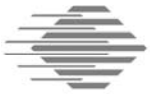
Fourth DoD Product Line Practice Workshop Report

John Bergey
Sholom Cohen
Matthew Fisher
Grady Campbell
Lawrence Jones
Robert Krut
Linda Northrop
William O'Brien
Dennis Smith
Albert Soule

October 2001

TECHNICAL REPORT
CMU/SEI-2001-TR-017
ESC-TR-2001-017



Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Fourth DoD Product Line Practice Workshop Report

CMU/SEI-2001-TR-017
ESC-TR-2001-017

John Bergey
Sholom Cohen
Matthew Fisher
Grady Campbell
Lawrence Jones
Robert Krut
Linda Northrop
William O'Brien
Dennis Smith
Albert Soule

October 2001

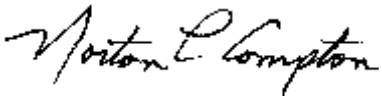
Product Line Systems Program

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Norton L. Compton, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

The Software Engineering Institute (SEI) held the Fourth Department of Defense (DoD) Software Product Line Practice Workshop in March 2001. The workshop was a hands-on meeting to identify industry-wide best practices in software product lines; to share DoD product line practices, experiences, and issues; to discuss ways in which the current gap between commercial best practice and DoD practice can be bridged; and to obtain feedback on the first draft of the DoD Acquisition Companion to the Framework for Software Product Line Practice written by the SEI. This report synthesizes the workshop presentations and discussions.

1 Introduction

1.1 Why Product Line Practice?

An increasing number of organizations are realizing that they can no longer afford to develop multiple software products one product at a time: they are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain a market presence, compensate for an inability to hire, leverage existing resources, and achieve mass customization. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. They are adopting a product line approach for their software systems.

A *software product line* is a set of software-intensive systems sharing a common, managed set of features that satisfy the needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

Product line practice involves strategic, large-grained reuse as a business enabler. Some organizations have already experienced considerable savings by using a product line approach for software system production. Other organizations are attracted to the idea but are in varying stages of integrating product line practices.

In January 1997, the Software Engineering Institute (SEI) launched the Product Line Practice Initiative to help facilitate and accelerate the transition to sound software engineering practices using a product line approach. The goal of this initiative is to provide organizations with an integrated business and technical approach to systematic reuse so that they can produce and maintain similar systems of predictable quality more efficiently and at a lower cost.

The SEI has also refined the workshop results through work with collaboration partners, participation in other workshops, and continued research. This transition strategy has been executed in part by a series of product line workshops organized by the SEI[1]. Five of these workshops (in December 1996, November 1997, December 1998, December 1999, and December 2000) brought together international groups of leading practitioners from industry to codify

[1] The SEI also organized the first international Software Product Line Conference (SPLC1) held in Denver, Colorado in August 2000 [Donohoe 00].

industry-wide best practices in product lines. The results of these workshops are documented in SEI reports [Bass 97, Bass 98b, Bass 99, Bass 00, Clements 01a]. These reports identify product line best practices, collectively refining and synthesizing some of the best ideas presented, and also identify issues that still require solution. In March 1998, the SEI hosted its first Department of Defense (DoD) Product Line Workshop, *Product Lines: Bridging the Gap—Commercial Success to DoD Practice.* Product line practices, DoD barriers and mitigation strategies, as well as similarities and differences between DoD product line practices and commercial product line practices were discussed and documented [Bergey 98]. A Second DoD Product Line Workshop was held in March 1999. This workshop marked a turning point from the SEI perspective in that the DoD participants talked about how they were implementing or going to implement product lines, as opposed to the familiar lament from past DoD forums that it would be impossible to implement product lines within the DoD [Bergey 99a]. A third workshop was held in March 2000 [Cohen 00]. At all three DoD workshops, the SEI was encouraged to continue to hold other DoD workshop events and to continue to bring best commercial practices to the DoD through these forums.

The SEI continues to refine the collective workshop results through work with collaboration partners, participation in other workshops, and continued research. In addition, the SEI is producing a conceptual framework for product line practice. The Framework for Software Product Line Practice written by the SEI (henceforth referred to as the framework) describes the foundational product line concepts and identifies the essential activities and practices that an organization must master before it can expect to field a product line of software or software-intensive systems successfully. The framework organizes product line practices into practice areas that are categorized according to software engineering, technical management, and organizational management. These categories represent disciplines rather than job titles. The framework is a living document that is evolving as experience with product line practice grows. Version 3 of the framework was made available on the Web in September 2000 [Clements 00].

