in the program office. These are the front-line practitioners who cope with the exigencies of the government acquisition system and are in a position to influence or observe positive deviance in acquisition performance.

Getting to the Teams

Because these teams are extremely busy actually doing SE and acquisition, it was difficult to schedule time with them. Therefore, we made the process as streamlined as possible, requiring only the following time commitment:

1. A brief meeting (less than an hour) with the PL to explain the process and objectives
2. A more focused session with the whole team for about two hours
3. A follow-up with the team to ensure that the case notes we captured were accurate and to solicit any further elaboration. (Usually the case notes were only a few pages and follow-up was accomplished quickly by email.)

The correspondence and briefings accompanying these steps are given in Appendix A, Study Methodology.

[16] R. Stevens, *Engineering Mega-Systems: The Challenge of Systems Engineering in the Information Age*, Boca Raton, Fla.: Auerbach/Taylor & Francis, (2011), pp. 118–120.

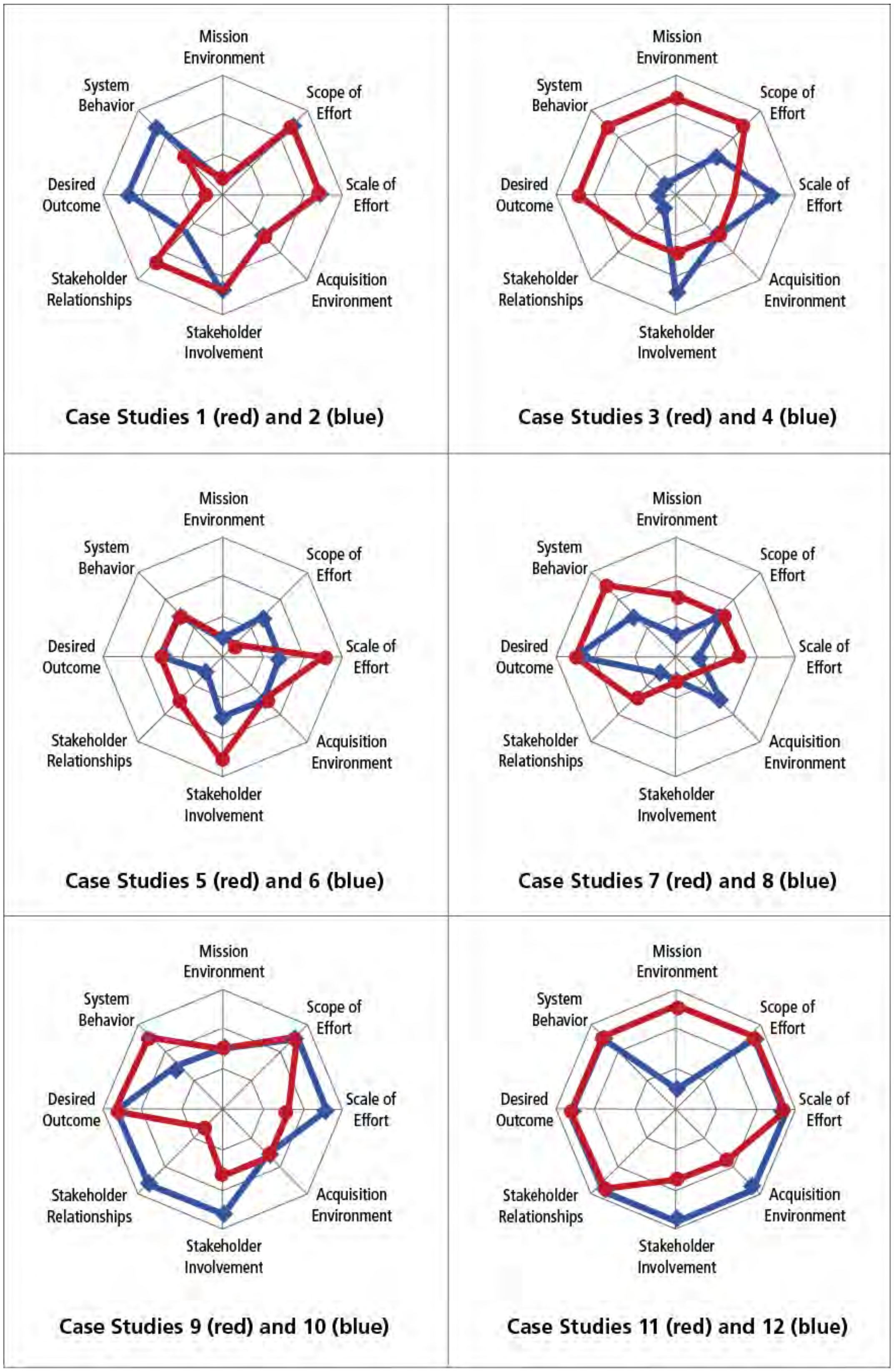


Figure 2-1. SE Profiles of the 12 Programs (2 per chart)

Listening and Learning
During the interviews, we were careful not to define success for the team members, but simply to ask them what they did and how they did it. Occasionally we would seed the conversation with questions like “what worked well?” and “what were you most proud of?” We guaranteed them that we would not publish the names of the programs nor the people we interviewed. For the most part, the conversation was free-flowing, and we did not shape it except to bring it back to a discussion of the practice if it strayed off the subject or toward hypothetical situations.

Talk Begets Talk
What is noteworthy is that the team discussion environment tended to spark individual memories, and the team members quickly became enthusiastic about their practices. In most cases, at the end of a two-hour session, a team wanted to continue to discuss their work and we generally accommodated them. It appeared that those on the front lines who are most involved in successful practices, seldom get the opportunity to talk about how they do what they do and reflect on what worked well. There is a wealth of knowledge buried in the minds of those involved in the everyday practice of engineering and acquiring IT-intensive systems.

2.4 Capturing Their Experience

Brief Case Notes
The results of the interviews were captured as case notes. These case notes identified the contributing practitioners, the name and nature of the program, and what the team did. The why, how, and when of what they did found their way into most of the case notes as well. Each of the draft case notes was reviewed and edited by the corresponding team for accuracy. Occasionally we had follow-up questions that they answered. In some cases, the interviewed team provided additional explanatory material. The final case notes for each program interviewed were distributed to that program's interviewed team members.

2.5 Communicating the Results

Emphasizing Patterns over Programs
Brief overviews of the programs we interviewed are presented in Appendix B, Case Overviews. The overviews provide high-level descriptions and salient features of each program but protect the anonymity of each program. The emphasis of this investigation is not on individual programs, but rather, by examining the 12 case notes, we were able to ascertain several repeated patterns of success in the practice. Details of the observed practice are incorporated into the patterns rather than being tagged to the specific programs. The emphasis is on the collective patterns rather than the individual programs.

Using Patterns of Positive Deviance
A simple analogy is useful in seeing how to use these patterns of positive deviance. If we think of the situation facing a program office (the context) as a landscape with topographic features like hills, valleys, and streams (the forces at work), a solution represents a successful path through the landscape. Repetition of a successful path for similar landscapes defines a pattern. Where government regulations define the landscape, patterns of positive deviance define credible mechanisms for dealing with them. Where regulatory or standard processes are mandated, patterns provide an acceptable tailoring of those processes. A properly chosen set of patterns provides the SE and manager with a map to guide them through myriad possibilities in government acquisition. Although the patterns are defined and discussed individually in this

report, in most cases they came together and worked collectively to help construct a successful acquisition program, rather than being isolated “silver bullets.”

3 Large-Scale Enterprise Patterns

Balancing the Supply Web and Harnessing Technical Complexity
This section defines the two large-scale (enterprise-level) patterns we observed: Balancing the Supply Web and Harnessing Technical Complexity. They represent the social and technical sides of the same coin. Both are based on successes in the systems engineering and acquisition of IT-intensive systems embedded in a larger Information Enterprise. Balancing the Supply Web addresses interdependencies between the enterprise stakeholders who have equities in the system being developed. Harnessing Technical Complexity addresses interdependencies among the system components that together deliver an operational capability and help to enable a unified enterprise. The acquisition coin has a social "head" and a technical "tail."

Über-Patterns (Patterns Within Patterns)
Each program achieved success because of the way it navigated these dual interdependencies. The large-scale patterns can be thought of as über-patterns with fine-scale properties defined by additional patterns (i.e., the patterns within the patterns), or they can be viewed as categories into which the fine-scale patterns fall. However they are viewed, they were consistently observed as patterns of positive deviance. They are explained Sections 3.1 and 3.2, and their corresponding fine-scale patterns are discussed in Sections 4 and 5.

3.1 Balancing the Supply Web

Satisfy and Suffice
This pattern sufficiently balances the web of potentially conflicting stakeholder demands as a strategy for supplying a new capability in a complex development and acquisition environment. It is akin to how a balanced ecosystem thrives while individual species have their ups and downs. It is illustrated in Figure 3-1, summarized in Table 3-1, and described in in the remainder of the section.

3.1.1 Context of Balancing the Supply Web Pattern

Supply Web versus Supply Chain
The reality of the government acquisition process in an IT-rich environment is that the program offices are best viewed as being immersed in a supply web rather than being the middlemen in a supply chain. In a traditional supply chain, goods and services flow in one direction and compensation flows in the reverse direction. Government acquisition is more complex. The money flows from tax dollars through the Congress down through various departments of government to a designated program office. From there it fans out to contractors as well as other government agencies involved in the acquisition. The goods and services flow from contractors to end users. Certain government agencies establish regulations and rules, then expect assurance of compliances to flow back to them. There are flows to and from independent test organizations

as well as various “watchdog” agencies. It is this supply web that is the real context of government acquisition and what makes it so difficult.

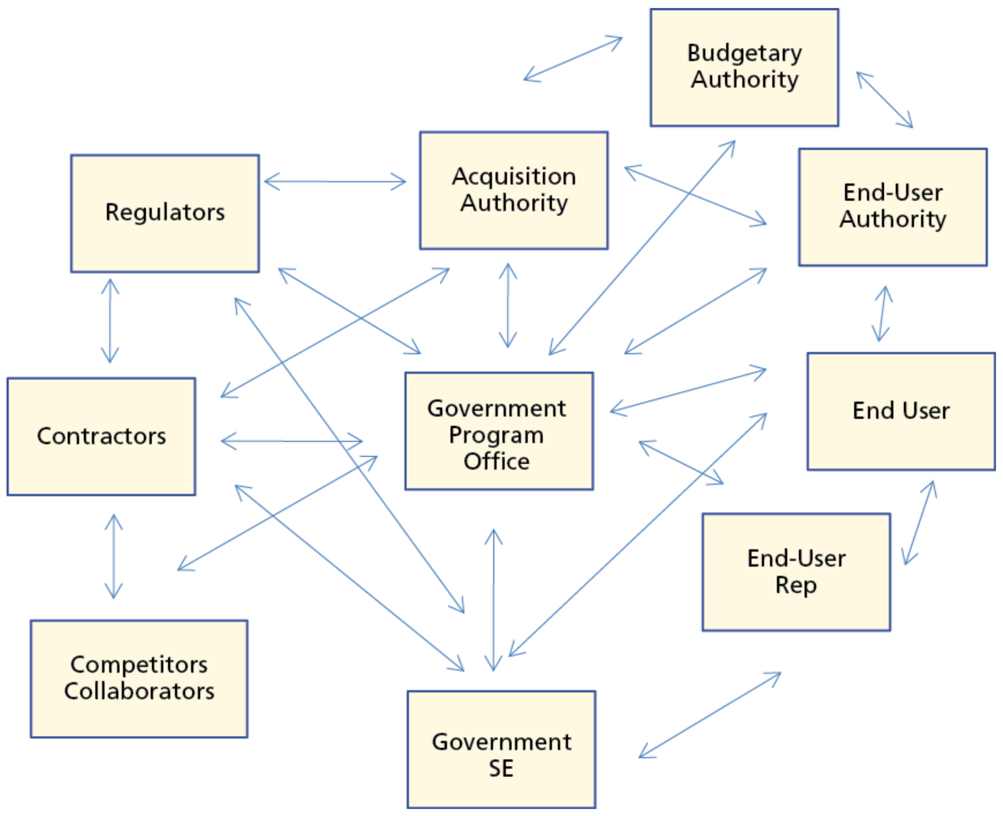


Figure 3-1. Balancing the Supply Web Pattern

Systems Seldom Stand Alone
In addition, increasingly, the government develops operational capabilities by combining new components with existing infrastructure and systems. Rarely does a new capability result from a system composed of a single piece of equipment or software or from investment by a single authority, that is, they are delivering a capability in the context of the enterprise. Acquisition now becomes more complex because of the web of dependencies between the various components and stakeholders.

Resources Are (Almost) Always Limited
Program offices have limited resources—money, talent, proven processes, and so on.—with which to perform a great deal of stage setting and planning. As their resources are depleted and resupplied or exchanged, they are in a constant competitive game, marked by intermittent milestones and payoffs. The traditional supply-chain is a platform-centric view of system development, whereas the supply web is a network-centric view of component development and capability evolution. With enterprises moving from a platform-centric to a net-centric character, the figures of merit for how well they are achieving their objectives are more diffused and implicitly defined. Control gives way to influence and accommodation as the system or component being developed must fit into the context of the larger enterprise rather than exist as a standalone system. The traditional supply chain of contractor-developer-user becomes a supply

web of all affected stakeholders. The one-dimensional supply chain view of acquisition often causes program offices to underestimate resources required to balance the supply web.

Table 3-1. Balancing the Supply Web Pattern

	Balancing the Supply Web
Summary	Balances social interactions between participating stakeholders in a complex development and acquisition enterprise.
Context	• Supply web versus supply chain. • Money flows down from Congress and fans out to contractors. • Goods and services flow from contractors to end users. • Regulation and standards flow from other government entities. • Capability is not delivered with a standalone system under a single authority.
Problem	• Need balanced strategy to address legitimate needs of all stakeholders. • Must cope with multiple needs and limited resources.
Forces	• Conflicting definitions of success among competing stakeholders. • Stakeholders use their resources and influence to compete. • Finite program office resources. • Current acquisition practice is aligned with a traditional supply chain concept.
Solution	• Model and monitor the entire web of stakeholders and their interactions over time. • Apply and adjust limited program resources judiciously to continuously balance stakeholder needs. • Leverage one stakeholder's resources to satisfy another's. • Steward the acquisition context as well as develop the capability.
Outcome or Resulting Context	• Systems successfully delivered new capabilities in a balanced supply web. • Delivered both operational capability and enterprise context.

3.1.2 Problem

Balancing Multiple Demands

The stakeholders have legitimate needs that they would like to have fully satisfied. Usually the program office does not have enough resources to do that. Therefore, a balanced strategy is needed that successfully copes with the interdependencies of the stakeholders and the sometimes conflicting constraints they levy. It must exploit not only its internal resources but also those available from the enterprise stakeholders. Because stakeholders' influence and equities differ from situation to situation, the interactions need to be strengthened with some stakeholders and lessened with others. The program office must manage not only stakeholder needs but also their expectations and, ultimately, their acceptance that the enterprise capability is sufficient and satisficing.

3.1.3 Forces at Work

Push and Pull

Different perspectives and values across a stakeholder community combine to create tensions and incompatibilities for the program office. The Balancing the Supply Web pattern is essentially a "stakeholder balancing act." We observed several principle forces at work:

> *Conflicting definitions of success*: Stakeholders have different views of the value of a capability, equipment, or software being developed. Some are differences in degree; others are differences in kind. The end user of a system may place a premium on its usability, reliability, or response time, whereas a CIO may especially value its interoperability within the enterprise. Although all stakeholders might agree on a common overarching program goal, each stakeholder frequently defines success differently. Often, different parts of the government are responsible for setting requirements, providing logistics support, defining key performance parameters, and developing the system. Accepting the different stakeholder views of success as immutable and attempting to satisfy them all can lead to confusion, frustration, and dissipation of both resources and effort, ultimately resulting in failure to deliver.
>
> *Competing stakeholder resources and influence*: Stakeholders have resources (e.g., funding or personnel) and influence (e.g., in the form of end-user legitimacy or policy compliance imperatives) that they use to define the value proposition of the item being developed. They use these resources to move other stakeholders toward their own definition of success. Ultimately the results of all this enterprise-wide cooperation and competition determine program success or failure.
>
> *Finite program office resources*: The program office has limited resources (e.g., funding or time) to develop and deliver an item that stakeholders view as providing value. Many forces conspire to affect this situation. One underappreciated force is the implicit view of acquisition as a supply chain, which therefore discounts the effort required to balance the supply web.
>
> *Reality and practice mismatch*: Current acquisition practice presumes a traditional supply chain, where a platform user (or proxy) provides resources to get an operational need established, and a contractor delivers the systems to meet that stated need. Here the government program office acts as a go-between, bringing funding and other resources to bear and in finding a price-performance point that satisfies both ends of the supply chain. In the reality of a supply web, stakeholders, as well as enterprise components, must come together to provide an operational capability. The complex and shifting reality of network-centric acquisitions cannot be fully addressed by the plan-and-execute practices of platform-centric development. Policies or requirements that enable enterprise-level capabilities impose constraints on the ways in which programs develop solutions. Although platform-centric (supply chain) and network-centric (supply web) views are not in direct conflict with each other, funding and acquisition policies, procedures, and processes remain largely platform-centric. They are changing to better align with the network-centric view, but at a slow pace. All of this makes acquisition more complex.

3.1.4 Solution

Seeing the Supply Web

Successful program offices strike a balance between the needs of the users and those of all other stakeholders. The positive deviants were very much aware of the interdependencies of their particular set of stakeholders. They viewed acquisition not as a supply chain but as a supply web, where contractors, competitors, budget authorities, regulators, and so on all interacted with the program office as well as with each other. As in the studies of System Dynamics[17] or Food Webs,[18] the program office must explicitly identify and understand the interdependencies to be able to manage and balance the supply web.

Influencing the Supply Web

Successful program offices also apply their limited resources judiciously across this supply web, expending more or less of their own resources and stimulating interactions between stakeholders as needed. Because they understand the interactions, they are able to go beyond balancing stakeholder needs to actively influencing stakeholder perceptions of the broader enterprise context and acceptance of its consequences.

Delivering Capability and Context

Depending on project circumstances and the environment, getting stakeholders to understand and accept a context beyond their own perspective is accomplished in different ways. We observed several recurring strategies in successful programs that balanced the supply web. Several of these strategies are captured in the fine-scale patterns discussed in Section 4. In all cases, delivering a capability to users remains job one, but delivering context that influences stakeholders is also a job that needs to be undertaken. Delivering this context is a form of influencing the supply web. We call it out separately because of its importance. Thus the new stakeholder balancing act requires delivery of capability *and* context.

3.1.5 Examples

Consistent Pattern with Variations

The supply web pattern was consistently observed, although variations in the domain and the circumstances of the program affected the character and intensity of the interactions. In cases such as “delivering an operational capability to a national security effort,” the user interaction was king. In situations such as “delivering network communications to military bases all over the world,” the budget authority and regulators ruled the roost. In air traffic management, safety was paramount and acceptance testing got focused attention. During source selection, “fairness” interactions with contractors were paramount. Unprecedented problems expanded interactions with technology providers. Cross-organizational operations called for interaction with enterprise architects. Without this supply web understanding, a program is likely to fall into a one-size-fits-all approach that leads to deep trouble or even failure.

[17] Sterman, J., Exploring the next great frontier: System dynamics at fifty. *System Dynamics Review*, 23, 2007, pp. 89–93.

[18] J. A. Dunne, R. J. Williams, and N. D. Martinez, Network structure and biodiversity loss in food webs: Robustness increases with connectance, *Santa Fe Institute Working Paper* 02-03-013), 2002.

3.1.6 Outcome

Successfully Balanced
Positive deviant government program offices work diligently to understand, accommodate, and leverage the enterprise context in which they have to deliver a capability. The entire set of stakeholders is serviced but with different intensities in different situations. Each program office successfully delivers a capability to users, while purposefully communicating the enterprise context to the other stakeholders. They balance the supply web by delivering operational capability and enterprise context to the stakeholders. In an extreme situation, some stakeholders may oppose and successfully stifle the needs of others. Although not observed in any of the cases examined, a program office may even be able to take advantage of this circumstance.