1.2 About the Workshop

The SEI held the fourth in a series of two-day DoD Product Line Practice Workshops in March 2001 to achieve the following goals:

- Identify industry-wide best practices in software product lines.
- Share DoD product line practices, experiences, and issues.
- Discuss ways in which the current gap between commercial best practice and DoD practice can be bridged.
- Provide feedback to the SEI on the DoD Acquisition Companion to the Framework for Software Product Line Practice written by the SEI (henceforth referred to as the companion).

The workshop participants were referred to version 3 of the framework and a pre-release draft of the companion to provide a common focus to structure the workshop presentations and discussions. All participants in this workshop were from the DoD acquisition and contractor community. They were invited based upon our knowledge of their experience with and commitment to software product lines as either DoD system acquirers or DoD system contractors. Together we discussed the issues that form the backbone of this report.

The workshop participants included

- Dan Allred, The Aerospace Corporation
- John Bergey, Product Line Systems Program, SEI
- Loring Bernhardt, MITRE Corporation
- Grady Campbell, Product Line Systems Program, SEI
- Robi Chadbourne, TASC
- Col. Dave Chaffee, U.S. Air Force, Electronics System Center, Director for Combat Air Forces C2 System
- Mark Dehlin, West Virginia High Technology Consortium (WVHTC)
- Ed Dunn, NAVSEA (Naval Undersea)
- Dan Dvorak, Jet Propulsion Laboratory
- Jack Ferguson, Director, Software Intensive Systems, Office of the Under Secretary of Defense (S&T)
- Matthew Fisher, Product Line Systems Program, SEI
- Cheryl Gable, USAISEC, Fort Lee, VA
- John Goodenough, Chief Technical Officer, SEI
- Clifford Hollander, The Hollander Group, Inc.
- Lawrence Jones, Product Line Systems Program, SEI
- Debbie Kulish, U.S. Army TACOM-TARDEC
- Linda Northrop, Director, Product Line Systems Program, SEI
- John Ohlinger, National Reconnaissance Office (NRO)
- Lt. Col. Glenn Palmer, U.S. Air Force, Director of the Computer Resources Support Improvement Program
- John Park, SAIC
- Dr. Rami Razouk, The Aerospace Corporation, General Manager Computer Systems Division
- Mary Rich, The Aerospace Corporation
- Wendy Roll, The Boeing Company
- Dennis Smith, Product Line Systems Program, SEI

- Albert Soule, Product Line Systems Program, SEI
- Jack Van Kirk, U.S. Army Aviation Electronic Combat Project Managers Office
- Diana Webber, Scitor Corporation

1.3 About This Report

This document summarizes the presentations and discussions at the workshop. As such, the report is written primarily for those in the DoD who are already familiar with product line concepts, most especially those who are already working with or initiating software product line practices in their own organizations. Those who desire further background information are referred to the documents listed below. Acquisition managers and technical software managers should also benefit from this report.

- *Basic Concepts of Product Line Practice for the DoD* [Bergey 00b]
- *A Framework for Software Product Line Practice, Version 3.0* [Clements 00]
- *Software Product Line Acquisition - A Companion to a Framework for Software Product Line Practice, Version 1.0* [Bergey 01d]
- *Software Product Lines: Practices and Patterns* [Clements 01b]

The remainder of this report is organized into four main sections that parallel the workshop format:

- Section 2: A Digest of SEI Overview Presentations on the State of Software Product Line Practice, which summarizes the two SEI presentations that set the context for the workshop
- Section 3: A Digest of DoD Presentations on DoD Experiences with Software Product Lines, which summarizes four DoD presentations
- Section 4: Software Product Line Practices: Working-Group Reports, which details four working-group reports that are organized around the contents of the companion. Each report reflects the interests, experiences, and style of the individual group. The emphasis and completeness of the information varies by group and by practice area. The practice areas discussed are important in their very selection.
- Section 5: Summary, which recaps the major themes of this report and suggests future directions

This report also contains a glossary and an appendix that contains focused feedback on the first draft of the companion.

2 A Digest of SEI Overview Presentations on the State of Software Product Line Practice

2.1 Introduction

Two SEI technical leaders in the product line work gave presentations aimed at setting the context for the workshop. Linda Northrop, Director of the Product Line Systems Program, led the session with an overview talk that highlighted the primary themes for the workshop. She reviewed the definition of a software product line, the state of commercial product line practice, the relevance of product lines to the DoD, the SEI Product Line Practice Initiative, and the framework.

She then distilled the results of the Fourth Product Line Practice Workshop held by the SEI in December 1999. She uncovered the issues and solutions related to tool support for product lines shared by experts from seven commercial organizations with real-world experience in developing and fielding software product lines.

Larry Jones then presented the SEI work on creating a draft of the companion.

2.2 Overview: SEI Software Product Line Practice Framework—Linda M. Northrop, SEI

2.2.1 What Is a Product Line?

An increasing number of organizations are realizing that they can no longer afford to develop multiple software products one product at a time: they are pressured to introduce new products and add functionality to existing ones at a rapid pace. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain a market presence, compensate for an inability to hire, leverage existing resources, and achieve mass customization. Many organizations are finding that the practice of building sets of related systems together can yield remarkable quantitative improvements in productivity, time to market, product quality, and customer satisfaction. They are adopting a product line approach for their software systems.

A *software product line* is a set of software-intensive systems sharing a common, managed set of features that satisfy the needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

This definition is consistent with the definition traditionally given for any product line—a set of systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission. But it adds more; it puts constraints on the way the systems in a software product line are developed because substantial production economies can be achieved when the systems in a software product line are developed from a common set of assets in a prescribed way. The product line architecture and components are central to the set of core assets used to construct and evolve the products in the product line. This common, product line software architecture[2] capitalizes on commonalities in the implementation of the line of products and provides the structural robustness that makes the derivation of software products from software assets economically viable.

Each product in the product line is formed by taking applicable components from the base of common assets, tailoring them as necessary through pre-planned variation mechanisms such as parameterization or inheritance, adding any new components that may be necessary, and assembling the collection according to the rules of the product line architecture.

By product line practice, we mean the systematic use of software assets to assemble, instantiate, generate, or modify the multiple products that constitute a product line. Building a new product (system) becomes more a matter of assembly or generation than creation. For each software product line, there is a predefined guide or plan that specifies the exact product-building approach.

Product line practice involves strategic, large-grained reuse as a business enabler. The key concepts are

- **the use of a common asset base** (with the architecture being the pivotal asset)
- **in the production** (according to a predefined and documented production plan)
- **of a set of related products** (whose scope has been clearly defined and validated with a business case).

[2] A software architecture of a computing system is the structure or structures of the system that consist of software components, the externally visible properties of those components, and the relationships among them [Bass 98a].

2.2.2 The State of Product Line Practice

A number of organizations have achieved their product line goals. They have already gained order-of-magnitude improvements in efficiency, productivity, and quality through the strategic software reuse afforded by a product line approach. However, even more important than significant cost savings, product line practice enables an organization to get its products to the market or field at the right time. Time to market has emerged as a critical success factor in a number of highly competitive product lines, such as cellular phones, pagers, and printers. If a product reaches the marketplace several months after its competitor, it may have lost its window of opportunity and become a failure regardless of its features or cost.

The Swedish naval defense contractor, CelsiusTech, turned to a product line approach in the development of its onboard ship command and control systems in the mid 1980s [Brownsword 96]. Its efforts resulted in a product line called Ship System 2000 that now spans 12 classes of ships from surface vessels to submarines. CelsiusTech has fielded more than 50 ship systems from the same architecture and set of components. The benefits of using this product line have included a reversal in the hardware-to-software cost ratio from 35:65 to 60:20 that now favors the software.

A number of other companies have had similar success using a product line approach. Hewlett Packard, which like CelsiusTech has been using a product line approach for the past 10 years, has collected substantial metrics showing cycle-time improvements of two to seven times with product lines. On one project it shipped five times the number of products, that were four times as complex and had three times the number of features, and with four times the number of products shipped per person.

Motorola used a product line approach for FLEXworks, a family of one-way pagers. They have shown a cycle-time improvement of four times with 80% reuse. Cummins Inc. uses a product line approach for the engine control software for its diesel family and cites an order-of-magnitude decrease in build and integration time since going to a product line approach. Among other commercial domains that have shown equally dramatic results are air traffic control (Thompson, CSF, Raytheon), commercial bank systems (ALLTEL), telecommunication systems (Ericsson, Nokia, Lucent, AT&T), college registration systems (Buzzeo), and consumer electronics (Philips). These organizations have not moved to product lines to break into the market. They have needed product line practice not only to improve time to market, but also to continue their health in the market, to maintain a market presence, to sustain unprecedented growth (especially poignant given today's employment market), and to compensate for an inability to hire.

Many more organizations are now attracted to the concept of software product lines to address their needs for faster, better, and cheaper software production. Before moving to a product line approach for software, an organization should first identify its business goals and then determine if product line practice is a viable strategy to reach those goals. Software product line practice is not a panacea, but it has demonstrated significant advantages in many organizations that had a business case to support a product line practice.