3.2 Harnessing Technical Complexity

Ordered Complexity
This pattern harnesses the technical interactions between cooperating systems in a networked enterprise. It is illustrated in Figure 3-2, summarized in Table 3-2, and described in the remainder of the section.

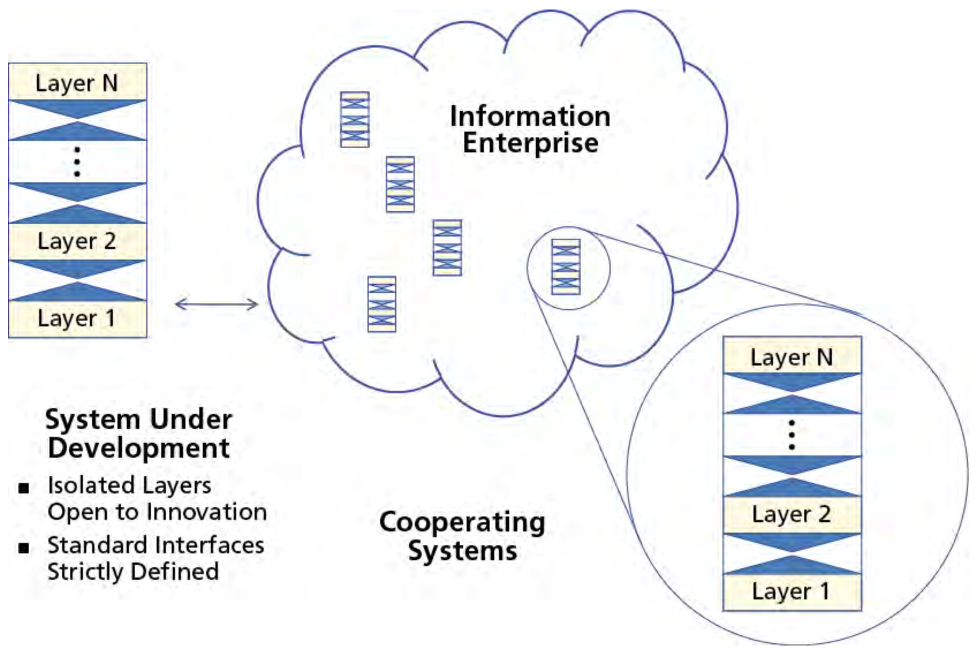


Figure 3-2. Harnessing Technical Complexity Pattern

3.2.1 Context of Harnessing Technical Complexity Pattern

Connected Islands of Integration and Innovation
Layered architectures have been pervasive since the early days of information systems. The 2-tier and 3-tier architectures that separated data and business logic from user interfaces gave way to Client-Server and eventually the N-tier architectures we see in the Internet and World Wide Web. Because each layer provides functionality different from each other layer and the interfaces between layers are standardized, they can be isolated from one another. That means changes in

one layer will not affect the others. This allows innovation within the layers. Because the protocols by which one layer accesses the services of the layer below it are standardized on an enterprise-wide basis, systems built with this layered architecture can cooperate by sharing functionality at any given layer.

3.2.2 Problem

The Devil Is in the Details

The program office is charged with creating a new and innovative capability while also integrating its system in an existing IT enterprise. The N-tier architecture is fairly well established and its operation is well understood. However, not all programs in the enterprise agree on the definition of layers or on what the coupling should be between layers. The devil is in the details.

Table 3-2. Harnessing Technical Complexity Pattern

	Harnessing Technical Complexity
Summary	Harnesses the technical interactions between cooperating systems in a networked enterprise.
Context	• N-tier architecture is pervasive information system model. • Innovation is enabled inside the layers. • Integration is enabled by standard interfaces between layers.
Problem	• Program office must create a new capability integrated into existing IT enterprise. • Not all programs agree on layer or coupling standards. • Funding and program direction focused on individual system capability, not changing to improve enterprise improvement.
Forces	• Enterprise governance mechanisms specify enterprise-wide standards. • Autonomous program offices use unique (to them) standards and technologies to meet local needs and want to continue to use what is already developed. • Pressure to change standards or agree on a common standard (to which all groups could translate). • Existing infrastructure is not optimized for the components being developed. • Continually evolving enterprise standards.
Solution	• Codify a small number of agreements in a Strategic Technical Plan. • Agree on standards (e.g., organizations of the layers and the coupling between them) by observing most common practices and those relatively easy to implement. • Enterprise-wide Strategic Technical Plan ensures some interoperability with others in enterprise. • Allow innovation within the layers. • Force integration with interface standards.
Outcome or Resulting Context	• A small amount of standardization resulted in a large amount of interoperability. • Enterprise complexity is harnessed through integration at convergence points and innovation in the layers between these convergence points. • The enterprise can adapt to changing technologies and operational needs without becoming too chaotic to keep up with change.

3.2.3 Forces at Work

Conflicting Standards and Protocols
Subgroups within the enterprise may have already developed infrastructure components or layer protocols that they want to continue to use. However, to create a true enterprise application or service, they would have to change their standard to accommodate other groups or agree on a common standard (to which all groups could translate).

Suboptimal Performance
When a program office develops a component to enable a new capability, it does so in conjunction with other components and existing infrastructure. The infrastructure most likely is not optimized for the component being developed, but rather is designed for broad use and influenced by what has come before. This is analogous to using the federal highway infrastructure to travel between two points even though it is not the shortest distance between start and destination.

3.2.4 Solution

Strategic Technical Plan
A Strategic Technical Plan codifies a small number of agreements on such things as the organizations of the layers and the coupling between them. These agreements are generally not conceived by experts from scratch as the proper way to do things, but rather are observed by practitioners as the common way things are done across the enterprise. Typically people observe a small number of common elements that are responsible for a large percentage of the workflow in an enterprise. These common elements represent points of strategic convergence and serve to bring order to the enterprise while allowing great variety between the convergence points. For the rare occasion when proposing a new “invention” may be warranted, simplicity and implementation ease are driving factors.

By adhering to an enterprise-wide Strategic Technical Plan, each separate program is assured of at least some interoperability with others in the enterprise. By carefully choosing a small number of standards that represents a large fraction of the enterprise workflow, the enterprise comes alive with possibilities not previously experienced. Some new capabilities are added without ever having been previously planned. In the cases where proprietary implementations are not able to be changed, they are encapsulated or translated to the enterprise standards.

3.2.5 Examples

Mimicking Familiar Systems
Most of the programs interviewed had legacy systems with 2-tier or 3-tier architectures. What the successful programs did was take advantage of Internet and World Wide Web standards. They adopted the standard ISO model for layers, Internet Protocol (IP) for enterprise routing and communication, XML for data representations, and so on. However, it was frequently the case that the databases, message sets, or service interfaces were not common across the enterprise. Some detailed examples are given in Section 5.2, Riding on the Infrastructure, and Section 5.3, Loose Couplers, on how this situation can be handled.

3.2.6 Outcome

Harnessed Complexity
A small amount of standardization results in a large amount of interoperability.

4 Supply Web Patterns

Eight Supply Web Patterns

The Balancing the Supply Web pattern discussed in Section 3.1 addresses the multiple conflicting and competing stakeholders involved in acquiring a new IT-intensive system. Those stakeholders include operational users, customer representatives, policy compliance organizations, contractors, and so on. Although all the programs had an ultimate goal to deliver operational capability to the end user, they each had a “back story” relating to the other stakeholders. Important differences in the stakeholder situations drove the programs to different paths for success. Differences in the scope and scale of program office development efforts, resources available, and other environmental factors change the dynamics of the forces at work, and therefore required that the programs adopt different strategies and techniques. However, we observed several recurring strategies in successful programs and captured them as design patterns within the overall Balancing the Supply Web pattern:

- ***Up Close and Personal*** establishes strong and intimate ties with end users to ensure satisfying a high-priority, pressing need.
- ***Close, But Not Too Close*** concentrates on getting a large number of end users to accept a standard set of capabilities and compensating them with rapid deliveries of their most valued capability.
- ***Divide and Conquer*** deals decisively with all stakeholders by dividing them into groups and satisfying each group's interests separately.
- ***Circle of Trust*** fosters positive social interactions among stakeholders to improve the willingness of opposing factions to compromise.
- ***Role and Responsibility Subnets*** clearly defines subnets within the stakeholder community for each decision or product to be supplied.
- ***Seek Secondary Sources*** seeks small flows of resources from secondary sources that have large impact on robustness of program and capability delivered.
- ***Network Beats the Node*** deliberately takes advantage of relationships in the network of stakeholders to create a resource greater than the sum of the parts.
- ***Top Cover*** uses informed acquisition authorities to shape the stakeholder environment.

The first three design patterns represent alternative strategies for matching stakeholder needs to available program office resources. The next two recognize the importance of positive social interactions and clear roles and responsibilities in managing a supply web. Seek Secondary Sources recognizes that in complex systems, small changes can have large effects. Network Beats the Node emphasizes that the power of the network is as important in the social context as the technical. Last, Top Cover relates back to Close, But Not Too Close, and recognizes the age-old maxim that the right kind of senior management involvement helps ensure project success. Although each pattern was strongly motivated by a particular program situation, the essential elements of multiple patterns were found in most of the programs. Each pattern is discussed separately in the following sections.

4.1 Up Close and Personal

This pattern establishes strong and intimate ties with end-users to satisfy a high-priority, pressing need. It is illustrated in Figure 4-1, summarized in Table 4-1, and described in the remainder of the section.

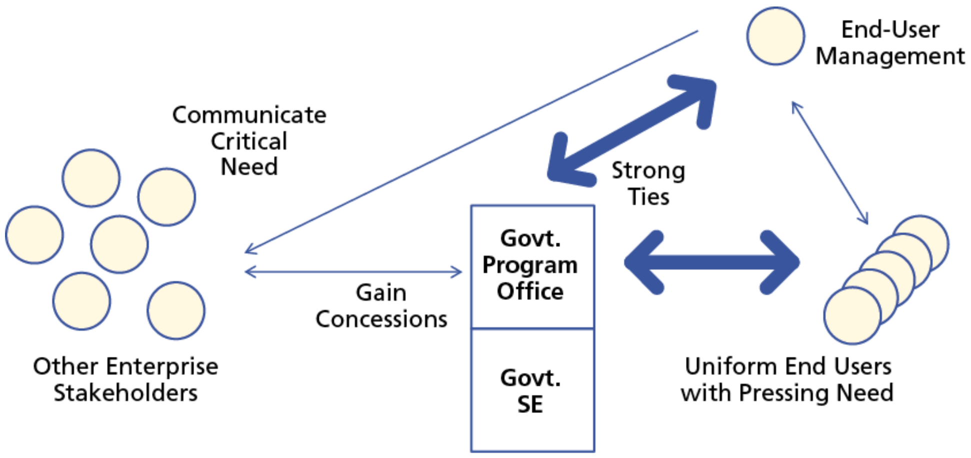

End-User Management
Communicate Critical Need
Strong Ties
Govt. Program Office
Govt. SE
Gain Concessions
Other Enterprise Stakeholders
Uniform End Users with Pressing Need

Figure 4-1. Up Close and Personal Pattern

4.1.1 Context of Up Close and Personal Pattern

Immersed in a Supply Web

This is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web. As is frequently the case, various stakeholders view program success differently. Satisfying one is usually at the expense of another. There seems to be no single, globally acceptable solution to the conundrum in which the program office finds itself. The program office must find a way to sufficiently satisfy the set of enterprise stakeholders.

A Pressing User Need

The context of the Up Close and Personal pattern has two factors that shape it. First, the user has a compelling and immediate need that the other stakeholders can recognize and acknowledge. Typically that need is motivated by circumstances like implementing an immediate safety requirement, responding to a new Federal law, or a delivering a critical operational capability. Second, the users are fairly homogeneous—they were trained alike and used similar equipment—so talking to one was like talking to another.

Table 4-1. Up Close and Personal Pattern

	Up Close and Personal
Summary	Establishes strong and intimate ties with end users to ensure satisfying a high-priority, pressing need.
Context	• Sub-pattern of Balancing the Supply Web. • A pressing user need that other stakeholders can well appreciate. • No globally acceptable solution. • Homogeneous set of users— talking to one was like talking to another.
Problem	• Program office must deliver needed capability in much less time than is normal for the formal acquisition process. • Difficult to meet a pressing need while satisfying all stakeholders.
Forces	• Immediate and compelling operational need drives development. • Other stakeholders advocate for their legitimate needs. • Meeting all needs may swamp ability to deliver critical need. • Falling short in delivering critical need constitutes failure.
Solution	• Allocate a large fraction of resources to interactions with end users. • Ensure that other stakeholders understand pressing need—get them vested in the end-user outcome to gain concessions. • Gain intimate understanding of end users' mission environment at all costs. • Establish credibility of systems engineering with users. • Frequently interact with users and higher headquarters to "show what you know." • Establish "First-Look" prototype where in-field user can provide feedback. • Maintain close relations despite personnel changeover. • Leverage user trust to enable some enterprise-level requirements to be met.
Outcome or Resulting Context	• Close relationships, rapid delivery, and responsiveness brought high user satisfaction. • High user satisfaction allowed some enterprise requirements to be met. • Other stakeholders relaxed requirements in return for some level of enterprise requirements being met, as well as the high-priority mission success.

4.1.2 Problem

Can't Satisfy Everyone
It is difficult enough to deliver a system that must meet a need on schedule while managing the requirements of a set of enterprise stakeholders. This becomes especially difficult when the user has a pressing and immediate need for the capability the system is to deliver. There is neither sufficient time nor resources to satisfy all stakeholders.

4.1.3 Forces at Work

Immediate Need Drives the Development
The strong user need is the pacing item for this development. The other stakeholders also have legitimate needs for which they advocate. Those can come in the form of regulations, financial

needs, additional requirements, competition, and so on. In the absence of any further shaping of the supply web, these might swamp the program office's ability to deliver the critical need. If the program office met the requirements of all the other stakeholders but fell short in delivering that critical need, the program would fail. Any balance to be achieved must favor the pressing user need.

4.1.4 Solution

Communicate Critical Need and Gain Concessions
Because the users had a compelling and immediate operational need, the program office saw its job as twofold: get close enough to the end user to ensure satisfaction, and diminish the effects of other stakeholders' pull on resources. The former was achieved by allocating a large fraction of program office resources to interactions with end users. The latter was generally approached by making sure the other stakeholders understood the pressing need—get them vested in the end-user outcome. When concessions were gained, a concentrated effort on the end-user outcome became more probable.

Delivering the Critical Capability Is Job One
The government systems engineers reasoned that their primary focus, or "North Star," had to be on delivering the needed capability to the operational user. Saying that the primary focus is on delivering value to the operational user is one thing. Actually doing it is quite another, particularly when the user is in field operations, thousands of miles away from the program office. The government systems engineers had to intimately understand the end users' mission environment at all costs. Establishing the credibility of systems engineering with operators was called "Job One." They pursued a deep and intimate understanding of operations and user lingo, as well as an ability to translate issues and ideas between the technical and operational communities. Because operators turned over frequently, systems engineers appreciated the importance of maintaining connections with in-theater users and quickly establish relationships with new personnel.

Show What You Know
Convincing the operational user's higher headquarters that the government systems engineers possessed the requisite knowledge was very important. They established frequent interactions with headquarters personnel and demonstrated their knowledge through such things as user-focused workshops and embedding program office systems engineers in both user and headquarters facilities. This created a climate of trust, which resulted in advocacy of program office decisions by headquarters. The program office team used transparency and openness to secure and leverage a better overall relationship with higher headquarters. (See Section 4.8, Top Cover.) The relationship with higher headquarters also helped the government team to understand and appreciate the role of the system under development in the broader mission context. The program office also used their extensive knowledge to give the regulatory agencies an appreciation of the user's critical need, resulting in some compliance requirements being relaxed.

4.1.5 Examples

Location, Location, Location
In several cases, government systems engineer knowledge of user operations was extensive. It was common for the systems engineers to travel to the field to gain firsthand knowledge of actual operations—even during military actions. It was also common for users, user reps, and focus

groups to be brought to the program office for consultation. In one case, the program established a facility where user reps could be housed during the development. This “up close and personal” approach established the credibility of the systems engineers with the users. Once established, it paid considerable dividends.

Knowledge, Knowledge, Knowledge
In more than one case, over the years, the systems engineering team had gained such intimate knowledge of the mission that they sometimes understood its context better than many users. With this systems knowledge and understanding of the broader mission context (gleaned from their relationship with higher headquarters), the systems engineering team gained a unique mission context perspective. Because of this, they were able to convince users of the importance of enterprise-level requirements (e.g., architecture, integration, and interoperability) to the success of the overall mission. Thus the systems engineers provided important context to the operational users who subsequently bought in to the enterprise aspects of the solution approach. A similar phenomenon happened with regulators. In one case, some key performance parameters (KPPs) were relaxed until the system’s data sources (legacy systems not designed with current KPPs in mind) were modernized. This enabled the current system to be fielded without delays and retrofitted later.

Continuity
In military situations, the program office realized that once credibility was established, they needed it to persist in light of frequent operator rotations and unplanned changes to in-theater operational concepts. One program established an effective first-look site at the commercial contractor’s development facilities that enabled in-field operators to work with pre-release versions of new features and functions. This rapid feedback kept the developers and users in lock step.

4.1.6 Outcome

Balanced Stakeholder Satisfaction
Close personal relationships, rapid delivery, and responsiveness to user feedback brought about a system with high user satisfaction. High user satisfaction enabled some enterprise shaping of deliveries beyond what the immediate users desired. Other stakeholders were willing to relax some of their requirements in return for some level of enterprise requirements being met, as well as the high-priority mission success.

4.2 Close, But Not Too Close

This pattern concentrates on helping an end-user authority establish policies and on compensating impacted end users with rapid deliveries of capability. It is illustrated in Figure 4-2, summarized in Table 4-2, described in the remainder of the section.

4.2.1 Context of Close, But Not Too Close Pattern

Similar Core Needs
This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) The context of the Close, But Not Too Close pattern is distinguished from that of the Up Close and Personal pattern by the large number of users with similar core needs who each expect a customized solution to fit their local situation. They have been accustomed to independently meeting their IT needs in isolation and in their own way.

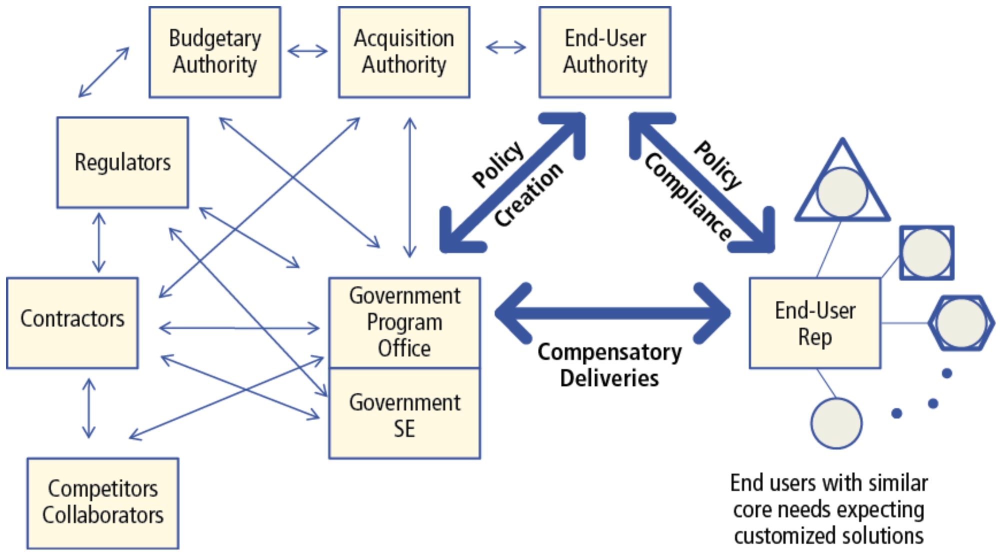


Figure 4-2. Close, But Not Too Close Pattern

4.2.2 Problem

Users Want Control
Even with similar core needs, a large number of users acting independently will come up with different solutions. They are accustomed to acquiring these themselves and do not want to relinquish control. There seems to be no single, globally acceptable solution. Satisfying one is usually at the expense of another. The program office must find a way to sufficiently satisfy the set of enterprise stakeholders.

4.2.3 Forces at Work

Reduce Costs, Improve Interoperability
The “Acquisition Authority” responsible for building and maintaining the IT enterprise charges the program with drastically reducing costs and improving interoperability. The budget is shifted from the user organizations to the program office. The user organizations believe that their capabilities will be diminished with this approach and are resistant to change. They also retain strongly held desires for tailored design solutions. The attitudes and vested interests behind the old way of doing business remain. The “User Authority” that sets policy is pressured to grant waivers based on seemingly well-reasoned arguments about the uniqueness of every user’s situation. However, doing so would exceed the available resources and stretch the schedule.

Developing an acceptable enterprise-wide solution and making it stick is a tough sell, and, in important respects, the technical problem is much less severe than the cultural one.

Table 4-2. Close, But Not Too Close Pattern

	Close, But Not Too Close
Summary	Concentrates on getting a large number of end users to accept a standard set of capabilities and compensating them with rapid deliveries of their most valued capability.
Context	• Sub-pattern of Balancing the Supply Web. • A large number of users with similar needs expect customized solutions. • Users accustomed to meeting their IT needs in isolation and in their own way.
Problem	• Program office must deliver an enterprise solution and sufficiently satisfy independent users. • Users do not want to relinquish control. • Users acting independently will come up with different solutions. • Satisfying any one user is usually at the expense of another. • Up Close and Personal doesn't scale to a large number of different users.
Forces	• Pressure to drastically reduce costs and improve interoperability. • Budget is shifted from user organizations to program office. • Users believe their capabilities will be diminished and resist change. • Users retain strongly held desires for tailored design solutions. • Insufficient resources for customized solutions. • User authority sets policy and issues waivers and sanctions. • Culturally entrenched user attitudes.
Solution	• Stay close enough to users to understand their legitimate needs. • Not so close as to get bogged down with individual users. • Divert resources to establish good relationship with user authority. • Secure top-down pressure from user authority. • Compensate users for lack of direct control by delivering early on capabilities valued most.
Outcome or Resulting Context	• Substantially reduced cost and improved integration of new systems. • Users sacrificed independence for speed of delivery. • Established credibility of the program office and the enterprise approach. • Close, But Not Too Close is an expedient way to navigate a contentious supply web.

4.2.4 Solution

Up Close and Personal Doesn't Scale Well

Program offices that started down the Up Close and Personal path that worked so well with a small number of homogeneous users found that it didn't scale to a large number of seemingly different users. They found themselves enmeshed in a quagmire of specialized requirements leading to individualized designs that were neither economically feasible nor a common, enterprise-wide solution. They soon adjusted their interactions with these users so they were close enough to still understand their legitimate needs, but not so close as to lose the view of the "enterprise forest" for the individual "user trees."

Top-Down Pressure, Sideways Rewards
Part of the solution was to divert resources away from strong interaction with individual users to establishing a good relationship with those in authority over the users. From a user's perspective, their demand for tailored solutions was superseded by compliance with their authority's policies. Non-compliance would bring sanctions. Another and equally important part of the solution was to offer these disgruntled users something other than a promise of benefits from enterprise solutions in an indeterminate future. For the most part, that something was rapid delivery of capability the users valued most. Once three things were in place— a good relationship with the User Authority, top-down policy, and commencement of rapid delivery—a self-reinforcing virtuous cycle began. Close, But Not Too Close may fly in the face of conventional acquisition strategy, but it turns out to be an expedient way to navigate a contentious supply web.

This pattern is similar to the Top Cover pattern discussed in Section 4.8. However, it differs in emphasis. That pattern focuses on using any Acquisition Authority (including the User Authority) to decrease the constraints on the program office. They both share the idea of compensatory deliveries to the users.

4.2.5 Examples

One program we studied followed this pattern precisely. Hundreds of user organizations all over the world were procuring their own brand of IT equipment for their individual operations. Not only was this very costly, but there were constant interoperability problems among the organizations. When the User Authority made the decision to centralize acquisition of the next generation of systems, there was a strong negative reaction—even non-compliance with the policy. The User Authority then cut budget and personnel at the user locations, making it impossible to non-comply. The program office looked at all their requirements and made a huge effort to deliver one of the most important capabilities quite rapidly. The distrust abated some. By continuing the rapid delivery of important capabilities, they eventually gained the trust of the users and the self-reinforcing cycle was in place. Eventually the users looked to the program office first to acquire new capability.

Other programs used this pattern with varying amounts of "closeness" to the users. In one case, the program office understood not only the mission very well, but also the international standards that the system must subscribe to. The involvement with the user was limited to bringing them an early prototype and getting their feedback. In another case, user representatives were convened to help solve system problems, but with the expressed condition that they leave their individual user-identities at the door. The peer pressure that the members of this user group placed on one another served to keep them focused on enterprise solutions, that is, solutions that were "close, but not too close" to their specialized and individualized needs.

4.2.6 Outcome

Close Enough for Engineering

The program office substantially reduced the cost and improved the integration of new systems into the field. The operational users sacrificed their independence and control for speed of delivery. The approach also helped establish credibility of both the program office and the enterprise approach and, over time, increased the receptivity of operational users to other changes.

4.3 Divide and Conquer

This pattern deals decisively with all stakeholders by dividing them into groups and satisfying each group's interests separately. It is illustrated in Figure 4-3, summarized in Table 4-3, and described in the remainder of the section.

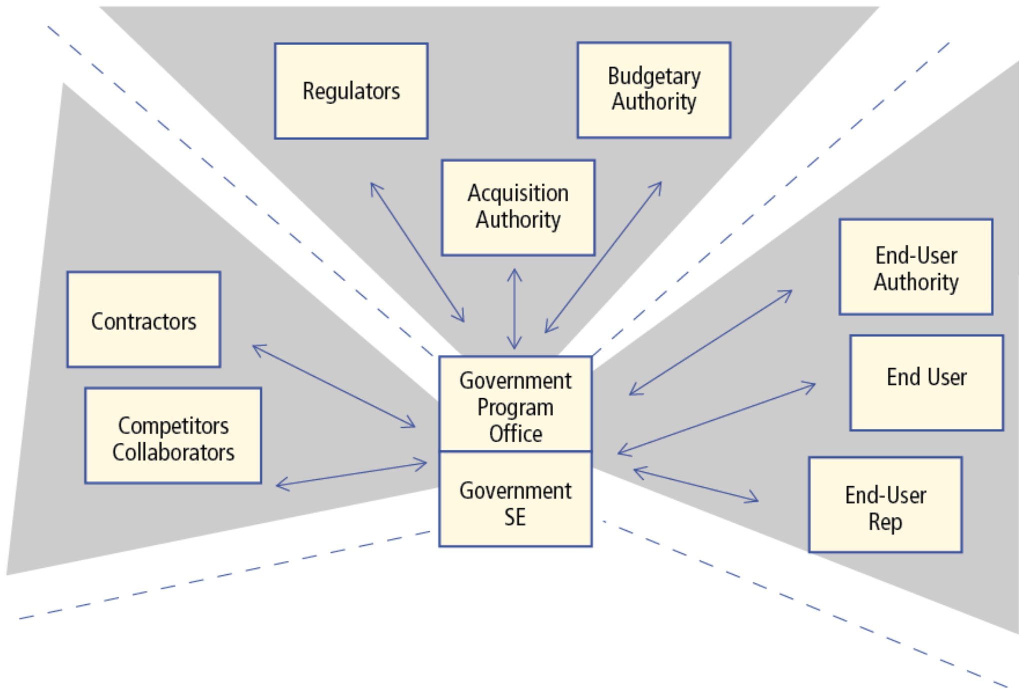


Figure 4-3. Divide and Conquer Pattern

* Chart of Divided Gaul from Vidal-Lablache, *Atlas général d'histoire et de géographie* (1894). From en.wikipedia http://commons.wikimedia.org/wiki/File:Politically_divided_Gaul,_481.jpg

4.3.1 Context of Divide and Conquer Pattern

Too Big to Fail

This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) The context of the Divide and Conquer pattern differs from the Up Close and Personal and Close, But Not Too Close patterns. It applies to a large and expensive program for which huge sums of money are at stake for development (and for current operations if they exist), and failing to deliver is untenable.

Table 4-3. Divide and Conquer Pattern

	Divide and Conquer
Summary	Deals decisively with all stakeholders by dividing them into groups and satisfying each group's interests separately.
Context	• Sub-pattern of Balancing the Supply Web. • Applies to a large and expensive program. • Too big to fail (failing to deliver is untenable).
Problem	• Program office faces powerful stakeholders. • Stakeholders resist solutions that favor others. • Cannot ignore stakeholders.
Forces	• Unwavering imperative to develop the system. • Powerful stakeholders with conflicting views of what should be done have vested interests and make for no easy battles. • Strategy that favors one stakeholder group over others is untenable. • Major effort and considerable resources needed to get stakeholders to understand each other's positions or the enterprise context. • Impossible to proceed if several stakeholders oppose program office plan.
Solution	• Divide and Conquer: Reduce power of resistance by dealing individually with stakeholders. • Battle Preparation: Bring the right talent to bear on the problem. • Understand commander's intent: Establish respected relationship with designated operational authority of users—understand their objectives. • Clear campaign objectives established a clear and consistent requirements baseline. • Contractor Front Campaign: Make contractors successful. • User Front Campaign: Establish credibility and trust with users. • Capture Strategic Positions: Leverage key third parties. • Ensure Logistics Supply: Solicit help of all who will support system. • Powerful stakeholders difficult to negotiate, sometimes middleman can help.
Outcome or Resulting Context	• Successful delivery by independently concentrating on each stakeholder group. • Expended considerable resources dealing with web of powerful stakeholders.

4.3.2 Problem

Powerful Stakeholders
Because of the size, scope, and importance of the system under development, the program office faces powerful stakeholders. These participants in the program are resistant to any solution that they perceive favors meeting others' needs over their own, and they cannot be ignored.

4.3.3 Forces at Work

No Easy Battles
There is an unwavering imperative to develop the system. This may be because of some new national priority or because the existing system is essential and reaching end-of-life. There are powerful stakeholders with large financial and institutional investment in the system under development. They can range from the U.S. Congress or a Department CIO who acts as the enterprise authority to the head of operations for the mission, the users themselves, the development contractors, and the commercial firms who assist in operations. Because of their powerful influences, a strategy that favors one stakeholder group over the others is not tenable. With everyone clamoring for attention and having conflicting views, finding a balance seems challenging, if not impossible. Getting stakeholders to understand each other's positions or the enterprise context will take a major effort and considerable resources. If several of the stakeholders were lined up in opposition to the program office plan, it would be nearly impossible to proceed.

4.3.4 Solution

Divide and Conquer
Successful program offices recognized early that they were surrounded by powerful stakeholders on all fronts. The development would inevitably entail many battles and in some ways would take on the characteristics of a "war." They reasoned that it was not prudent or even possible to deal with all of them all at once. They needed a stakeholder campaign plan. Frequently they decided to take a Divide and Conquer strategy by dealing with each stakeholder group separately and devoting sufficient resources to "win them over." The objective was to get stakeholders to appreciate each other's needs, to understand the larger enterprise context, and to accept the program office's solutions.

Battle Preparation
A consistent key to success with the stakeholders was to establish the credibility of the program office by demonstrating competence. A prerequisite for demonstrating competence was that the program office assembled an internal team of seasoned and talented systems engineers and acquisition experts. Again and again, we saw good systems engineering supported by quantitative analysis carry the day. Preparing for battle helped win the battles.

Understand the "Commander's Intent"
Successful programs recognized leaders in the office of the operational users as important stakeholders. The program office took great pains to establish a respected relationship with this operational authority and to understand their objectives and decision criteria. When the Authority insisted on delivery milestones that exceeded available funding, the program office showed them alternative courses of action shored up by a technically sound analysis. The combination of trusted personal relationships and good systems engineering secured Authority support. The program systems engineers reinforced this cycle of knowledge and support by making sure they

themselves were clear on the operational "commander's intent" and by consistently providing good situational awareness to this Authority.

Clear Campaign Objectives

Successful programs established a requirements baseline that everyone could sign on to. Frequently the requirements exceeded the funds available and were not well understood. Working closely with stakeholders, they prioritized requirements and developed options that fit the budget. It was important not only to invest the time with stakeholders, but also to establish trust through transparency. If everyone knew what the requirements baseline was, even though it changed, the development proceeded more smoothly.

The Contractor Front

Contractors were another important set of stakeholders. For the program to be successful, the contractors had to be successful. Rather than an adversarial relationship between the program and contractor systems engineering teams, each leveraged the other's unique perspective to get the job done. Helping the contractors started at source selection—choosing the right contractor to do the right job. It continued through to acceptance test and deployment. Opposing tendencies, such as the government keeping the contractor in a competitive environment and the contractor seeking profit, were balanced through a working relationship marked by mutual respect. Sometimes the government assumed more responsibility, sometimes the contractor, and sometimes a new contractor was brought in. What seemed to be constant was clarity of who had the responsibility and a commitment on all sides to program success through contractor success.

The User Front

Establishing credibility and trust with the end user was another important front in the Divide and Conquer campaign. This was accomplished through frequent visits to user locations as well as by convening regular user group meetings. By continued attention to addressing user needs and concerns, the program office was better able to enlist user support to move the program forward. The relationships also helped get users to accept needed, new, or changed business rules.

Capturing Strategic Positions

In addition to the major stakeholders who held obvious positions in the acquisition process, interested and influential third parties sometimes also held key strategic positions. Sometimes their influence was based on their historical role in the system. Sometimes they were in a position to act as "go-betweens" in a dispute because they had the "ear" of a recalcitrant stakeholder or because they were trusted and respected by multiple stakeholders. Successful programs identified and leveraged the key positions held by such third parties.

Ensuring Logistics Supply

Large programs tended to have many groups who supplied goods and services to the program. Successful programs went beyond just awarding contracts. They actively solicited the help of all who would support the systems and found roles for stakeholders that were important to the stakeholder and helped ensure broad "ownership" of program success. Some were other commercial firms who would support the actual development. Some would support operations once the system was in the field. Sometimes "sister" organizations within the government would become suppliers. Research outfits from academia and government labs also had a role. These organizations not only had a direct effect on the program, but they also had an indirect influence on other stakeholders through their "back door" relationships.

4.3.5 Examples

The Divide and Conquer strategy expends a great deal of resources individually addressing the concerns of each stakeholder group. A given program didn't necessarily have success on all fronts, but it had enough success to achieve the program objectives. We observed several notable examples in various programs.

Several of the large programs rose from the ashes of failed programs. "Battle Preparation" by bringing to bear the right talent (government systems engineers and contractors) was the key step in reconstituting a program. In one case, even though the program was on the chopping block for three years of poor performance, cancellation was not an option because more than three billion dollars a year were being spent to perform its function. A talented team of systems engineers was assembled under the leadership of an experienced government chief engineer. They rewrote the program plan and schedule and delivered it to the program manager, who was so besieged by stakeholder attacks that he had little time to do it himself. To disentangle stakeholder demands, they also developed a detailed map of roles and responsibilities of each of the stakeholder communities. They were now prepared. We saw this theme repeated in several successful programs.

In several programs, we observed the government chief engineer establish a solid relationship with the acquisition authority, the user authority, and their senior advisors. "Commander's Intent" served as a compass for moving the program forward.

Several programs communicated "Clear Campaign Objectives" by using online tools available to all stakeholders. This included a requirements baseline and a matrix of roles and responsibilities. (See Section 4.5, Role and Responsibility Subnets.) Transparency allowed problems to be addressed early before options became limited.

Careful attention to "Contractor Success" helped many programs succeed. We saw such tools as the government delivering user-approved prototypes or capability reference implementations to the contractor to help clarify what the desired system should look like. (See Section 5.1, Seeing Is Believing.) In one case, the government built a user-guide web site as a prototype to make the goal of this contract deliverable very clear. The contractor's job then became one of making a robust version of it. When the contractor was poorly performing, the government stepped in quickly. In one notable case, they helped work off risk with the contractor on a module-by-module basis. Another successful strategy was to keep the contractors "on their toes" in a competitive environment by prequalifying several of them and then awarding separate task-oriented contracts based on performance and capabilities. One major program also made the explicit leveraging of subcontractors a continuing requirement of the prime contractor. This brought quick response to problems as they cropped up as well as success to the contractor and the program.

On the "User Front" of large programs, it was not unusual for government and contractor systems engineers to share facilities with users. Sometimes it was at the operational location; sometimes facilities were provided for users at the development site. It was common for the government chief engineer to establish and run the users group. Not only did this give the program office the ability to address user questions and concerns, but also it gave the chief engineer the visibility needed to establish a trusting relationship with users. In a program that had been previously plagued by mistrust between the program office and the users, the chief engineer got to know users well and established credibility as a "straight shooter" by telling them the "whole and unvarnished truth." Successful program offices placed great emphasis on satisfied users. That meant delivering functionality that worked. Often it meant encapsulated legacy capability in the new delivery to avoid user "shell shock" from new equipment.

In one reconstituted program, the organization that was to have an important role in operations had previously been responsible for developing the system (and had failed). For both these reasons, they had important knowledge and were recognized as having a "Strategic Position" in the acquisition. The chief engineer invited the key representative of this organization in and listened to everything this person had to say. The chief engineer kept up the dialog until this stakeholder was won over. This organization was also given a lead role on one of the product teams. Without securing the support of this strategic stakeholder, the development might have had a very different outcome.

One of the programs was building a system that would have worldwide reach. During operations of this system, government personnel hired commercial contractors to provide services to be integrated into their mission. To "Ensure Logistics Supply," the program office called in special agents from these service providers for consultation. Likewise they brought in the personnel who would hire these service providers and arranged a facility to house them while the program was proceeding. The combination of developers, users, and suppliers took an enterprise view of the system and was quite successful.

4.3.6 Outcome

The program office successfully delivered the system by independently concentrating on each stakeholder group's interests. They expended considerable resources dealing with a web of powerful stakeholders.

4.4 Circle of Trust

This pattern fosters positive social interactions among stakeholders to improve the willingness of opposing factions to compromise. It is illustrated in Figure 4-4, summarized in Table 4-4, and described in the remainder of the section.

4.4.1 Context of Circle of Trust Pattern

Meeting and Not Meeting Needs

This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) Stakeholders are aware of the different viewpoints and equities that they carry, and they are concerned that other stakeholders needs will be met at the expense of theirs not being met.

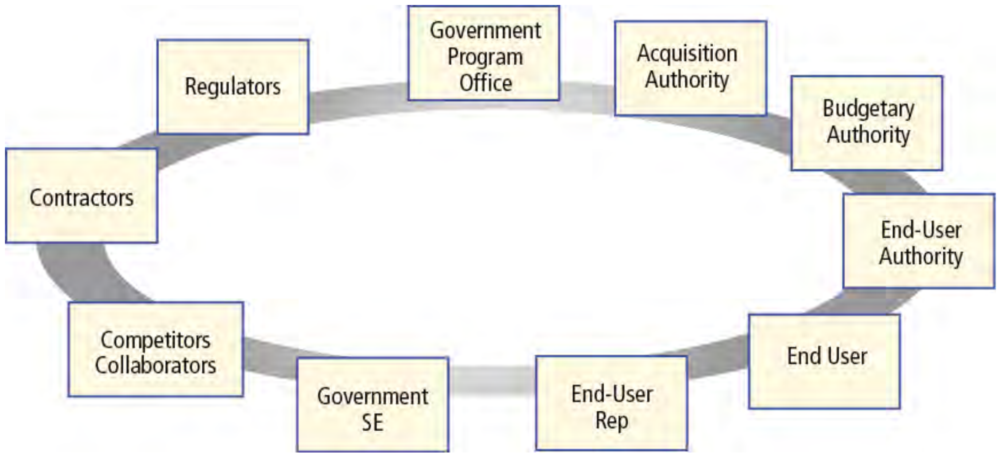


Figure 4-4. Circle of Trust Pattern

4.4.2 Problem

Fostering Cooperation and Compromise
The program office has no obvious way of satisfying all the stakeholders simultaneously. For example, usually there is not enough funding from their acquisition authority to meet all the requirements of their users. Some requirements are unmet. Also, if all the enterprise governance is adhered to, there may not be enough time to complete the work on schedule. Some requirements get delayed. The program office must find a way to influence the stakeholders to cooperate and make compromises for the good of the enterprise as a whole.