2.2.3 The Relevance of Product Lines to the DoD

There is a growing recognition within the DoD that new acquisition approaches leveraging best commercial practices need to be implemented. At the top DoD policy levels, acquisition reform has focused on using these best practices to reduce cost, schedule, and technical risks, and to advance architecture-based approaches to reuse that support open systems, interoperability, and commercial off-the-shelf (COTS) products. Statements by present and former top-level DoD officials all express a need for the DoD to leverage the best commercial practices that have turned around American commercial industry over the last decade. It is important for the DoD to use innovative, commercially proven practices to reduce cycle time, improve quality, reduce cost, improve efficiency, and reduce technical risks. At an operational level, it is not exactly clear how this will happen. Support is needed to understand which commercially proven practices cut cycle time and cost while improving quality and efficiency; what the viable architecture-based approaches to reuse are; and how systematic software reuse is adopted in a DoD organization.

There have been some DoD product line success stories that show results comparable to the commercial successes cited above. However, there are many other people within the DoD who are attracted to product line concepts but don't know how to proceed. Though progress has been made, we are not at the point where product lines are a truly viable, repeatable practice within the DoD: there is a gap between best commercial practice and routine DoD practices. Part of this gap is related to the standard acquisition approach of acquiring a single stovepipe system at a time, and part is attributable to the fact that the commercially successful practices have remained proprietary. The workshop summarized in this report is one of the planned activities of the Product Line Practice Initiative of the SEI, which is attempting to bridge this gap.

2.2.4 The SEI Product Line Practice Initiative

In January 1997, the SEI launched the Product Line Practice Initiative to help facilitate and accelerate the transition to sound software engineering practices using a product line approach. The vision of the SEI Product Line Practice Initiative is that product line development will one day become a low-risk, high-return proposition and that techniques for finding and exploiting system commonalities and for controlling variability will be standard software engineering practice in the DoD, government, and industry. Our strategies to achieve these goals are to

- Identify and mature product line practices of demonstrated effectiveness.
- Integrate and codify a business and technical approach to product line practice, accommodating multiple entry points, system types, organizational contests, and domains.
- Provide materials for implementing product line practice.
- Build a community and an infrastructure to transition product line practice.

The technology maturation work has focused on architecture-based development, reengineering and asset-mining techniques for legacy systems, and business and acquisition guidelines for software product lines. To this extent the SEI has defined guidelines and techniques for product line analysis [Chastek 01], architecture definition, architecture evaluation (the Architecture Tradeoff Analysis Method[SM]) [Kazman 00], and component development, reengineering legacy systems [Bergey 01c], assets (Options Analysis for Reengineering) [Bergey 01b], business case development for product lines [Cohen 00], product line concepts of operations, and acquisition strategies for product lines [Bergey 01a].

One of the ways in which the SEI executes its strategies and pilots these techniques is to collaborate directly with organizations on product line efforts. The NRO, the Joint National Test Facility, the U.S. Army Special Operations Aviation Technical Application Program Office, the F-22 Pilot Training Program, the Robert Bosch Corporation, and Caterpillar are some of the organizations that the SEI is working with to achieve success with software product lines. These efforts are aimed at maturing product line practices and targeted transitions.

Widespread transition has been effected through a series of workshops and presentations, targeted at well-defined audiences. The SEI has hosted five workshops for commercial leaders [Bass 97, Bass 98b, Bass 99, Bass 00, Clements 01a], organized nine for researchers and technologists, and hosted four for the DoD community [Bergey 98, Bergey 99a, Cohen 00]. Presentations have been given at countless government, commercial, and technical forums.

[SM] The Architecture Tradeoff Analysis Method is a service mark of Carnegie Mellon University.

There are challenges, both technical and nontechnical, that still need to be addressed:

- the lack of widespread understanding in the DoD of software architecture and its connection to the business life cycle and to other architectures
- no codified architecture and product line migration strategies for the vast number of legacy systems
- the lack of funding models to support strategic reuse decisions
- the lack of adequate support for product line tools
- few examples of acquisition strategies that support systematic reuse though product lines
- the lack of repeatable, integrated technical and management product line practices

However, there are current trends that are highly supportive of software product lines—trends that reduce the risk of product lines and make a move to product lines more viable. These trends include

- the growing acceptance of the importance of software architecture
- the proliferation within major organizations of self-sustaining architecture centers
- the maturity of object technology
- the standardization of commercial middleware
- the growing popularity of the notion of "rapid development"
- the community's acceptance of well-defined processes for software development
- the growing acceptance in the software engineering community of the importance of product line practices and the rising recognition of the amazing cost savings that are possible

There is also considerable evidence of the growing maturity of product line practice. Universities have latched onto software product lines as an area of research. Software product line concepts are being taught in some universities. European intercompany collaborations have been established. Product line workshops continue to be organized. In August 2000, 185 technologists from across the globe attended the first product line conference [Donohoe 00].

On the government front, the Control Channel Toolkit (CCT) product line effort of the NRO was completed on schedule and within budget, with no outstanding risks or actions. The CCT produced the following reusable assets:

- generalized requirements
- domain specifications
- a technical reference architecture
- component implementations

- test procedures
- a development environment definition
- a reuse guide

The first reuser of the CCT assets is achieving tremendous benefits in terms of lower costs, defect rates, and the staff and time required. These benefits are comparable to those of successful commercial product lines. The CCT represents a real success story about product line use in the government.

Government product lines are challenging in that the government is inherently an acquisition organization. However there are multiple options that a DoD organization can choose from in pursuing a product line approach:

- Scope the product line and develop the architecture.
- Acquire a product line architecture.
- Acquire the core-asset base.
- Acquire a product built using product line technology.
- Acquire a product and some set of reusable assets.
- Acquire products built from a government-asset base.
- Acquire an entire product line.
- Acquire products built from a nongovernment-asset base.

Before a DoD organization selects one of these strategies, it should first be careful to understand the domain(s) involved in the product, to scope the product line properly, and to build a business case stating the goals to be achieved by the product line and the justification for the selected strategy. The DoD environment is currently much more positive about product line thinking, and several major DoD contractor organizations have begun product line initiatives.

The contexts for product lines vary widely in the nature of products, the nature of market or mission, the organizational structure and culture, the process maturity, the technical skill, and the existence of legacy artifacts. Nonetheless, the SEI has noted through direct customer collaboration, workshops, and focused research, that there are some universal activities and practices that are key to successful product lines.

2.2.5 Software Product Line Practice Framework

We are capturing those essential activities and practices in the framework, which is conceptual, available as a Web-based, evolving document, and targeted primarily at members of organizations who are in a position to make or influence decisions regarding the adoption of product line practices.[3]

As depicted in Figure 1, at its essence, fielding a software product line involves *core-asset development*, *product development* from the core assets, and *management* to staff, orchestrate, and coordinate the entire product line effort. The arrows signify the high degree of iteration involved and the fact that there is no prescribed order as to how these activities take place.

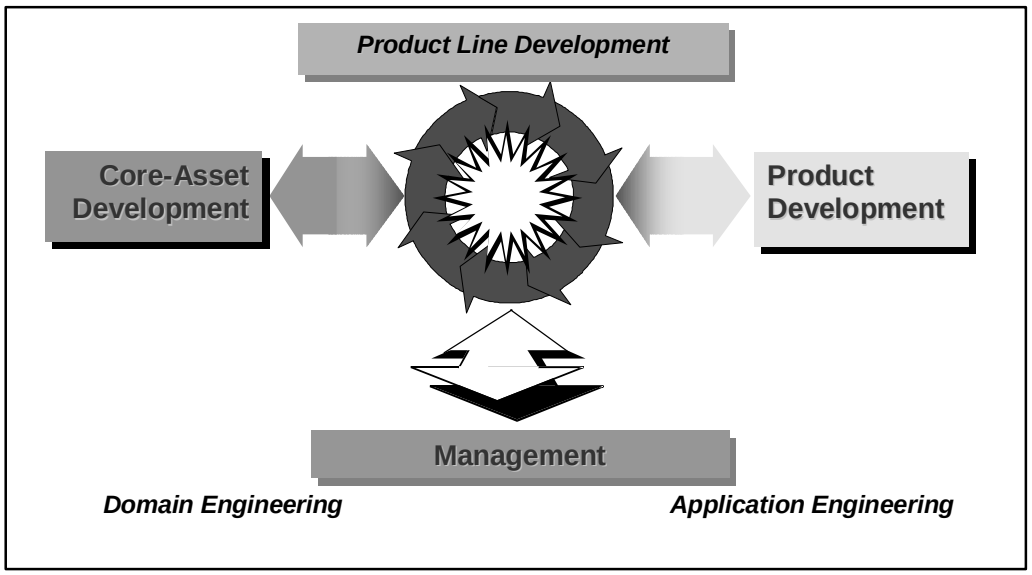


Figure 1: Essential Activities for Product Line Practice

On the left side of Figure 1, the critical core assets involved are the architecture and components. Inputs to the development of core assets are product constraints found by analyzing the similarities of and differences between current and projected products; the production constraints such as those found in a technical architecture; a production strategy for the assets; an inventory of pre-existing assets; and styles, patterns, and architectural frameworks. The outputs are the core assets, a preliminary list of the products they will support, and a production plan for how the core assets will be used in the development or acquisition of products.

[3] At the time of this workshop, version 3.0 of the framework was available at <http://www.sei.cmu.edu/plp/framework.html>. Workshop participants were asked to read the framework.

On the right side of Figure 1, individual products are developed from the core assets using the production plan that has been established. Product requirements are developed and refined with the existing core assets in mind, and products that systematically reuse the core assets are output.