4.4.3 Forces at Work

More Trust, Less Risk
In order to trust other members of the supply web, stakeholders make an assessment of the risks of giving that trust. Isolation and lack of situational awareness drive up their perception of their risks and make a decision to trust less likely. Social interaction helps the stakeholders know and appreciate the larger enterprise context, and it also helps the enterprise-level authority better understand local stakeholder circumstances. Positive social interactions serve as a template for solving collective problems. Even sanctions can be positive if administered fairly among stakeholders. Reciprocity and cooperation are more effective means of solving these problems than the imposition of solutions by any single entity. Even with perfect knowledge of all stakeholder circumstances, the interdependencies among stakeholders make viable solutions not obvious. Although trust can be fragile, it enhances cooperation and limits the risk of stakeholders' defecting from collective actions. Its power is that it is self-reinforcing and contagious.

4.4.4 Solution

Transparency and Limited Exposure
Successful programs worked hard at establishing a circle of trust among stakeholders in the supply web. Trust was a by-product of positive social interactions. Trust helped move stakeholders from gridlocked positions to accommodation of enterprise concerns. It built bridges between stakeholders where no previous connection existed. Trust can be established in many ways, but one constant we observed is that it is earned and not mandated. We observed it earned through transparency, so stakeholders always knew what was happening. We also observed it

earned through fairness, so stakeholders never felt anyone was playing favorites. Collective behavior was reinforced and "me first" behavior was met with sanctions. Trust was the sine qua non of harnessing the supply web—it supplied the "social grease" for the "enterprise" skids.

Table 4-4. Circle of Trust Pattern

	Circle of Trust
Summary	Fosters positive social interactions among stakeholders to improve the willingness of opposing factions to compromise.
Context	• Sub-pattern of Balancing the Supply Web. • Program office must sufficiently satisfy a set of enterprise stakeholders. • Stakeholders concerned that other stakeholders' needs will be met at the expense of theirs not being met.
Problem	• Program office has no obvious way to satisfy all stakeholders simultaneously. • Some requirements are unmet, some delayed. • Program office must help influence stakeholders to cooperate and make compromises for the good of the enterprise as a whole.
Forces	• Isolation and lack of situational awareness drive up stakeholder perception of their risks and make decision to trust less likely. • Social interaction helps stakeholders know and appreciate others' contexts. • Positive social interactions serve as template for solving collective problems. • Sanctions can be positive if administered fairly among stakeholders. • Reciprocity and cooperation solve these problems more effectively than imposing any single entity's solutions. • Interdependencies make viable solutions not obvious. • Trust enhances cooperation and limits risk of stakeholders defecting from collective actions. • Trust is self-reinforcing and contagious.
Solution	• Establish circle of trust among stakeholders in supply web as by-product of positive social interactions. • Establish trust through transparency and fairness. • Reinforce collective behavior and impose sanctions against "me first" behavior. • Trust is the sine qua non of harnessing supply web.
Outcome or Resulting Context	• Trust became the means for making compromise and accommodation to find a good enough solution acceptable to all stakeholders. • Benefits were multiplicative—stakeholders helped find and implement solutions.

4.4.5 Examples

Several successful programs established trust with the user community by visiting and spending time in the field observing operations firsthand. The knowledge they gained of operations assured the users that their risk of not getting their needs met was small. Other programs had an experienced operator join the development team.

In other study cases, the government SE team knew as much if not more than the users about their mission. This also instilled confidence that their needs would be met. In one program, the government SE team acted as authority for IV&V test. Here they provided fair, immediate, and written feedback to the program manager and the contractor involved—again this established trust. Another program spent time successfully getting agreement on rules of engagement for the stakeholders: how they would bring technical concerns forward for consideration by the group, how they would operate when the world worked perfectly and when it did not.

4.4.6 Outcome

Trust Brought Progress
Once the circle of trust was established, it became the means for making compromise and accommodation to find a good enough solution acceptable to all stakeholders.

4.5 Role and Responsibility Subnets

This pattern clearly defines subnets within the stakeholder community for each decision or product to be supplied. It is illustrated in Figure 4-5, summarized in Table 4-5, and described in the remainder of the section.

4.5.1 Context of Role and Responsibility Subnets Pattern

Distributed Expertise, Authority, and Responsibility
This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) Many stakeholders are involved in decisions that need to be made. For any given decision, some stakeholders have the information or expertise, some the authority to make the decision, some the responsibility to carry it out, and some are simply affected by the outcome.

4.5.2 Problem

Everyone's Job Is No One's Job
The program office must ensure that all the tasks necessary to complete the acquisition are accomplished on time and with the resources available. Many tasks require the efforts of multiple groups to get to a product or decision point. The multiple stakeholders involved in decisions can lead to inaction or to tasks not getting done. The program office must avoid endless debate as well as not leave appropriate stakeholders out of the decision process. They must also guard against everyone thinking someone else is taking care of things. When something is everyone's job, it is no one's job.

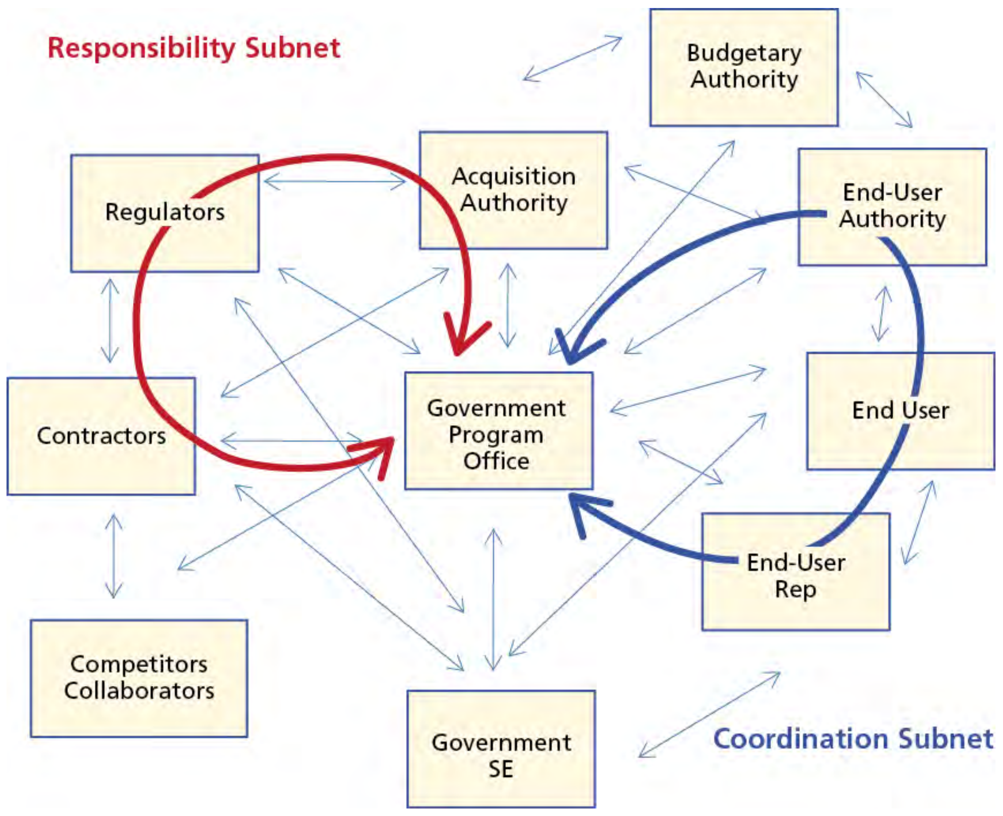


Figure 4-5. Role and Responsibility Subnets Pattern

4.5.3 Forces at Work

From "Too many cooks in the kitchen" to "It's not my job"

Ambiguity in roles and responsibilities causes problems in large projects. This is compounded in an enterprise environment where all the participants are not even in the same organization and there is no single authority that can enforce assignments. When roles and responsibilities are linked in a web, each product or decision point is affected by the roles and responsibilities of the stakeholders. Various people with responsibility have differing views of the right answer. Some stakeholders without authority still want to be consulted or advised of what's happening. Too many conflicting opinions can stall progress. ("Too many cooks in the kitchen.") Autonomous stakeholders sometimes solve the "wrong" problem rather than address a pressing problem. ("It's not my job.") Unfortunately the program office seldom has enough resources to police the entire set of stakeholders for every task that needs to be done.

4.5.4 Solution

Setting Up the Role and Responsibility Subnets

The successful program offices clearly identified the subnet of stakeholders needed to bring a problem to resolution—who had responsibility and authority, who needed to be consulted, who needed to be informed. Essentially they set up a well-defined subnet for each decision point, product delivery, or task. They even went beyond that, setting standards for how those groups were to work together. They established a culture of moving to resolution and relied on the subnets to self-enforce the processes. Not only did these subnets police themselves, but also all the stakeholders were aware of the composition of all the subnets. If they disagreed with what was or was not their responsibility, they could bring it forward for resolution. Although in some respects the roles and responsibilities were "gerrymandered," the resulting clarity produced action instead of paralysis.

Table 4-5. Role and Responsibility Subnets Pattern

	Role and Responsibility Subnets
Summary	Clearly defines subnets within the stakeholder community for each decision or product to be supplied
Context	• Sub-pattern of Balancing the Supply Web. • Multiple stakeholders can be involved in each decision. • Expertise, authority, responsibility, and interest are distributed among stakeholders.
Problem	• The many stakeholders involved in decisions can lead to inaction or to tasks not getting done.
Forces	• Ambiguity in roles and responsibilities leads to delays and even paralysis. • In an enterprise environment, participants are not in the same organization and no single authority can enforce assignments. • Each product or decision point is affected by roles and responsibilities of stakeholders. • People with responsibility have differing views of the right answer. • Those without authority want to be consulted or advised. • "Too many cooks in the kitchen" can stall progress. • "It's not my job" mentality results in unsolved problems.
Solution	• Clearly identify stakeholders for each decision point, product delivery, or task. • Set up well-defined subnets defining who has responsibility, authority, and interest. • Set standards for how groups are to work together and police themselves. • Make all stakeholders aware of all subnets before execution so conflicts can be resolved.
Outcome or Resulting Context	• Clarity breeds success—avoids disharmony and conflict. • Allows stakeholders to self-govern within each of their subnets. • Program office finds and fixes conflicts and holes at subnet level.

4.5.5 Examples

There were many different implementations of the Role and Responsibility Subnets pattern. One program constructed a detailed matrix showing, for each task, which stakeholders were responsible, accountable, to be consulted, and informed (RACI). This was such an important ingredient to their success that it became a major task of the integration contractor to maintain it. It also became the glue that kept the distributed decision making working as a whole. Other implementations shared the attribute of clarity of roles and responsibilities. Even though a subnet of people shared decisions, the clarity of who had what roles in those groups seemed to be a key factor in success.

Some programs partitioned the subnets along organizational lines (e.g., Army, Navy), while others used functional lines (e.g., systems engineering, program management). Others formed an integration council or team to handle the interfaces between groups. In one of the study cases, the operator organizationally reported to the program office; in another, the user and developer worked together side by side in operations and development. In each of these study cases, the subnet acted as a group and had a single interface to the program office.

In some of the study cases, the systems engineering expertise and/or operational knowledge of the government team clearly far exceeded that of the contractors. This devolved into a simple subnet with the program office making most of the decisions and the contractor carrying them out. (See the Architect.org pattern.) However, the clarity of roles and responsibilities kept the program on a successful track.

The self-governance of the subnets also seemed to be an ingredient of the success. The subnets cultivated a culture of positive behavior—“un-do the constraints” and “push toward a solution.” Clarity brought with it accountability. Anyone who wanted to bring a new requirement to the table had to bring new resources, accept a schedule slip, or replace an existing requirement. Within the subnets, people tended to “leave their badges at the door.” This was self-enforced—those who violated the protocol were replaced. One program manager who formed such subnets said it went from “watching paint dry” to “action.”

4.5.6 Outcome

Clarity Breeds Success
Clarity in roles and responsibilities avoids disharmony and conflict. It also allowed the stakeholders to self-govern their process within each of their subnets. The program office was able to find and fix conflicts and holes.

4.6 Seek Secondary Sources

This pattern seeks small flows of resources from secondary sources that potentially have a large impact on the robustness of the program as well as the capability of the system delivered. It is illustrated in Figure 4-6, summarized in Table 4-6, and described in the remainder of the section.

4.6.1 Context of Seek Secondary Sources Pattern

Desires Exceed Resources
This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) Stakeholders often desire a set of capabilities that exceed the resources available. The resources are both dollars and expertise. Nonetheless, as part of the condition for a program to move forward, these desires must be translated into validated system requirements that the program office is obliged to meet.

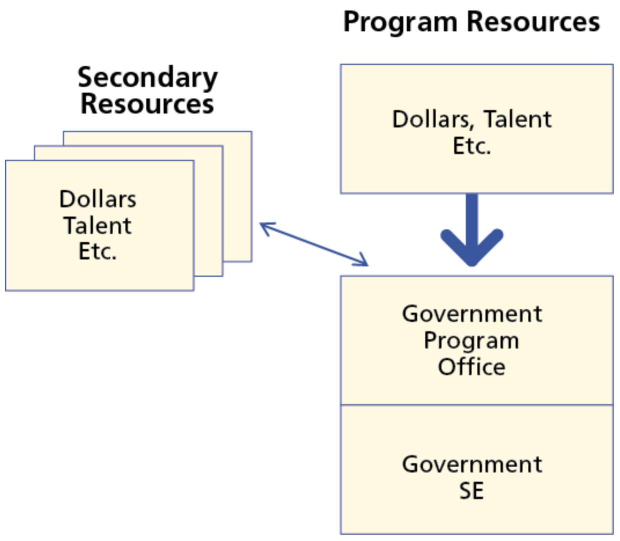


Figure 4-6 (a) Seeking Secondary Sources

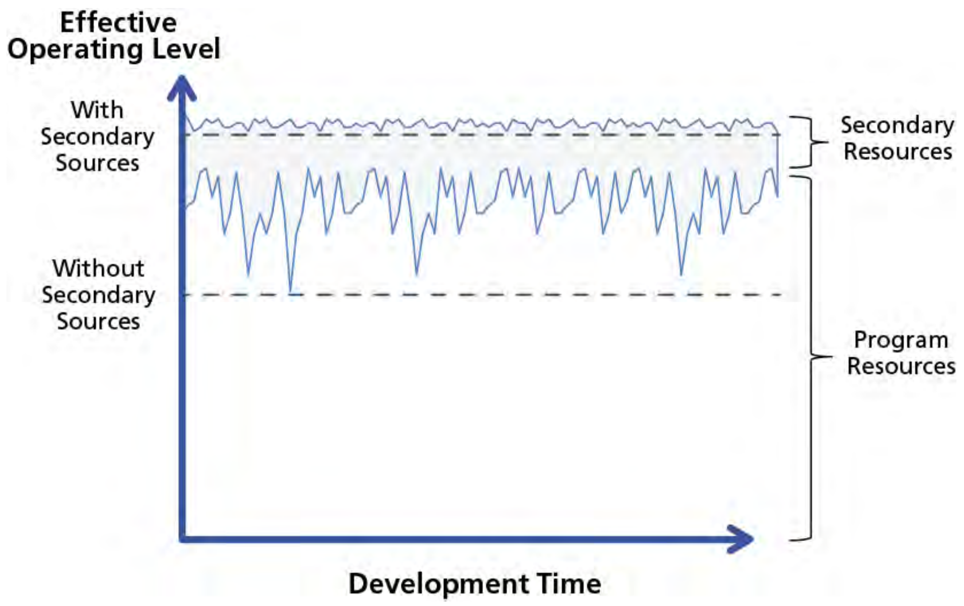


Figure 4-6 (b) Operating With and Without Secondary Sources

4.6.2 Problem

"Check off the Square" or "Fall Below the Line"
When a program office is caught in a resources-requirements mismatch, it may decide to deliver a capability as inferior quality (i.e., "check off the square"), or not deliver it at all (i.e., recognize the requirement as valid, but prioritize it to "fall below the line" in the budget). In an enterprise environment, the former strategy degrades the larger networked enterprise, whereas the latter usually omits those capabilities that most often serve the wider enterprise.

4.6.3 Forces at Work

Small Changes and Large Effects
Programs prioritize their needs against the resources available to them from their traditional primary sources, for example, a funding authority, a contractor, and its local government work force. However, the supply available from these primary sources can vary due to external influence outside the control of the program office or the source itself. If the program is operating with little reserve in its resource pool, these fluctuations have drastic effects. A small amount of additional resource can save the program from these effects and even present the opportunity for the program to flourish from an enterprise viewpoint, for example, by sharing a system capability across the wider enterprise. In a networked enterprise, stakeholders frequently have similar ends or means for accomplishing those ends. (See, for example, Section 5.2, Riding on the Infrastructure.) Although a set of programs may not have enough resources for each one to develop all the desired capabilities, the collective set may.

4.6.4 Solution

Leverage Network of Sources
The flip side of having to satisfy a web of stakeholders is that those stakeholders also represent diverse sources of needed supply, support, and aid. Successful programs tended to be very aggressive in marshaling all the resources available from the enterprise stakeholders. When the system being developed had the potential of satisfying the need of some stakeholder who is not normally a primary source, the program office took advantage of resources from that stakeholder. This smoothed out the bumps in needed resources as well as provided resources for enterprise-level requirements that would otherwise not be met. A similar effect is observed in both food webs and communications networks. Food webs with species having diverse sources of food are more robust against cascading extinction than simpler webs.[19] In fact, some food sources that are not preferred by any species can turn out to be critical to the sustainment of the ecosystem. Communications systems with more diversity in their link transmissions are more robust against fading and loss of messages.[20] Networks are inherently nonlinear. Small changes can have large effects.

4.6.5 Examples

The clearest example of this strategy is in the development of a system that continually accumulated information to conduct large-scale daily operations. Several sister organizations found knowledge of this plan to be useful in their operations; however, the program did not have enough resources to deliver the information to them in a timely fashion. They solicited the help

[19] Jennifer A. Dunne and Richard J. Williams, Cascading extinctions and community collapse in model food webs, *Phil. Trans. R Soc. B*, June 27, 2009; 364(1524): 1711–1723.
[20] P. A. Bello, Binary error probabilities over selectively fading channels containing specular components, *IEEE Trans. Commun. Technology*, vol. COM-14, Aug.. 1966, pp. 400–406.

of funding sources who had a keen interest in enterprise information sharing. On one occasion, they obtained money to deliver the daily plans to the sister organizations (using standardized XML message sets), and on another occasion, they secured funds from a different source to deliver plan updates (via emerging RSS technology). In both cases, the funds obtained were considerably smaller than the total program costs, but the effects were large. Not only did they satisfy enterprise requirements that had fallen below their funding line, but they also became a showcase program for the delivery of enterprise capability.

Other successful programs sought and obtained help from interested parties in commercial industry, academia, national labs, and other programs developing related products. In one case, an existing international system was successfully used as the infrastructure on which a new U.S. system was built. This benefited the program by reducing cost, and benefited the international one by helping establish its infrastructure as an accepted standard.

Table 4-6. Seek Secondary Sources Pattern

	Seek Secondary Sources
Summary	Seeks small flows of resources from secondary sources that have large impact on robustness of program and capability delivered.
Context	• Sub-pattern of Balancing the Supply Web. • Stakeholders desire capabilities that exceed resources. • These desires often become validated system requirements. • Limited program resources from primary sources are inadequate to satisfy all requirements—some that benefit the enterprise end up unfunded.
Problem	• Often primary sources of resources (funding, talent) are inadequate to meet all program requirements. • Performance is degraded or enterprise capability is denied.
Forces	• Programs prioritize needs against resources available to them from traditional primary sources. • Tendency to “check off the square” or prioritize “below the line.” • Tendency to shave resources from mainstream activities to service lower priority items. • Resources available from primary sources vary and fluctuations can have drastic effects on program. • Small amount of additional resource from secondary source can have large effect. • Networked stakeholders frequently have similar ends or means to leverage. • There may not be enough resources in a set of programs for each one to develop all capabilities. • There may be enough resources across the collective set to meet all requirements.
Solution	• Seek and leverage secondary sources in network to fill in gaps. • Smooth out bumps in needed resources as well as provide resources for enterprise-level requirements that would otherwise not be met. • Nonlinearity enables small changes in effort to have large effects in outcome.
Outcome or Resulting Context	• Gaps in primary funding or talent filled by secondary sources. • Program flourishes by going beyond “above-the-line” requirements and by providing enterprise with new, welcomed capabilities.