There is a strong feedback loop between the core assets and products. Core assets are refreshed as new products are developed. In addition, the value of the core assets is realized through the products that are developed from them. As a result, core assets are made more generic by considering potential new products on the horizon. There is a constant need for strong and visionary management to invest the resources in the development of the core assets and to develop the cultural change needed to view new products through the filter of the core assets.

There are essential practices in a number of specific areas that are required to produce the core assets and products in a product line and to manage the process at multiple levels. The framework describes the essential practice areas for software engineering, technical management, and organizational management, where these categories represent disciplines rather than job titles. A *practice area* is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line. For individual practice areas, the framework provides

- an introductory description of the practice area
- aspects of this practice area that are peculiar to product lines
- how this practice area is applied to core-asset development
- how this practice area is applied to product development
- specific practices in this practice area
- risks in this practice area
- references

2.2.5.1 Software Engineering

The software engineering practice areas include those practices necessary to apply the appropriate technology to create and evolve both core assets and products as follows:

- Architecture Definition
- Architecture Evaluation
- Component Development
- COTS Utilization
- Mining Existing Assets
- Requirement Engineering

- Software System Integration
- Testing
- Understanding Relevant Domains

2.2.5.2 Technical Management

The technical management practice areas include those management practices necessary to engineer the development and evolution of the core assets and products as follows:

- Configuration Management
- Data Collection, Metrics, and Tracking
- Make/Buy/Mine/Commission Analysis
- Process Definition
- Product Line Scoping
- Technical Planning
- Technical Risk Management
- Tool Support

2.2.5.3 Organizational Management

Organizational management is the name we give to the management of the business issues that are visible at the enterprise level, as opposed to those at the project level. Enterprise management includes those practice areas necessary to position the enterprise to take fullest advantage of the product line capability. The organizational management practices include

- Achieving the Right Organizational Structure
- Building and Communicating a Business Case
- Customer Interface Management
- Developing and Implementing an Acquisition Strategy
- Funding
- Launching and Institutionalizing a Product Line
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Technology Forecasting
- Training

2.2.6 Future Direction of the Framework

The framework is intended to be a living document. We made a decision to make it available before all of the practice areas are complete. Version 1.0 was the first step in engaging the community to provide feedback on the framework's accuracy and usefulness. We incorporated community feedback and our growing experience base in version 2.0 and included a frequently asked questions section. Version 3.0 built upon that experience base. Version 4.0 will be published in *Software Product Lines: Practices and Patterns* [Clements 01b][4] and subsequently on the SEI Web site. We were encouraged to learn that more than 50 organizations reported to us their use of the framework in their software product line efforts. The community response has been most favorable.

2.3 The DoD Acquisition Companion to the Framework—Lawrence G. Jones, SEI

Originally, the framework was envisioned as meeting the needs of two broad communities interested in implementing a software product line approach: software development organizations and DoD acquisition organizations. However, these two communities differ in at least one obvious and significant way. While software development organizations will generally use acquisition in a supplemental role, DoD acquirers rely primarily (if not exclusively) on acquisition to obtain systems. Moreover, the DoD has some unique standards for software development (for example, the Joint Technical Architecture [JTA]) and often uses terminology that is not universally understood. The framework authors discovered that trying to address both communities in a single document was cumbersome, often resulting in awkward wording and compromising the needs of both groups.

Our solution to this dilemma was to create two separate documents. The framework would focus on the general needs of the software development community, and the companion would be written to address the needs of DoD acquirers. This presentation provides an overview of the companion.

We assume that the audience for the companion will consist of experienced, responsible acquirers. Thus, the companion will not be a primer on basic acquisition. Also it will not appeal to the acquirer who wishes merely to "outsource and forget." While we have targeted the DoD, our initial experience shows that the companion will also be relevant to other government organizations.

Our development strategy is similar to that used in writing the framework. We will develop the companion iteratively, releasing increasingly polished and complete versions over time. The first version, released in July 2001, contained roughly one-third of the envisioned content. Participants in this workshop received an initial draft version. It was our expectation that

[4] Version 4.0 of the framework and the book were not available at the time of this workshop.

the feedback received during this workshop would result in significant changes that would be reflected in version 1 of the companion.

The workshop version of the companion is structured into three sections. Section 1 provides a general introduction to the document. Section 2 describes software product line concepts, briefly discusses the general DoD acquisition context, and concludes with a broad overview of major concepts and issues in software product line acquisition. The heart of the document is Section 3, which is structured using the same 29 practice areas and three practice-area categories found in the framework.

A framework practice area describes a body of work or collection of activities that an organization must master to *develop* software product lines successfully. Similarly, a companion practice area describes a body of work or collection of activities that acquirers must master to *acquire* software product lines successfully. The practice areas in both documents are classified according to one of three categories: software engineering, technical management, and organizational management. The complete list of practice areas by category is shown in Table 1.

Table 1: Practice Areas

Software Engineering Practice Areas	**Technical Management Practice Areas**	**Organizational Management Practice Areas**
Architecture Definition*****	Configuration Management	Achieving the Right Organizational Structure*****
Architecture Evaluation*****	Data Collection, Metrics, and Tracking*****	Building and Communicating a Business Case*****
Component Development	Make/Buy/Mine/Commission Analysis	Customer Interface Management*****
COTS Utilization	Process Definition	Developing and Implementing an Acquisition Strategy*****
Mining Existing Assets	Product Line Scoping*****	Funding*****
Requirements Engineering*****	Technical Planning	Launching and Institutionalizing a Product Line
Software System Integration	Technical Risk Management*****	Market Analysis
Testing	Tool Support	Operations*****
Understanding Relevant Domains		Organizational Planning
		Organizational Risk Management
		Technology Forecasting
		Training
***** Indicates practice areas to be included in version 1.0 of the companion		

Software engineering practices are those practices necessary to apply the appropriate technology to create and evolve both core assets and products. In the typical case, the details of carrying out these practices are largely the concern of the developer or for the purposes of this discussion, the *supplier*. The acquirer's role in these practices is generally to

- Set the direction for the supplier and establish the acquisition team's role through the appropriate acquisition documents.
- Monitor and review the supplier's progress.
- Monitor the supplier's technical process and products.
- Participate (strongly) in the establishment and evolution of requirements.

Technical management practices are those practices that are necessary to engineer the creation and evolution of core assets and products. Again the details of carrying out these practices are typically the concern of the supplier. The acquirer's role in technical management practices is generally to

- Set the direction for the supplier and establish the acquisition team's role through the appropriate acquisition documents.
- Monitor and review the supplier's progress.
- Monitor the supplier's technical management activities.
- Participate (strongly) in product line scoping.

Organizational management practices are those practices necessary for the orchestration of the entire product line effort. Here we have a shift in relative emphasis from the supplier to the acquirer. Generally, the acquirer will have the major responsibilities in these practices. While the supplier may have parallel activities in some practice areas (e.g., "Organizational Risk Management" or "Organizational Planning"), in others the supplier may have little or no role (e.g., "Building and Communicating a Business Case" or "Developing and Implementing an Acquisition Strategy").

The bulk of Section 3 of the companion consists of detailed information about the practice areas. For the workshop draft version, each practice area was presented with the following information:

1. an introductory description of the practice area. This contains a summary of the practice area from a supplier's point of view (i.e., a summary of the practice area as described in the framework). The purpose is to provide the acquirer with an appreciation for the supplier's tasks.
2. aspects related to single-system acquisition. Many, if not most, of the practice areas are important for the successful acquisition of any software-intensive system, not just those which include software product lines. We include this information to provide a basis for understanding what makes product line acquisition unique.

3. aspects related to product line acquisition. This builds on the single-system acquisition information to show the special concerns of product line acquisition.
4. references. This will be a list of source documents referenced in the practice area.

In keeping with our philosophy of building a living document in an iterative fashion, version 1 and subsequent versions may depart from this structure. As mentioned, we intend to incorporate feedback from this workshop[5] and other sources prior to publication. Once the companion is published on the SEI Web site (<http://www.sei.cmu.edu>), we will invite readers to provide feedback and suggestions for improving the document.

[5] Indeed based on feedback from this workshop, the author team subsequently decided to eliminate the section on "Aspects Related to Single-System Acquisition" and add a section on "Frequently Asked Questions" for each practice area.

3 A Digest of DoD Presentations on DoD Experiences with Software Product Lines

3.1 Introduction

The following four presentations related to DoD software product lines provided the DoD context that was helpful in framing subsequent discussions:

1. A Strategy for Common Battle-Management Software, by Loring Bernhardt of MITRE Corporation
2. Avionics Software Product Line Development, by Wendy Roll of Boeing
3. The Variation Point Model: A Graphical Representation of Variation Points for Use on a Family of Systems, by Diana Webber of Scitor Corporation
4. A Business Case for Strategic Reuse in the NRO, by Sholom Cohen (of the SEI) and John Ohlinger (of the NRO)

3.2 A Strategy for Common Battle-Management Software—Loring Bernhardt, MITRE Corporation

The Common Battle-Management Software (CBMS) program is motivated by the need to modernize several Air Force battle-management systems that currently use outdated, "ruggedized" computers. These systems/environments include the Region Air Operations Center/Air Defense Sector (RAOC/ADS), Ground Theatre Air Control System (GTACS) Control and Reporting Center (CRC), and, maybe in the future, the Airborne Warning and Control System (AWACS). As part of this modernization effort, the Air Force wants to exploit the commonality across these systems by combining common battle-management functions into a single operations capability. This would include using a common architecture to lay the groundwork for adopting a software product line approach for CBMS. CBMS is the union of the software functionality for all three of these environments. The benefits of such an approach would be reduced cost, enhanced combat effectiveness, increased interoperability, and better utilization of personnel.