4.6.6 Outcome

No Slipping Below a Threshold
Actively seeking secondary sources resulted in filling the gaps in needed talent or dollars. It also afforded the program the opportunity to flourish by going beyond requirements and presenting the enterprise with new and welcomed capabilities.

4.7 Network Beats the Node

This pattern takes advantage of relationships in the network of stakeholders to create a resource greater than the sum of the parts. It is illustrated in Figure 4-7, summarized in Table 4-7, and described in the remainder of the section.

Figure 4-7. Network Beats the Node Pattern

4.7.1 Context of Network Beats the Node Pattern

Distributed Resources Across Stakeholder Web
This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) This web has the potential to deliver a great deal of power to the program office. The information and resources in the web containing the program office are greater than those in the program office alone.

4.7.2 Problem

Bringing Collective Intelligence to Bear
The program office must solve several complex and difficult problems during development that could benefit from the expertise available through the network of stakeholders. The program's local government work force and contractor may not have the knowledge needed to solve some of these difficult problems.

4.7.3 Forces at Work

Many Potentially Beneficial Collaborations
Some of the needed information and expertise resides outside the program office and even outside the prime contractor. Subject matter experts (SMEs) exist throughout the stakeholder community. The many interactions among these stakeholders have the potential for finding hidden expertise as well as generating innovation. However, it is natural for the prime contractor to attempt to keep the work and the profits in-house. People also have a natural tendency to reject solutions "not invented here."

Table 4-7. Network Beats the Node Pattern

	Network Beats the Node
Summary	Deliberately takes advantage of relationships in the network of stakeholders to create a resource greater than the sum of the parts.
Context	• Sub-pattern of Balancing the Supply Web. • The information and resources contained in the supply web are greater than those contained in the program office alone. • Supply web can provide additional resources to program office.
Problem	• Program office has complex and difficult problems to solve. • Government work force and contractor together may not have knowledge needed to solve these difficult problems. • Solutions can benefit from expertise available in supply network.
Forces	• Many potentially beneficial collaborations. • Some needed information/expertise resides outside program office and even outside prime contractor. • Subject matter experts (SMEs) exist throughout stakeholder community. • Interactions between stakeholders find expertise, and generate innovation. • Prime contractor tends to keep work in house. • People have natural tendency to reject solutions “not invented here.”
Solution	• Set up acquisition to harness the power in the supply web. • Proactively set up or stimulate collaborative groups of diverse stakeholders. • Encourage contractor to engage the right subs at the right times. • Favor contractors highly leveraged with appropriate subs. • Form coalitions with stakeholders whose self-interest matches the program’s interest.
Outcome or Resulting Context	• The collective knowledge found and harnessed in the supply web results in a better product than would have been otherwise possible. • Diversity keeps out biases and spawns innovation.

4.7.4 Solution

Harness the Power of the Supply Web

Successful program offices tended to set up their acquisition to harness the power of the supply web. This is a variation of the Seek Secondary Sources pattern (see Section 4.6) in which collaborations may go beyond a second source—it refers to the power of the web. The program offices were proactive in setting up or stimulating collaborative groups of diverse stakeholders. The number of synergistic relationships that can be formed with such a web far exceeds the number of pair-wise relationships that the program office can forge with its contractors. When a stakeholder’s self-interest matched the program’s interest, their SMEs would frequently form a synergistic coalition. Harnessing the power in the supply web is akin to the use of collective intelligence discussed by Surowiecki in *The Wisdom of the Crowds.*[21]

[21] J. Surowiecki, *The Wisdom of the Crowds*, New York: Anchor Books, 2004.

4.7.5 Examples

One of the successful strategies observed in multiple programs was that the program office favored awarding the development contract to bidders who were highly leveraged with appropriate subcontractors. Once the contract was awarded, they encouraged their prime contractor to continue to engage the right subcontractors at the right times throughout the program. The right subcontractor can quickly bring a prime contractor up the learning curve or can quickly react to new problems. Because a contractor can engage a subcontractor much more quickly than the government, new problems can be solved in a timely manner. The acquisition became agile. However, to do this, the prime contractor must be skilled in subcontractor management. Otherwise there is a risk of assuming the inefficiencies of a distributed work effort and greater integration difficulty and not reaping the expected benefits.

Another successful strategy observed was the bringing in of appropriate SMEs to advise the program office. In one program, the SMEs were industrial companies that would benefit from operations once the new system was deployed. These companies were willing to lend their considerable expertise for the potential of future business.

One program convened a technical group from diverse stakeholders. The group established ground rules regarding the fair assessment of technical problems and solutions. The technical biases of the various members tended to cancel one another out, yielding better solutions than any one individual provided. Similarly, another program established a successful national team that they viewed as a “Manhattan Project” approach.

4.7.6 Outcome

Collective Knowledge and Unbiased Solutions
The collective knowledge found in the supply web results in a better product than would otherwise be possible. Diversity keeps out biases and spawns innovation.

4.8 Top Cover

This pattern uses informed acquisition authorities and capitalizes on them to shape the stakeholder environment. It is illustrated in Figure 4-8, summarized in Table 4-8, and described in the remainder of the section.

4.8.1 Context of Top Cover Pattern

Strings Attached
This pattern is a sub-pattern of Balancing the Supply Web. The program office is immersed in a supply web, with all that entails. (See Section 4.1.1.) Stakeholders in the supply web are inextricably tied to the program office. Each imposes potentially incompatible constraints on a program office. Likewise, each program office decision in a course of action (COA) has potential consequences for the stakeholders.

4.8.2 Problem

Loosening the Ties that Bind
The complex stakeholder environment can make it difficult to act decisively at each stage of the development. Each time a COA is met with resistance by any one of the stakeholders, the program office could easily be impeded from continuing. The program office must somehow

influence the stakeholders to offer less resistance so they can act quickly in response to new information or circumstances. Without loosening the ties that bind, the program has difficulty finding a path forward and it could become paralyzed.

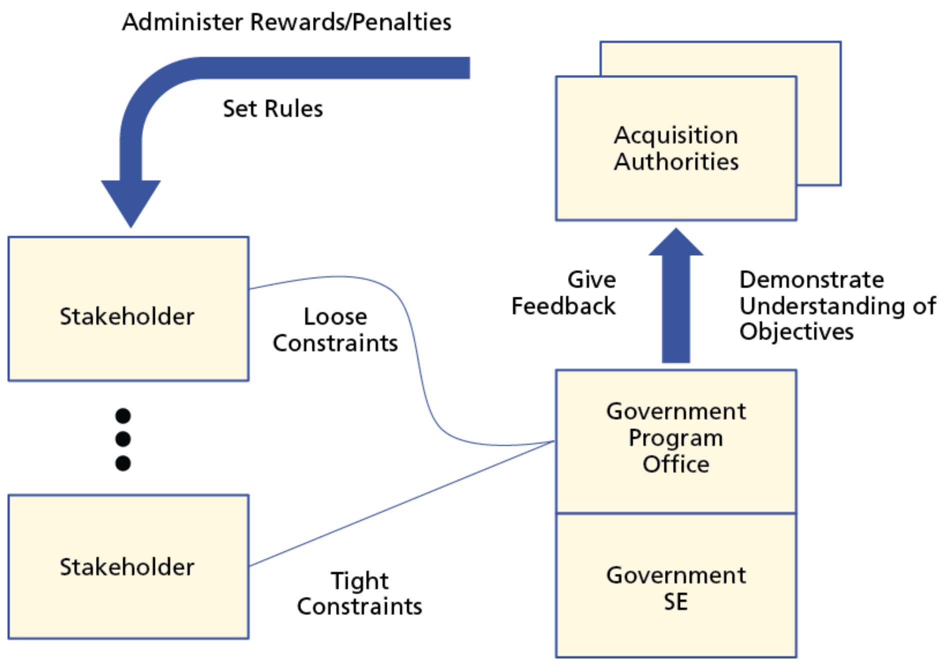


Figure 4-8. Top Cover Pattern

4.8.3 Forces at Work

Conflict and Control

Program offices have a certain amount of slack or leeway in which to operate without constantly seeking approvals from stakeholders. As the system development proceeds, the program office must adjust the program to suit the changing conditions (e.g., in funding, requirements, or technology). Predictably, stakeholders tend to act in their own and their organization's self-interest, fulfilling their legitimate roles in the enterprise acquisition process. Stakeholders can not only disagree on a COA (the means) for achieving some desired goals (the ends), but also on the goals themselves. Unchecked, the ability of stakeholders to impede a COA can easily paralyze a program. If a program office ignores the conflicts, they tend to escalate and stakeholders become entrenched. Strong authority figures in the enterprise have the ability to diminish these conflicts. They influence stakeholder behavior by defining the rules and administering payoffs and penalties. Stakeholders "play the game" according to how it is scored. Sometimes, top cover is more subtly achieved via real or perceived stature, stakeholder skill in implying top-level support, or the expected future payoff from maintaining a good relationship with a powerful authority.

Table 4-8. Top Cover Pattern

	Top Cover
Summary	Uses informed acquisition authorities to shape the stakeholder environment.
Context	• Sub-pattern of Balancing the Supply Web. • Stakeholders impose potentially incompatible constraints. • Program office decision in a course of action (COA) has potential consequences for stakeholders.
Problem	• Complex stakeholder environment with many interdependencies. • Difficult to act decisively when met with stakeholder resistance. • Unbridled ability of stakeholders to impede a COA can paralyze program.
Forces	• Program office has some leeway in which to operate without constantly seeking stakeholder approval. • Program office must adjust program to suit changing conditions (e.g., in funding, requirements or technology). • Stakeholders act in their own and their organization's self-interest. • Ignored conflicts tend to escalate and stakeholders become entrenched. • Higher acquisition authorities influence stakeholder behavior by defining rules and administering payoffs and penalties. • Stakeholders "play the game" according to how it is scored.
Solution	• Gain confidence of higher acquisition authorities. • Provide acquisition authority with feedback so "top cover" is informed. • Supply stakeholders with immediate benefits.
Outcome or Resulting Context	Program office was able to act quickly and decisively as program and enterprise conditions changed.

4.8.4 Solution

Shape Stakeholder Environment

Successful programs leveraged higher authorities in the larger acquisition system to obtain the freedom of action they needed to solve problems as they arose. They generally accomplished this in three ways:

- Gain the confidence of the higher acquisition authorities.
- Provide higher acquisition authorities with feedback so their "top cover" was informed.
- Supply other stakeholders with some immediate benefits to gain their support.

By establishing a strong relationship with one or more authorities in the acquisition process, the program office obtained some leeway in their development and acquisition. To secure the trust of authorities, they demonstrated deep knowledge of the operations for which the system was intended and strong evidence (technical and otherwise) of why one solution is to be preferred over another. The key to gaining their confidence was to demonstrate that they understood the authorities' goals. They got close enough to the acquisition authorities to learn what they value and what criteria they used to make decisions. Once the program office established the trust of the higher authorities, they were able to provide them with viable alternatives as well as feedback regarding other stakeholders. This "informed top cover" allowed acquisition authorities to set rules and establish rewards and penalties that generally decreased stakeholder resistance. They did such things as separate funding, establish clear roles and responsibilities, and control expectations. Top cover also shaped the objectives of the stakeholders by discouraging behavior that was considered detrimental to the enterprise as a whole.

Successful program offices also paid close attention to stakeholder needs. Recognizing that stakeholders needed to agree with or accommodate the way forward, they made some effort to "grease the skids" with them by building confidence and trust. They generally made a concerted effort to deliver some immediate value to stakeholders. Informed top cover shaped stakeholder behavior, and program office interactions reinforced it.

This pattern is similar to the Close, But Not Too Close pattern discussed in Section 4.2. However, it differs in emphasis. That pattern focuses on disengaging with the operational user and diverting resources to the User Authority, who in turn, levies sanctions against the non-compliant users. Both patterns share the idea of compensatory deliveries to the users, which can reduce pushback to even higher authority direction.

4.8.5 Examples

In each case, top cover was a strong force to bring about freedom of action. When requirements exceeded funds available, one program was able to influence the top cover authority to accept alternative COAs. When the top cover authority agreed, the rest of the stakeholders fell into line. Another program with a critical mission and a technology challenge was granted a waiver by the proper authorities to acquire the system without following the long-standing acquisition regulations. They leveraged this waiver and used information technology as the "glue" that coupled systems together. They did it in a way that decreased dependencies (interactions) between systems and therefore stakeholders. Another successful program moved quickly under its top cover but always made sure to deliver compensatory capabilities to reduce resistance from stakeholders.

4.8.6 Outcome

Agility Achieved
Successful program offices exploited the freedom of action to find a path through the difficult terrain of the system development process. They earned their freedom of action "spurs" by gaining support of the right top cover, building trusted personal relationships, producing viable COAs, and delivering on promises.

5 Harnessing Technical Complexity Sub-Patterns

The Harnessing Technical Complexity pattern discussed in Section 3.2 addresses the large number of possible combinations of systems and components that can be brought together in an information enterprise and the resulting interdependencies among those components. The complexity of the information enterprise is a mixed blessing. It is precisely this complexity that makes the enterprise capable of accomplishing unprecedented work. The current N-tier architecture of the Internet and the World Wide Web provides a successful mechanism for harnessing that complexity and taking advantage of its strong points. Whereas Section 4 focused on the interdependence of the stakeholders in the enterprise, Section 5 focuses on that of the technologies.

We observed several recurring strategies in the programs we interviewed and captured them as design patterns within the Harnessing Technical Complexity pattern:

- ***Seeing Is Believing***—builds a capability reference implementation for the enterprise that shows what can be done, how it works, and what it should do when done.
- ***Riding on the Infrastructure***—builds new capabilities on top of the existing infrastructure.
- ***Loose Couplers***—establishes isolation between layers and integration across the enterprise.
- ***Social and Technical Alignment***—aligns people, processes, and technologies to match development and acquisition to the enterprise structure.
- ***Plan to Re-plan***—stimulates desired behavior through feedback and incentives, and then learns from results what behavior is desired next.
- ***Technology Surfing***—uses an ongoing process of identifying new and emerging technologies, experimenting with them, and integrating what works into the evolving enterprise—"catch the next technology wave" rather than "create or wait for the big one."
- ***Architect.org***—the government program office team assumes full responsibility for architecting and overseeing development of the system capability.

The capability reference model used in the first pattern helps cut through complexity not only by presenting a proof-of-concept prototype, but also by establishing a baseline from which users, contractors, and other system developers can work. Building new capabilities from infrastructure and loosely coupled components (as in the next two patterns) imitates the WWW architecture that has been so successful in harnessing complexity. The Social and Technical Alignment pattern leverages all these patterns by structuring the acquisition to match the technical architecture. The next two patterns emphasize an evolutionary approach that specifically takes advantage of experimentation and learning for re-planning and technology insertion. The final pattern, Architect.org, uses a government architect to manage the system development from inception to operation in such a way that it adheres to the enterprise context. Each of these patterns is discussed separately in the following sections.

5.1 Seeing Is Believing

This pattern builds a capability reference implementation for the enterprise to show what can be done, how it works, and what it should do when done. It is illustrated in Figure 5-1, summarized in Table 5-1, and described in the remainder of the section.

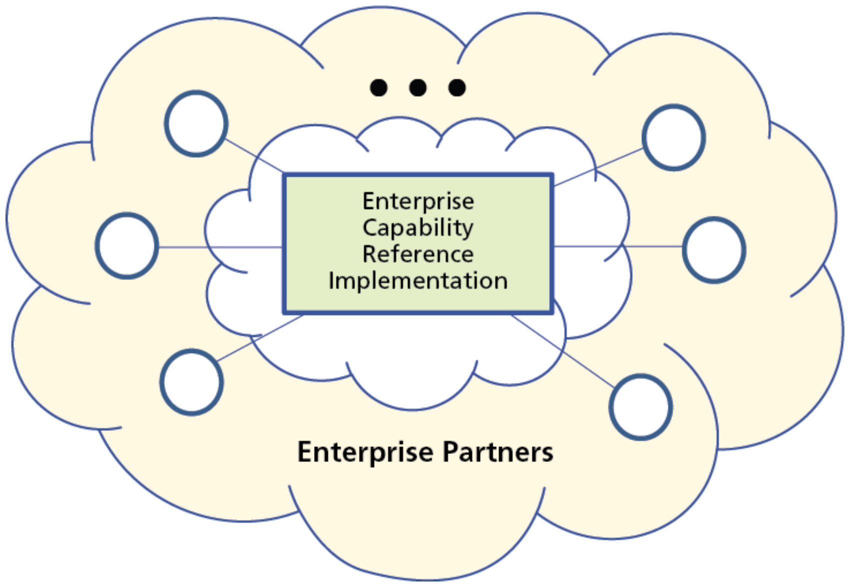


Figure 5.1. Seeing Is Believing Pattern

5.1.1 Context of Seeing Is Believing Pattern

We All See the Elephant Differently
This pattern is a sub-pattern of Harnessing Technical Complexity. The context of the Seeing Is Believing pattern is that an information enterprise is made up of diverse systems, many of which have already been developed or have their own separate development tracks. The stakeholders involved in these various systems may have different perceptions of what a new enterprise-wide capability will look like, and this is reflected in their system implementations. Whether this capability has become desirable because a new technology is maturing or a new need to share work across the enterprise has surfaced, there is no reason to assume the cooperating systems will be compatible.

5.1.2 Problem

Building the "Right" System
Risk mitigation is traditionally focused on building a system or component correctly according to some specification. In a large and complex enterprise, there is the risk that the specification is either misstated by the government or misinterpreted by the contractor. Users have rejected many well-built and functioning systems because they don't fit into the larger enterprise context. The program office must ensure that they not only build their system "right," but also that they build the "right" system.

Table 5-1. Seeing Is Believing Pattern

	Seeing Is Believing
Summary	Building a capability reference implementation for the enterprise shows what can be done, how it works, and what it should do when done.
Context	• Sub-pattern of Harnessing Technical Complexity. • Diverse systems—already developed or on development tracks. • Stakeholders' perceptions reflected in different implementations. • Can't assume compatibility of cooperating systems.
Problem	• Risk that specification is either misstated by government or misinterpreted by contractor. • System works but doesn't fit into larger enterprise context. • Build system "right," but also build the "right" system.
Forces	• Different stakeholders are likely to have conflicting perceptions of what implementation should be. • Stakeholders across enterprise may not view the new capability as useful or even believable. • Implementations may not fit into cost/schedule/performance profiles of existing programs. • Specification ambiguities may result in implementations within diverse programs not being interoperable. • Ambiguity results from users not having a clear view of requirements when they have little experience with technology possibilities. • Contractors increase the price of developing an unprecedented system as a hedge against high risk.
Solution	• Government - Develops a working Capability Reference Model that roughly approximates new capability or new technology. - Demonstrates capability in user forum (e.g., exercises) to prove system is feasible and verify its value to users. - Removes ambiguity by supplying reference implementation as baseline to contractor. - Allows parallel developments by enterprise partners developing separate but cooperating systems. - Encourages non-proprietary solutions. - Establishes a minimum acceptable standard for contractors to meet.
Outcome or Resulting Context	• Helps users better define requirements. • Gives contractors a clearer picture of what requirements were. • Cost reduced by shifting risk from contractor to government. • Enhanced interoperability. • Allows concurrent developments cooperating systems.

5.1.3 Forces at Work

Conflicting Perceptions

Implementation of either new technologies or new workflows by themselves is inherently risky. When an implementation of either one is embedded in an enterprise environment, different stakeholders are likely to have different perceptions of what the implementation should be. This can result in such things as incompatible formats for data and messages, different implementations of new technologies, and even conflicting workflow patterns.

At the front end of a program, stakeholders across the enterprise may not view the new capability as useful or even believable. Additionally, implementing the new capability may not fit into the cost/schedule/performance profiles of existing programs. Even if all these hurdles were overcome, ambiguities in the specification may result in the implementations within diverse programs not being interoperable. Ambiguity can result from the users not having a clear view of the requirements because they have little experience with the technology possibilities. Generally, contractors increase the price of developing an unprecedented system as a hedge against the high risk.

5.1.4 Solution

A Capability Reference Implementation
If the government has the technical capacity to develop an operational prototype, it can shift the risk away from the contractor and save money for the program. That also encourages non-proprietary solutions because the government has no preferred product to sell. A software implementation is used to increase awareness and familiarization of a new capability across the enterprise. It serves first as an operational definition of the new capability. The users provide feedback until the final implementation suits them. (Requirements are better known after a system is produced than before.) It is also the basis of more accurate cost, schedule, and performance estimates. When it comes time to produce the operational version, not only is ambiguity removed from the specification (because there is a working example in hand), but also enterprise partners developing separate but cooperating systems can pursue parallel developments. Generally the capability reference implementation is made available to the contractors free of charge in the form of a Capability Reference Model containing the software code, documentation, and development/deployment tools. It establishes a minimum acceptable standard that the contractors must meet. They can deliver more, but certainly not less.

5.1.5 Examples

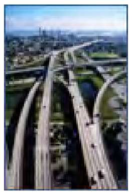
Several of the programs used Capability Reference Models to develop enterprise-wide solutions. One implemented a burgeoning Internet information exchange technology to share information across international systems. It “sold” the capability to users, to funding agents, and to enterprise partners using the credibility gained from the reference implementation. It even obtained funding from outside its own program office because of that credibility. (See the Seek Secondary Sources Pattern in Section 4.6.) It then turned that over to the contractors to build industrial strength versions. Another program built a portable reference model that manifested both the technical requirements and the operational procedures and brought it to its several customer locations worldwide. Once the requirements and procedures were wrung out and accepted by these users, the portable reference model was turned over to contractors for development.

5.1.6 Outcome

Risk Shifted, Interoperability Enhanced
Cost was reduced by shifting the technical risk from the contractor to the government. The Capability Reference Implementation helped clarify the requirements for the users and the expectations for the contractor. Cost and schedule risks were minimized through the experience of building the reference implementation. Interoperability was enhanced because the reference

implementation served as a standard across the enterprise and allowed concurrent developments of cooperating systems by separate organizations.

5.2 Riding on the Infrastructure



This pattern builds new capabilities on top of the existing infrastructure. It is illustrated in Figure 5-2, summarized in Table 5-2, and described in the remainder of the section.

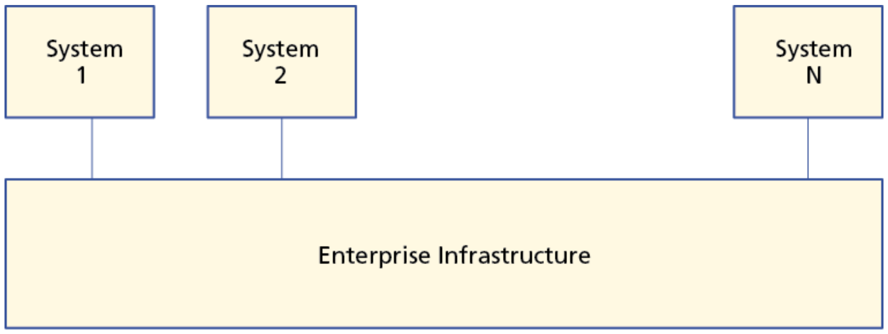


Figure 5.2 Riding on the Infrastructure Pattern

5.2.1 Context of Riding on the Infrastructure Pattern

Infrastructure Supports New Development
This pattern is a sub-pattern of Harnessing Technical Complexity. The World Wide Web (WWW) infrastructure (consisting of the Internet, Browsers, standard protocols, etc.) has become a pervasive model of how myriad organizations and individual users participate in an endless variety of cooperating workflows. To the extent that a government enterprise leverages an information infrastructure, (typically like the WWW), new systems being developed must “ride on the infrastructure” to deliver their capability to that enterprise. This is akin to the way in which the economic infrastructure (e.g., banks, money supply, regulations) and the transportation infrastructure (roads, railways, airports, etc.) support new activities in those domains. Development of new activity is supported by infrastructure.

5.2.2 Problem

Uncertainty Increases Risk, Infrastructure Adds Constraints
The program office must decide when and how to reuse infrastructure, to modify it, and to build new infrastructure components. Uncertainty in the infrastructure increases risk to the program, and the infrastructure itself constrains the design.

5.2.3 Forces at Work

Infrastructure Benefits Outweigh Costs
Infrastructure saves the cost of building such components over and over again, and it also promotes enterprise interoperability. Because infrastructure is shared by large numbers of enterprise users, it yields great benefits in cost, schedule, interoperability, maturity of design, and so on. However, the needed infrastructure components may not yet exist—they may be in

development by another program or they may need to be developed by this program. Even if they exist, they may not be available to this particular program. That may be for security reasons or simply because there is no sharing mechanism (e.g., cost sharing or service level agreements). Due to rapidly changing technology, existing infrastructure components may become obsolete and abruptly be replaced by new ones. Cost and schedule constraints so heavily favor a program office incorporating infrastructure that these problems must be overcome.

Table 5-2. Riding on the Infrastructure Pattern

	Riding on the Infrastructure
Summary	New capabilities are built on top of the existing infrastructure.
Context	• Sub-pattern of Harnessing Technical Complexity. • Internet and WWW infrastructure is a pervasive model of how users participate in endless variety of cooperating workflows. • Akin to how economic and transportation infrastructures support new activities. • Infrastructure supports development of new activity.
Problem	• Program office must decide when and how to reuse infrastructure, to modify it, and to build new infrastructure components. • Infrastructure uncertainty adds risk to program. • Infrastructure adds constraints to design.
Forces	• Infrastructure saves cost of building such components over and over. • Infrastructure promotes enterprise interoperability. • Needed infrastructure components may either not yet exist or not be available to a particular program, (e.g., for security because no sharing mechanism exists). • Infrastructure components may become obsolete and abruptly be replaced by new ones.
Solution	• Use pervasive standards and loosely coupled architecture. • Keep applications developers "honest," that is, not changing infrastructure as short cut to achieving capability. • Keep infrastructure developers focused on enterprise-wide needs.
Outcome or Resulting Context	• Infrastructure is a sine qua non of harnessing technical complexity. • As the infrastructure evolves, it makes adding new applications faster and cheaper.

5.2.4 Solution

Use Pervasive Standards and Loosely Coupled Architecture

We observed programs that only developed infrastructure, programs that only developed applications to ride on infrastructure, and programs that developed both. The solution is twofold: build the infrastructure with pervasive technologies and build the whole enterprise using a change-tolerant architecture. For example, a network communications infrastructure is fairly well defined by Internet standards and is relatively safe to be leveraged. The transition to version 6 of the Internet Protocol will cause some disruption, but then it will return to a "safe bet." Likewise several higher level protocols used in the World Wide Web are fairly stable. Where performance of government systems requires features that go beyond those found in commercial standards, modifications are made that still leverage the pervasiveness of these standards. To guard against uncertainties associated with emerging or competing standards and protocols, enterprises are built with a layered architecture. Special consideration is given to the layer interfaces. As long as the layer interfaces are adhered to, variations (due to uncertainty or innovation) can be tolerated within the layers. Moreover, changes at one layer or layer interface do not ripple through the other layers.

5.2.5 Examples

A few case study programs did not use the Internet and WWW standards. There were some modifications for security or performance, but overwhelmingly infrastructure came from those pervasive commercial standards. The DoD’s development of SIPRNET and NIPRNET are two historical examples that preceded this study.

Two of the most successful cases we studied used a strategy in which the program that developed the infrastructure was kept separate from the programs that developed the applications. The resulting infrastructure became very “pure” in the sense that it had high utility to a large number of disparate developers and, in turn, to their users.

Another program charged with developing both infrastructure and applications used this same “separation” principle. They went to great lengths to isolate contracts for building the infrastructure components from those of the mission modules. Again, this was done to ensure that the integrity of the infrastructure wasn’t compromised to suit one particular application at the expense of any others.

Some of the programs used the simplicity of implementing each new application as a measure of quality of infrastructure. This became a learning loop to improve infrastructure. (See Plan to Re-plan pattern in Section 5.5.)

Many programs used a service-oriented architecture that allowed loose-coupling of new applications (services) to existing infrastructure.

One variation of the Riding on the Infrastructure pattern was observed when the program office was faced with integrating several large and expensive existing systems into a coherent and cooperating enterprise. Even if the powerful system stakeholders could have been convinced to engineer interfaces with each other’s systems, the cost would have been prohibitive. Instead, the program office designed a tailor-made infrastructure that acted as “glue” between the systems and assumed the bulk of the burden of integration. Each system had only to build an appliqué that acted as a single interface to this specially created infrastructure. Thus, rather than build the systems as “riding on the infrastructure,” they “jacked up” the systems “slipping the infrastructure under.” This approach was minimally invasive to the existing systems and turned out to be cost effective, as well.

5.2.6 Outcome

In each case, infrastructure was the sine qua non of the enterprise solution. The addition of new applications became faster and cheaper as the development and the infrastructure matured.

5.3 Loose Couplers

This pattern establishes isolation between layers and integration agreements across the enterprise. It is illustrated in Figure 5-3, summarized in Table 5-3, and described in the remainder of the section.

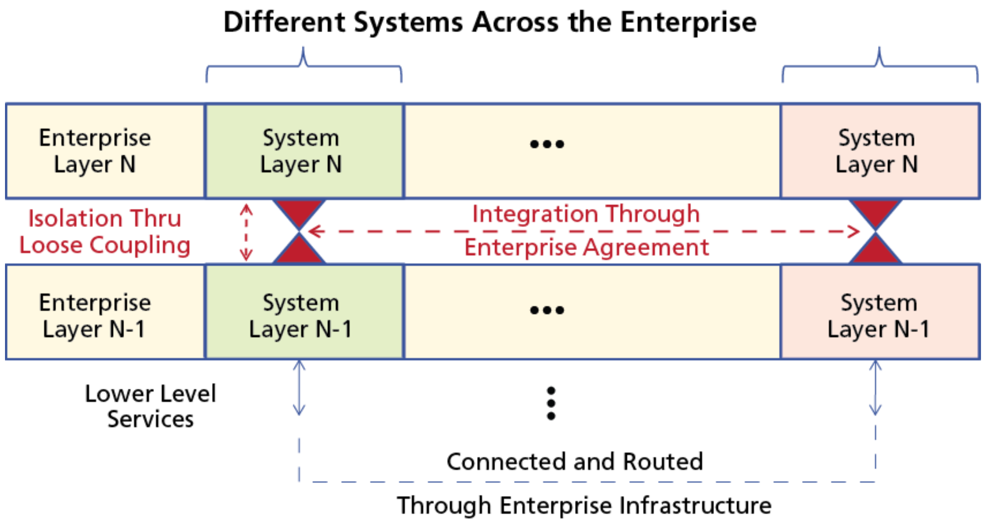


Figure 5-3. Loose Couplers Pattern

5.3.1 Context of Loose Couplers Pattern

Loose Couplers Organize an Enterprise

This pattern is a sub-pattern of Harnessing Technical Complexity. Couplings between components in a system and between systems in an enterprise affect the character of that enterprise, for example, its stability, agility, efficiency, and effectiveness. The most common variety of information enterprise encountered in the study cases consisted of systems partitioned into hierarchical layers loosely coupled to one another. In addition, these loose couplers were standardized across the enterprise and the enterprise had mechanisms to route service requests across the enterprise.

5.3.2 Problem

Choosing the "Right" Loose Couplers

The coupling in an information enterprise has two dimensions: the (vertical) integration of modules that make up a single system and the (horizontal) interoperability of modules that produce capabilities across the enterprise. The way in which the coupling is done determines the extent to which they operate as a coherent whole or as a collection of stovepipe systems. Coupling also determines the agility of the enterprise, that is, the degree to which it can easily and quickly adapt to change. Without standardizing the way in which some of these interactions are implemented across the enterprise, the number of different implementations becomes too large to manage. A program office has to adopt the “right” coupling implementations in its system design without the ability to control the rest of the enterprise.

Table 5-3. Loose Couplers Pattern

	Loose Couplers
Summary	Establish isolation between layers and integration across the enterprise.
Context	• Sub-pattern of Harnessing Technical Complexity. • Most common variety of information enterprise encountered consisted of systems partitioned into hierarchical layers loosely coupled to one another and standardized across the enterprise with mechanisms to route service requests. • Many protocols are so pervasive they become de facto standards. • Many interfaces are proprietary or not used by the bulk of the enterprise. • Couplings between components in a system and between systems in an enterprise affect the character of the enterprise.
Problem	• Couplings between components and systems affect the character of the enterprise, e.g., its stability, agility, efficiency, effectiveness. • Program must adopt the “right” coupling implementation to promote easy integration and interoperability—without the ability to control the enterprise.
Forces	• Coupling in an information enterprise has two dimensions—vertical integration of modules in a system and horizontal interoperability of modules to produce an enterprise capability. • “Looseness” of coupling determines enterprise agility, i.e., the degree to which it can easily and quickly adapt to change (i.e., isolation allows economical change). • Number of potentially different interfaces grows exponentially with enterprise size. • Non-standardized coupling becomes too large to manage. • Some enterprise parts are not layered. • Some top-down directed (government) enterprise standards are not universally accepted or used. • Industrial contractors attempt to establish their products as de facto standards. • Consortia come together to establish widely accepted standards.
Solution	• Seek out de facto standards. • Identify interfaces and protocols associated with preponderance of enterprise workflow. • Advocate these interfaces and protocols as potential de facto standards for the enterprise. • Work with peer organizations, sometimes in community of interest groups, and regulatory bodies to agree on which can become de jure standards. • Legacy systems employ conversion/translation approaches.
Outcome or Resulting Context	• Programs achieved a great deal of interoperability with a small number of loose coupler standards. • Followed 80/20 Rule: 80% of interoperability from agreement on 20% of interfaces.

5.3.3 Forces at Work

Mandates and Common Practice
Well-designed systems are modularized to prevent changes in one module from rippling through to require changes in neighboring modules. This is done by fixing the interfaces. Well-designed information enterprises allow capabilities inherent in one system to be mixed and matched with capabilities in others—more interfaces. The number of potentially different interfaces grows exponentially with the size of the enterprise. Two forces are at work combating this problem. Regulators establish top-down directed enterprise standards, and commercial industry is busy attempting to establish their products as pervasive de facto standards. In some cases, industry consortia (e.g., WWW Consortia, Open Systems Group) come together to establish widely accepted standards.

5.3.4 Solution

Observe and Advocate
Usually a small number of workflow elements accounted for a large fraction of the total workflow in the enterprise, for example, the routing of packets in the enterprise communications network. The interfaces and protocols associated with these elements were potential de facto standards for the enterprise. The successful program office worked with peers across the enterprise, sometimes in Community of Interest user groups, as well as regulatory bodies to agree on which can become de jure standards.

5.3.5 Examples

The majority of the study cases used Internet standards to build their enterprise-wide communications network (TCP/IP). Where legacy systems used other means within their own operations, they added a conversion to Internet standards to allow cooperation across the enterprise. At the higher level protocol layers, they relied on emergent WWW standards (e.g., XML, RSS, and more sophisticated web services) to provide loose coupling. At the data and message exchange levels, they sought out the most frequently used data elements, like those describing "what," "when," and "where," and then constructed brief XML-based messages that became enterprise standards. The larger and more ambitious programs convened Community of Interest User Groups to come to agreement on data and message models.

5.3.6 Outcome

80/20 Rules
Programs achieved a great deal of interoperability for the price of instituting a small number of loose coupler standards. Typically the coupling followed an 80/20 Rule—80 percent of the interoperability work could be accomplished with agreement on only 20 percent of the interfaces. The commercial marketplace provided easy choices for communications and routing (TCP/IP) and simple but pervasive standards like RSS were also relatively easy to "sell" to enterprise partners. The process of developing more extensive data and message standards yielded previously unachievable interoperability, but did so at a higher cost in labor to gain enterprise agreements.

5.4 Social and Technical Alignment

This pattern seeks alignment of people, processes, and technologies to match development and acquisition structure to the enterprise structure. It is illustrated in Figure 5-4, summarized in Table 5-4, and described in the remainder of the section.

5.4.1 Context of Social and Technical Alignment Pattern

The Acquisition System Is a Complex Enterprise
This pattern is a sub-pattern of Harnessing Technical Complexity. Managing a program that acquires a new system for use in a complex enterprise is itself a complex undertaking. Just as the systems engineering effort is immersed in an enterprise environment, so is the management effort. Multiple funding sources, contracts, and teams of people are frequently working on the acquisition—and not all are under the control of the program office. Their interactions can produce unanticipated consequences, both positive and negative.

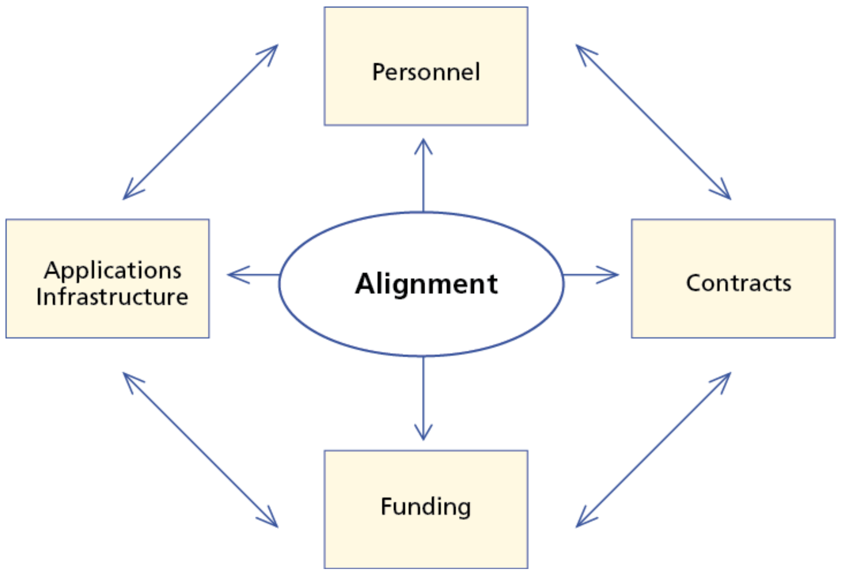


Figure 5-4. Social and Technical Alignment Pattern

5.4.2 Problem

Management Lags SE in a Complex Enterprise

There is little consensus on the best way to organize the elements of a program that will deliver a system and an enterprise capability. Conventional program management has experienced the same kind of failures as conventional systems engineering—being overwhelmed by myriad interactions with and constraints from the enterprise. How should a program be organized to fit into the architectural style of the enterprise?

5.4.3 Forces at Work

Technology's Structure and Style

Information technology has driven the architecture and elements of information enterprises. Technical complexity has been harnessed using layers, loose couplers, web services, infrastructure, and Internet and WWW standards. Successful systems engineering efforts have naturally been organized around these separate elements. However, the acquisition process is largely oriented to the development of a single, self-contained system, e.g., radar, an aircraft, or a command center. Historically, after a program is approved, funding flows down from a single source and a contract is awarded to produce the product. Lack of alignment between the way the program management is organized and the way the systems engineering must be done to realize enterprise success causes tension and confusion in the development process.

Table 5-4. Social and Technical Alignment Pattern

	Social and Technical Alignment
Summary	The alignment of people, processes, and technologies to match development and acquisition to the enterprise structure.
Context	• Sub-pattern of Harnessing Technical Complexity. • Managing a program that acquires a new system for use in a complex enterprise is itself a complex undertaking. • The people, processes, and technologies involved in delivering a new system have synergies with others across the enterprise.
Problem	• Conventional program management is also overwhelmed by complexity. • Little consensus on the best way to organize the elements of a program that will deliver a system and an enterprise capability. • Management science lags SE in dealing with complex enterprises.
Forces	• Technical complexity has been harnessed using layers, loose couplers, web services, infrastructure, and Internet and WWW standards. • Acquisition process is largely oriented to developing a single, self-contained system. • Multiple funding sources, contracts, and teams of people frequently work on the acquisition—and not all are under program office control. • Lack of alignment between how program management is organized and how successful systems engineering must be done causes tension and confusion in the development process.
Solution	• Organize program office, systems engineering, contracts, etc. to match the layered and segmented nature of the enterprise. • Separate personnel into separate teams to acquire applications, services, data stores, infrastructure, etc. • Align contracts to these separate activities.
Outcome or Resulting Context	• Management complexity was harnessed in the same style as technical complexity.

5.4.4 Solution

Align Management with Architecture

Successful program offices tended to organize their teams, contracts, and funding sources/cost centers to match the layered and segmented nature of the technical enterprise. They organized personnel into disjoint teams to separately acquire applications, services, infrastructure, data stores, and so on. They aligned contracts to these separate activities. They used the organization provided by the technology to also harness the complexity in the business processes.

5.4.5 Examples

A recurring theme in successful programs was separation of the development of applications from that of infrastructure. Taking the lead from an architecture that separated applications from infrastructure, successful programs tended to also separate the people and processes developing these separate parts. The enterprise architecture served as a template for this alignment. Typically, separate engineering teams were formed to deliver applications and infrastructure. They acted as product development units with full responsibility for cost, schedule, design, and marketing of their piece of the system within the context of the enterprise.

One program spent the bulk of its resources developing a robust infrastructure and then helped a variety of users to develop applications or modify existing ones to ride on that infrastructure. Another program used “time-to-market” as a driving requirement for its new applications and services. Whenever the current infrastructure slowed down delivery time, the infrastructure team was challenged to remove the impediment. As the development progressed, “time-to-market” decreased dramatically.

Frequently successful programs secured funding for different parts of the system from different sources, and for cross-enterprise capabilities from cross-enterprise funding sources. When one program found important cross-enterprise capabilities falling below the funding line of its immediate sponsor, it solicited funds from outside organizations interested in these capabilities. See Seek Secondary Sources for further information. Although the capability was out of scope for the program, it was a small investment with a big payoff. The program got credit for delivering a much needed enterprise capability, the enterprise got what it needed and program resources were not depleted. A win-win based on the alignment of the architecture and management process.

5.4.6 Outcome

Agile Management
Successful programs tended to harness the complexity within the development and acquisition process by aligning people and processes with the layered and loosely coupled technical architecture. Management complexity was harnessed in the same style as the technical complexity.

5.5 Plan to Re-plan

This pattern uses feedback to learn better courses of action and incentivize desired behaviors. It is illustrated in Figure 5-5, summarized in Table 5-5, and described in the remainder of the section.

5.5.1 Context of Plan to Re-plan Pattern

Complexity Requires Adaptation
This pattern is a sub-pattern of Harnessing Technical Complexity. A fundamental concept to the development of complex systems is that the system and the enterprise evolve and adapt in their environment. Adaptation requires feedback and learning.

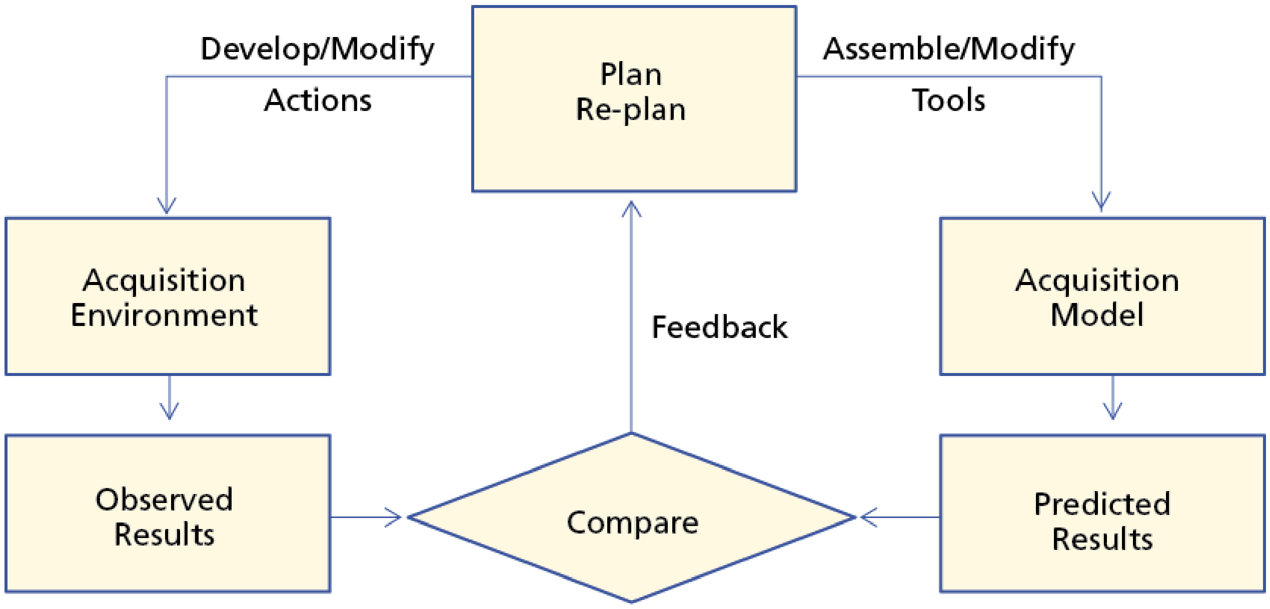


Figure 5-5. Plan to Re-plan Pattern

5.5.2 Problem

Experience in Conflict with Expectations
At any moment, the program office has a plan for the course of action that will develop a system to achieve certain goals. However, in a large and complex enterprise, actions frequently do not have the desired effects. Reality does not match expectations. Programs find themselves forced to constantly re-plan, potentially changing ends and means during program execution.

5.5.3 Forces at Work

Ambiguity and Change
The size and scope of the enterprise make it nearly impossible for the program office to know everything it needs to construct a lasting plan. The complexity of the situation may make accurate prediction essentially impossible, and therefore any actions taken may not give the predicted results. The enterprise environment may be different than assumed because as systems are added or new technologies are inserted, the changing nature of the enterprise makes both the parameters and the consequences of some actions unknowable. Because there is usually not a single enterprise user or monolithic goal, there is competition among stakeholders, maybe even hidden or conflicting stakeholder goals. The requirements may be ambiguous or changing. Funding may change. Like battle plans that seldom survive first engagements, program plans erode when confronting a complex enterprise.

5.5.4 Solution

Feedback and Incentives

What we observed in successful programs was a deliberate attempt to gather feedback, learn from it, and re-plan courses of action. The re-planning consisted of deliberations followed by decisions that began the next cycle of the Actions-Feedback-Plan process. The planning and re-planning effort resulted in actions that accomplished several things:

- Implemented choices of or changes to technology, architecture, design, and so on.
- Established or changed incentives to help shape the development environment.
- Improved the tools used in the feedback process.

Table 5-5. Plan to Re-plan Pattern

	Plan to Re-plan
Summary	Stimulate desired behavior through feedback and incentives, then learn from results what behavior is desired next.
Context	• Sub-pattern of Harnessing Technical Complexity. • Complex systems must evolve and adapt to their environment. • Adaptation requires feedback and learning.
Problem	• Experience (reality) is frequently in conflict with predictions and expectations. • Actions do not have desired effects. • Programs are forced to constantly re-plan.
Forces	• Accurate prediction is essentially impossible in a complex enterprise. • The enterprise environment may be different than what is assumed. • Enterprise changes if systems are added or technologies are inserted. • Requirements may be ambiguous or continually changing—no single enterprise user or monolithic goal. • Funding may change. • Stakeholders tend to act in their own self-interests.
Solution	• Make deliberate attempt to gather feedback, learn from it, and re-plan. • Use feedback to give program office situational awareness and to inform stakeholders how their actions are contributing or impeding the acquisition. • Use incentives to modify or adapt to stakeholder self-interests and drive behaviors. • Continually improve the tools used in the feedback process.
Outcome or Resulting Context	• Finding a path through the complex landscape. • Programs become dynamic learning, adaptive organizations. • Learn from successes and failures in time to successfully modify desired outcomes or courses of action.

When the results of executing the current plan did not match predictions or expectations, the program office took the opportunity to learn and correct its course of action. The current plan accounted for the development path that would deliver the desired capability. It relied on an evolving model of the entire acquisition environment: a picture of what the technology could achieve, of what the user requirements were, and of what the stakeholders would provide in terms of support for the chosen path. If a given technology failed to deliver an adequate prototype, it was changed. If the technology worked properly but user feedback was negative, the requirement was re-examined. If contractors, partners, or funding authorities were not meeting obligations, responsibilities were re-assigned or rules of engagement were changed—in particular, incentives were modified to induce multiple stakeholders to change behaviors. As the development proceeded, the learning process continued. They constantly tweaked their actions to find a successful path through the maze of complexity. Feedback was fundamental to developing an accurate picture of how things were going, and incentives were essential to driving the behavior of stakeholders. Re-planning was an integral part of the strategy.

Feedback and incentives are known to be effective strategies in complex problem solving. Companies like Wal-Mart have made their success in building feedback loops that allow them to stock what will sell best at any given time. Management guru Peter Senge has extolled the virtues of learning organizations.[22] Social scientist Dietrick Dorner showed that one of the traits of people who consistently did well in solving problems in complex situations was their habit of continually re-evaluating and re-planning their courses of action.[23] More than one industry has found that basing incentives on "realized sales" promotes manufacturing and sales divisions' working together to find the best combination of production runs and order mix—enterprise behavior.

5.5.5 Examples

Feedback Examples

For those programs where developers were collocated with users, the feedback was immediate and the learning was continuous. Successful prototypes were used operationally on a temporary basis while the design was fed back into the development of a more robust version. In other programs where the deployment was done through user operations centers with development personnel present, the feedback had a strong and immediate effect on the next development cycle. In one of the study cases, the program office also ran the operations facility, including the Help Desk. In that case, the feedback was contained within the same organization and the incentives to learn and improve were therefore built in. One of the programs used a combined government-contractor systems engineering team to decide on objectives and an integration contractor to keep a careful and public record of who was doing what, when. The learning loop was kept open to all.

Successful learning was useful during testing. An independent but well-informed testing organization was a key enabler of success. When the development and test communities banded together to view testing as "illuminating performance in critical situations" as opposed to simply "getting a passing grade," learning and performance was enhanced.

[22] P. Senge, *The Fifth Discipline: The Art & Practice of the Learning Organization*, Doubleday, 2006.
[23] D. Dorner, *The Logic of Failure: Recognizing And Avoiding Error in Complex Situations*, Cambridge, Mass: Perseus Books, 1997.

Influence Examples
Incentives – both positive and negative—were used to create and change the acquisition environment whenever feedback indicated it was necessary. In several of the programs for which 3–12 month task orders were written, the award fee was adjusted to specifically address the desired results at that particular point in the delivery. In multi-user environments, a special emphasis was placed on collaborative behavior. Some of the joint organizations had an ethic of leaving their "badges" at the door and dealing even-handedly with the solutions to problems. This was augmented with a "one-strike-and-you're-out" agreement to deal with any members showing strong bias toward their own organization's interests or proprietary solutions rather than the enterprise as a whole.

5.5.6 Outcome

Finding a Path Through the Complex Landscape
By making specific accommodation for gathering and processing feedback, program teams became learning, adaptive organizations, better matched to the dynamics of enterprise change. They learned from successes and failures in time to successfully modify desired outcomes and courses of action.

5.6 Technology Surfing

This pattern describes an ongoing process of identifying new and emerging technologies, experimenting with them and integrating what works into the evolving enterprise—"catch the next technology wave" rather than "create or wait for the big one." It is illustrated in Figure 5-6, summarized in Table 5-6, and described in the remainder of the section.

5.6.1 Context of Technology Surfing Pattern

Many Technology Choices
This pattern is a sub-pattern of Harnessing Technical Complexity. Myriad technology choices are available for the systems under development. Different programs make different choices, so the enterprise is never a monolith.

5.6.2 Problem

Exploiting the Right Technologies
Both the enterprise and the technologies are evolving. The SE effort must make predictions and decide which technologies would be a "best" fit to solve the problem at hand with the system under development and to endure in the enterprise.

5.6.3 Forces at Work

Change in Technology and Enterprise
Users want and expect delivered systems to take advantage of current technology. Predictions about which technologies will be effective at the end of the development cycle carry the risk of uncertainty. The longer the technology forecast horizon, the greater the development risk.

Choosing a current technology at the beginning of a program and sticking with it for the entire development time risks that the system will be obsolete when it is deployed. Choosing a future technology carries with it the risk of making one large "bet" that could be wrong, either because the technology doesn't mature or it doesn't gain industry support. Although a given technology may work well for the system at hand, it may not be fit for the enterprise as a whole (e.g., it may not be supportable across the enterprise). Products that are early to market may influence the choices that other enterprise stakeholders make.

5.6.4 Solution

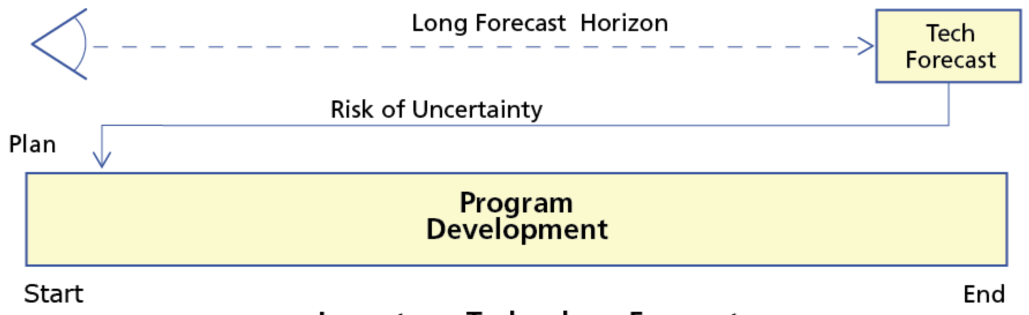
Evolutionary Strategy—Ride the Current Waves

Successful programs that required the latest technologies be implemented tended to use short evolutionary cycles to build and deploy their capabilities. The evolutionary cycle practice decreased the maximum risk exposure for any technology implementation in that cycle, and allowed re-planning for the next cycle. The practice enabled the technology implementation to be tested to see if it was not only fit for purpose in the system, but also fit for the enterprise. The short evolutionary cycles provided the opportunity to integrate unforeseen or previously immature technology jumps into the system development. Last, the short, frequent cycles created pressure on other enterprise stakeholders to choose technologies that would converge with theirs. This approach can be thought of as an extension of spiral development [24] used in large software projects—focused on technology choices and expanded to the enterprise. It is akin to idealized design as defined by Gharajedaghi,[25] which relies on three constraints: technological feasibility, operational viability, and learning and adaptation. By incrementally developing the systems components and having a continual technology forecasting cycle, the program office incurs less risk in each of these areas. First, the technology forecast is more accurate because its time horizon is closer in. Second, the technology is more likely to be operationally viable because the resources involved in implementing it are easier to estimate. Last, because the iterative cycles are much shorter than the full development time, there is greater opportunity for feedback and learning. (See the Plan to Re-Plan Pattern in Section 5.5.)

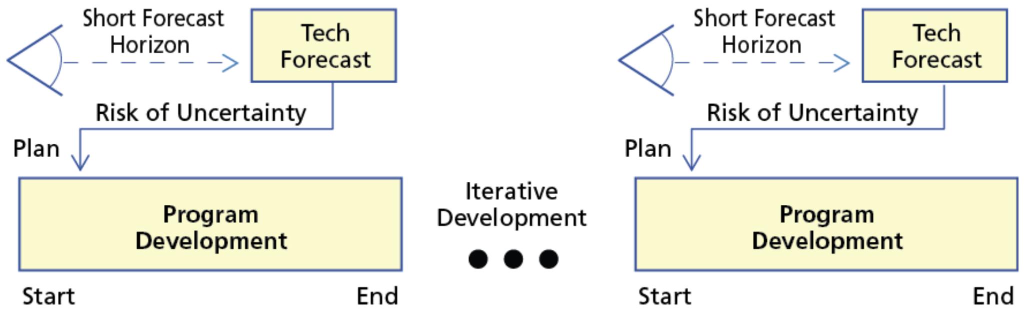
5.6.5 Examples

Successful programs that were required to deliver the latest technologies in their systems and to the enterprise tended to use evolutionary cycles. They geared not only their other development processes but also their technology forecasting activities to each cycle. In multiple cases, they wrote 3- to 12-month task orders for contractors and tagged award fees to the outcomes. They avoided the bias of proprietary solutions by having their own competent systems engineering staff act as honest brokers on the technology choices. In some cases, they augmented organic staff with representative from outside organizations, for example, users, industry experts, and other quasi-government organizations. In other cases, the developer and the user were collocated and the development cycles were weeks long instead of months. In all cases, a process was installed that gave rapid feedback to the developers on the efficacy of the particular technology choices.

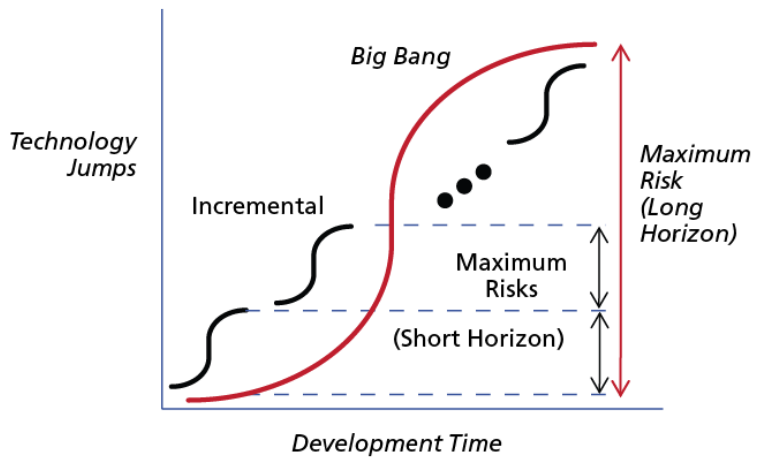
[24] Boehm B., A spiral model of software development and enhancement, *ACM SIGSOFT Software Engineering Notes*, *ACM*, 11(4):14–24, August 1986.
[25] J. Gharajedaghi, *Systems Thinking: Managing Chaos and Complexity*, Butterworth-Heinemann, 1999, pp. 129–130.



a. Long-term Technology Forecast



b. Short-term Technology Forecast (Surfing)



c. Comparison of Max Risks

Figure 5-6. Technology Surfing Pattern

Table 5-6. Technology Surfing Pattern

	Technology Surfing
Summary	An ongoing process of identifying new and emerging technologies, experimenting with them and integrating what works into the evolving enterprise—"catch the next technology wave" rather than "create or wait for the big one."
Context	• Sub-pattern of Harnessing Technical Complexity. • Many technology choices are available for the systems under development. • Different programs make different choices, so the enterprise is never a monolith.
Problem	• To make predictions and decide which technologies would be a "best" compromise for the system under development and the enterprise.
Forces	• Users desire and expect current technology. • Choosing a current technology for the entire development time risks that the system will be obsolete when it is deployed. • Choosing a future technology risks making one large "bet" that could be wrong. • Given technology may work well for the system at hand, but it may not be fit for the enterprise as a whole. • Early-to-market products influence other enterprise stakeholders.
Solution	• Short evolutionary development cycles: - Decrease maximum risk exposure for any technology in that cycle. - Afford the opportunity to integrate unforeseen or previously immature technology jumps into system development. - Test the enterprise to see if that technology is not only fit for purpose in the system, but also fit for the enterprise. - Pressures other enterprise stakeholders to choose technologies that would converge with theirs.
Outcome or Resulting Context	• Development team makes course corrections to stay reasonably current with technology and enterprise change throughout development cycle.

5.6.6 Outcome

Lower Risk Horizon
Using shorter development and implementation cycles mitigated the risk of choosing a technology implementation that would not be feasible or fit for the enterprise. Rapid feedback provided course corrections. The systems stayed reasonably current with technology and enterprise changes.

5.7 Architect.org

In this pattern, the government program office team assumes full responsibility for architecting and overseeing development of the system capability. It is illustrated in Figure 5-7, summarized in Table 5-7, and described in the remainder of the section.

5.7.1 Context of Architect.org Pattern

Strong Government Team

This pattern is a sub-pattern of Harnessing Technical Complexity. It applies primarily to high-risk programs for which the system under development is relatively unaffected by changes to other enterprise systems. It may be that the system operates in isolation or that it is loosely coupled to the rest of the enterprise (see Section 5.3). Either way, the system under development is not strongly affected by changes elsewhere in the enterprise.

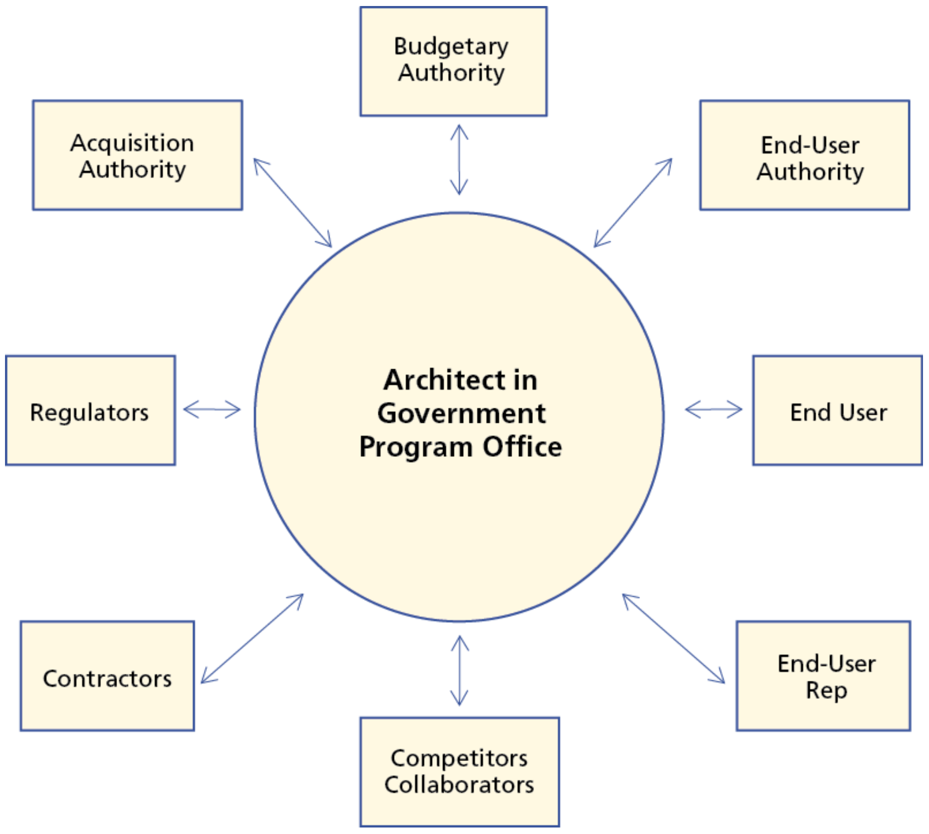


Figure 5-7. Architect.org Pattern

5.7.2 Problem

Costly Overruns

On high-risk programs, there is frequently a large gap between what the user expects or needs and what the contractor delivers. It is more the rule than the exception that closing this gap results in cost escalation and overruns.

5.7.3 Forces at Work

Risk, Knowledge, and Change

Uncertainty in the technology, requirements, and funding increases a program's risk. Contractors naturally bid the price of a proposal to cover the mitigation of these risks. Program risk shifts between the government and the contractor depending on which organization performs the systems engineering and assumes responsibility for what gets built and how. A government team with sufficient knowledge and expertise can save money by clarifying what the contractor is to do and what it should cost. When the government assumes more of the risk, it has to fund the implementation of any change, but it does not have to fund the contractor's contingency planning, risk mitigation, or false starts. Naturally, in a cost-plus-award-fee contract, the government funds the implementation of any change. Likewise, if the government team does not have sufficient knowledge and expertise, it will not effectively mitigate the risks. In this case, continually renegotiating the contract and funding engineering change orders can become expensive. In the worst case, the program will fail.

Table 5-7. Architect.org Pattern

	architect.org
Summary	Government program office team assumes full responsibility for architecting and overseeing development of the system capability.
Context	• Sub-pattern of Harnessing Technical Complexity. • High-risk program for which the system under development is relatively unaffected by changes to other enterprise systems. • System under development is isolated from or loosely coupled to wider enterprise.
Problem	• There is frequently a large gap between what user expects or needs and what contractor delivers. • Closing this gap often results in cost escalation and overruns.
Forces	• Uncertainty in technology and requirements increases program risk. • Contractors bid up price to cover risk mitigation. • When government assumes more risk, it has to fund the implementation of any change, but does not have to fund the contractor's contingency planning, risk mitigation or false starts. The government funds any changes in a cost-plus contract.
Solution	• Government puts together a highly skilled and centralized systems engineering and acquisition team. • They assume much of the risk away from contractors by taking responsibility for the system and enterprise architectures. • They act as single entry point (i.e., Architect.org) for all problems from start to finish—their knowledge of the technology and mission is extensive. • Government team directs pool of contractors to execute the development.
Outcome or Resulting Context	• Well-directed programs in the face of uncertainty and change.

5.7.4 Solution

"Building Trade" Model: SE as Architect, Contractors as Builders.
The relative independence of the system under development from the rest of the enterprise allows the government to put together a highly skilled and centralized systems engineering and acquisition team. They must be capable of acting as a single point of contact (i.e., Architect.org)

from start to finish—their knowledge of the technology and mission is extensive. Last, there is a pool of contractors able to execute the development.

This highly competent systems engineering and management team takes much of the risk away from the contractors by assuming responsibility for the system and enterprise architectures. If it is possible to define the system requirements such that they remain reasonably stable, the government team can specify what to build up front and use contractor resources and capacity to build it. When there is uncertainty and frequent change in the program, the government can assume a role similar to that of general contractor in the building trade, shortening the time horizon of the contractor's next action. Thus the direction of the development path can continually shift, even though each contractor action is well defined. The government takes a major role in integrating the elements of the system under development, as well as in integrating the system into the broader enterprise. For any problems that arise, "go to Architect.org."

The Architect.org pattern worked well when the system parameters and environment were known and relatively unchanging, as well as when they were not. The former case is close to the traditional systems engineering model, where the government systems engineering team writes the specification and monitors the contractor through all phases of development. The latter case is more agile and evolutionary. In those cases, the scope of the programs didn't change much, but the specific path to development changed frequently.

5.7.5 Examples

The government systems engineering teams of several successful programs had an excellent grasp of technology and operations in the mission domain. They were able to act quickly and cost effectively when a high-risk technology needed to be evaluated or a requirements change needed to be implemented. Clear direction flowed from the program office to the contractors. In one case, the government chief engineer served also as the Contracting Officer's Technical Representative (COTR) with responsibility for $11B of spending over three years. Because a skilled SE team produced cost estimates that were very good, negotiations with contractors went quickly and gave the government a fair price for the value delivered. The ripple effect of having good cost estimates and continually delivering on them gave the program credibility with the users and caused the funding authorities to keep the funding stable. That in turn benefited the contractors as well as the government. Uncertainty and change were not eliminated, but they were handled efficiently.

The importance of a sufficiently knowledgeable government SE and acquisition team was illustrated in another way. Several successful programs rose out of the ashes of failed attempts. A key ingredient to the restructuring of these programs was reconstituting the program office team with the skills necessary to assume the inherent program risks. Many of the success stories began with finding people with the right SE and acquisition skills within the government.

5.7.6 Outcome

Well-Directed Programs in the Face of Uncertainty and Change
Risk mitigation in the face of uncertainty and change required agility. The government was able to provide that agility, and the contractor provided the horsepower to actually build the system. Shifting program risk from the contractor to a highly skilled SE team within the program office resulted in both contractor and program success.

6 Summary

We observed the behaviors of successful systems engineers and managers in the government acquisition of IT-intensive systems—focusing on those cases in which their performance deviated from the norm in a positive way. We made the assumptions that some implicit knowledge that the positive deviants possessed drove their successful behaviors, and that if that knowledge were made explicit, others could benefit. We captured the knowledge and behaviors of the positive deviants explicitly in case notes and documented recurring themes as design patterns. Each pattern addresses a solution to a specific problem, and the full set of patterns forms a rough image of how to successfully systems engineer IT-intensive systems in the real world of government acquisition. By capturing successful acquisition and SE experiences in design patterns, we preserve them for application to future programs and provide value to government acquisition.

Appendix A Study Methodology

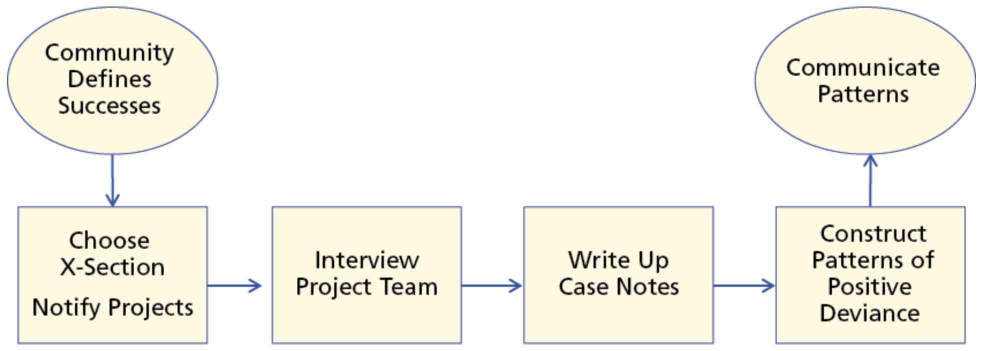
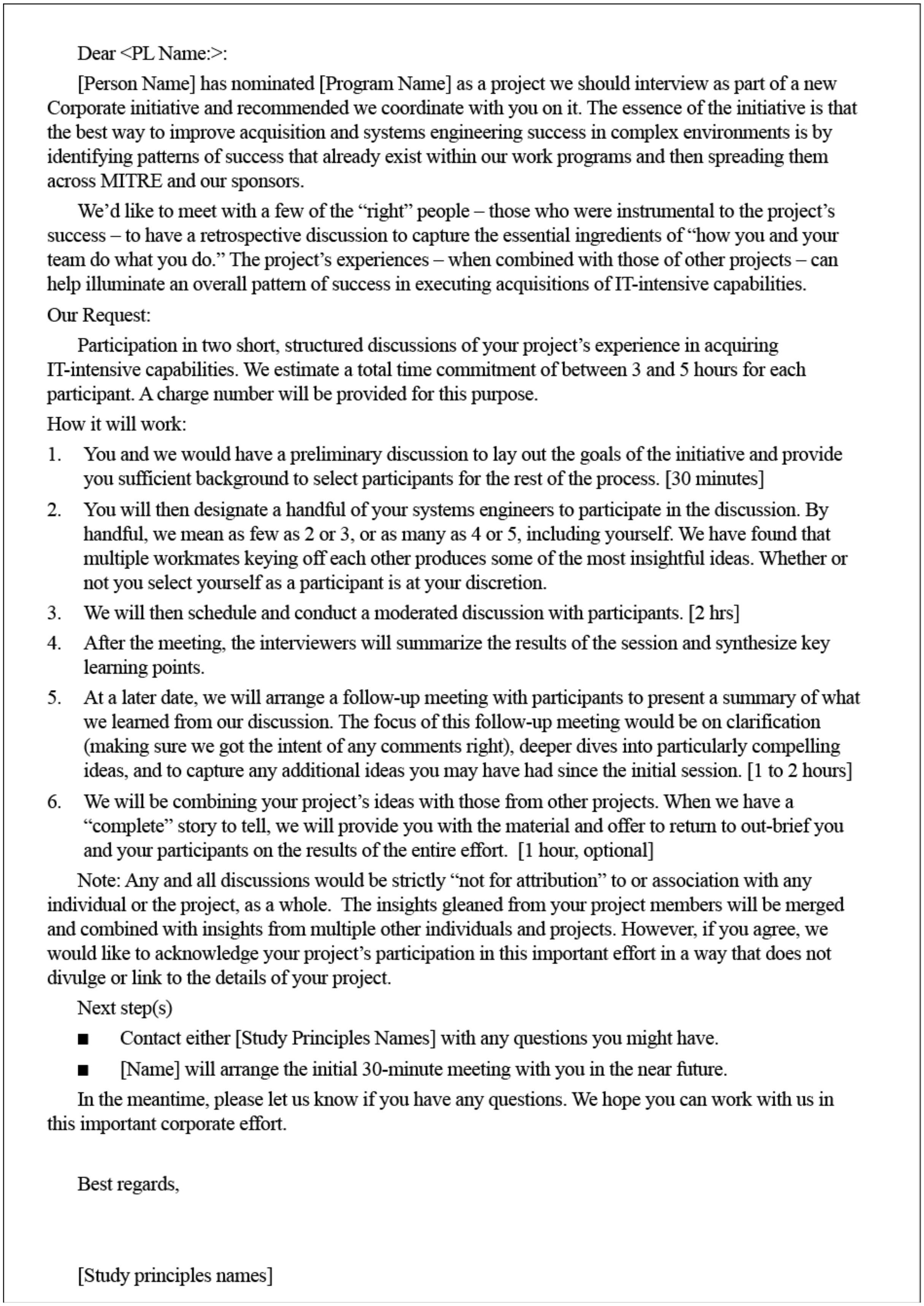


Figure A-1. Flowchart of Methodology

This is the correspondence initially sent to project leaders:



Dear <PL Name:>:

[Person Name] has nominated [Program Name] as a project we should interview as part of a new Corporate initiative and recommended we coordinate with you on it. The essence of the initiative is that the best way to improve acquisition and systems engineering success in complex environments is by identifying patterns of success that already exist within our work programs and then spreading them across MITRE and our sponsors.

We'd like to meet with a few of the "right" people – those who were instrumental to the project's success – to have a retrospective discussion to capture the essential ingredients of "how you and your team do what you do." The project's experiences – when combined with those of other projects – can help illuminate an overall pattern of success in executing acquisitions of IT-intensive capabilities.

Our Request:

Participation in two short, structured discussions of your project's experience in acquiring IT-intensive capabilities. We estimate a total time commitment of between 3 and 5 hours for each participant. A charge number will be provided for this purpose.

How it will work:

1. You and we would have a preliminary discussion to lay out the goals of the initiative and provide you sufficient background to select participants for the rest of the process. [30 minutes]
2. You will then designate a handful of your systems engineers to participate in the discussion. By handful, we mean as few as 2 or 3, or as many as 4 or 5, including yourself. We have found that multiple workmates keying off each other produces some of the most insightful ideas. Whether or not you select yourself as a participant is at your discretion.
3. We will then schedule and conduct a moderated discussion with participants. [2 hrs]
4. After the meeting, the interviewers will summarize the results of the session and synthesize key learning points.
5. At a later date, we will arrange a follow-up meeting with participants to present a summary of what we learned from our discussion. The focus of this follow-up meeting would be on clarification (making sure we got the intent of any comments right), deeper dives into particularly compelling ideas, and to capture any additional ideas you may have had since the initial session. [1 to 2 hours]
6. We will be combining your project's ideas with those from other projects. When we have a "complete" story to tell, we will provide you with the material and offer to return to out-brief you and your participants on the results of the entire effort. [1 hour, optional]

Note: Any and all discussions would be strictly "not for attribution" to or association with any individual or the project, as a whole. The insights gleaned from your project members will be merged and combined with insights from multiple other individuals and projects. However, if you agree, we would like to acknowledge your project's participation in this important effort in a way that does not divulge or link to the details of your project.

Next step(s)

- Contact either [Study Principles Names] with any questions you might have.
- [Name] will arrange the initial 30-minute meeting with you in the near future.

In the meantime, please let us know if you have any questions. We hope you can work with us in this important corporate effort.

Best regards,

[Study principles names]

Appendix B Case Overviews

The ground rules of the study included that the authors would not publish the names of the programs or the people interviewed. In this section, we have provided their salient features without revealing identities. We renamed the programs as follows:

Case 1: *High Priority and Expensive*

Case 2: *Centralized Like Never Before*

Case 3: *Large Program in Trouble*

Case 4: *Integrating Disparate Systems*

Case 5: *U.S. Development of International System*

Case 6: *Bogged-Down Stakeholders*

Case 7: *The System No One Was Using*

Case 8*: One Cutting-Edge Technology*

Case 9: A Product Line Tailored for Users

Case 10: *Sophisticated Worldwide Planning*

Case 11*: Worldwide Support Services*

Case 12*: Expedition into the Unknown*

A brief summary of each renamed program with a quote from one of its team members is given below.

Case 1: High Priority and Expensive

The Case 1 program was initiated to fill a high-priority government need to integrate the individual capabilities of several cross-agency systems into a globally synchronized capability. The integration effort cost a small percentage of the many-billion-dollar investment in the total capability. Because of its priority, Case 1 was given latitude to work outside the traditional acquisition processes and regulations, but it also inherited close political scrutiny and oversight.

> "Because many of the systems already existed, the approach ... became collaborative ... Here's what we'd like to do... Here's what we think we can deliver."

Case 2: Centralized Like Never Before

The Case 2 program was a multi-billion dollar procurement of information infrastructure components and services. The worldwide users had previously purchased their own systems and resisted the new centralized development and procurement strategy.

> "A lesson learned was that moving fast and loose works well up to a certain size and scope of problem."

Case 3: Large Program in Trouble

The Case 3 program had been stopped and was in danger of being canceled. It provided essential services to dozens of different classes of users, thousands of whom use the system concurrently, and it generated more than a billion dollars per year in commerce with supporting industries and government organizations. The restructured program was the subject of the case study.

> "The requirements were not well understood and exceeded the dollars available. We spent considerable time with stakeholders to prioritize requirements and come up with options that would fit the budget."

Case 4: Integrating Disparate Systems
The Case 4 program was an attempt to build a seamless network of cooperating users, linking their systems though a new service-oriented architecture. These systems were expensive, and the users were not accustomed to sharing information. The integration effort was orders of magnitude less costly than each of the disparate system programs. Thus the challenges were as much social as technical.

> "This has not been easy ... in changing mindsets, but several [customer] believers are paving the way for a mindset change."

Case 5: U.S. Development of International System
The Case 5 program was a U.S. development for an international organization. The modernization affected several IT-intensive centers with hundreds of operators and thousands of commercial and government clients. It had to deliver modernized capability within the context of current operations.

> "A bad reputation can follow a program long after it is no longer deserved."

Case 6: Bogged-Down Stakeholders
The Case 6 program was a multi-billion dollar development of a single system with multiple user communities and worldwide deployment. The program had a troubled development history with at least one major re-plan. It was bogged down in an inability to resolve valid technical issues among the stakeholders.

> "The technical communities debated solutions ... endlessly chasing their tail and never coming to a conclusion. The [government program director] said it was like watching paint dry."

Case 7: The System No One Was Using
The Case 7 program had spent over a half billion dollars in 10 years with few of the intended users actually adopting the system. It was slated to correct the situation with a budget of over $25M per year. It had three different government agencies in charge of requirements, development, and use. It was a large information system with worldwide deployment.

> "Top cover also helped keep the 6000 mile screwdriver at bay."

Case 8: One Cutting-Edge Technology
The Case 8 program provided a single function with high technology, expensive piece parts to a small community of users. The government's system engineering work force consisted of 150 individuals from several government and quasi-government organizations.

> "Any disciplined systems engineering process will work as long as it is provided with high-quality, unbiased information."

Case 9: A Product Line Tailored for Users
The Case 9 program was to build a family of products to serve multiple users performing a similar function in their own unique ways. It delivered an information infrastructure and a product line of plug-in modules.

> "Early failures pointed to a need to better track platform progress and maturity."

Case 10: Sophisticated Worldwide Planning
The Case 10 program delivered a collection of software components to perform sophisticated planning, execution, and assessment of operations. It operated with hundreds of users in about a dozen locations around the world.

> "The question ... How do you deal with frequent operational community turnover? The answer is credibility through continual presence of A-Team resources within the operational community."

Case 11: Worldwide Support Services
The Case 11 program delivered modernized enterprise information services to a large number of users involved in support operations. It was responsible for development as well operations.

> "[The program] was not defined by technology choices ... sharing agreements were critical. Being responsible for operations as well as development was also a key to our success."

Case 12: Expedition into the Unknown
The Case 12 program was to develop an advanced technology information system as the centerpiece of the future enterprise framework and act as an enabler for yet-to-be-defined operations.

> "We worked directly with the user every day, and when a successful prototype was developed, we continued to use it, but fed it back into the acquisition process to get the '-ilities' taken care of."