As part of this effort, the CBMS team will also be looking at increasing combat effectiveness by accommodating new missions and improving timeliness—improvements that currently aren't feasible due to the degree of difficulty of changing the legacy-system software. Another goal related to moving towards a common software baseline is to improve operations

by providing a common user interface which, in turn, will improve interoperability and reduce training costs. In addition to development cost savings, there are potentially large long-term savings in operations and maintenance (O&M) costs since the system will replace several legacy systems and be easier to maintain.

Currently the CBMS team is developing a competitive acquisition strategy to encourage industry competition. The acquisition strategy includes the architecture, program requirements, and provisions to ensure that any proposed solution will provide users with the desired functionality and system-quality attributes, and meet the schedule needs of the various commands participating in the acquisition. The goal is to obtain the "best value" solution that industry can offer and that is consistent with CBMS goals and within an acceptable level of technical and programmatic risk.

There is a CBMS Architecture Team that is working to involve key CBMS stakeholders in developing a comprehensive architecture specification. These stakeholders include many commands throughout the Air Force as well as representatives from Canada. The team is using the C4ISR framework[6] as its initial starting point for developing the "to be" architecture for CBMS—a common architecture for two existing battle-management operations/systems (i.e., RAOC/ADS and GTACS-CRC). As part of developing the architecture specification, the team is working with the various stakeholders and assisting them in their modernization efforts by

- characterizing the common operational behavior requirements for CBMS for the 2008-2010 timeframe and increasing their understanding of the "big picture" context
- describing architecture differences among the existing legacy systems and identifying unique application-software needs
- identifying the support required for common air-defense analysis and characterizing shared CBMS components
- analyzing and characterizing the information-exchange requirements with external systems and increasing understanding of the interoperability and integration requirements
- identifying the architecture constraints, providing insight into tradeoffs, exposing requirements issues, and supporting their timely resolution
- establishing common terminology and specifying architectural concepts and requirements for the CBMS acquisition

[6] The C4ISR is the Department of Defense's *Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance* architecture framework that is described in terms of three views: technical, operational, and systems. That framework should not, however, be mistaken for a complete specification of a software architecture. According to Bass and associates, a software architecture includes the structures of a system (which comprise software components), the externally visible properties of those components, and the relationships among them [Bass 98a].

As part of the acquisition strategy, each offeror will be required to propose a "point-of-departure" system based on existing air-defense software that can be extended to meet the totality of CRC and RAOC/ADS requirements. As part of their technical presentation, all offerors will be required to demonstrate both their proposed approach (i.e., point-of-departure system) on an existing platform and the stability of their software products. In addition, all offerors will have to provide a "capabilities road map" that describes how they can extend their point-of-departure system to meet all the CBMS operational requirements. They will also have to provide a system's evolution road map that describes how they will actually evolve the legacy systems to meet the system CBMS requirements and software architecture requirements. This acquisition-strategy approach leverages past lessons learned about how a point-of-departure system affects architecture and the ability to integrate new and legacy software products such as COTS applications.

A government evaluation team will evaluate each offeror's proposed approach to determine if it complies with the CBMS requirements specification and to what degree it represents the "best value" to the government. In support of this role, the CBMS architecture team is developing the technical evaluation criteria to uniformly evaluate each offeror's proposed point-of-departure software architecture and associated products. The architecture-evaluation criteria will include such factors as

- how extendable the point-of-departure architecture is
- how the point-of-departure architecture will accommodate legacy-system interfaces
- how the point-of-departure architecture will accommodate both strategic and tactical missions

The architecture and software components that the winning contractor develops and delivers will be candidate core assets for a battle-management software product line.

One requirement of the CBMS software architecture is that it must be compatible with three different C4ISR system architectures that reflect the scope of the acquisition and the challenge entrusted to the architecture team. These three different system architectures correspond to different sensor, track, and communication system requirements; can operate in an unclassified, classified, or mixed mode; and have different physical configurations for system deployment.

Risks include resolving tradeoffs such as selecting a single contractor to be responsible for all system deliveries versus having multiple contractors involved, coordinating and integrating the CBMS acquisition effort with ongoing modernization efforts, and trying to satisfy schedule commitments for several systems with a single software delivery. Other challenges include negotiating agreements with different commands that will satisfy diverse stakeholder needs and coordinating and resolving issues that can potentially affect all users of the soft-

ware products. While these risks and challenges are significant, they are seen as being far from insurmountable.

The CBMS team believes that there can be significant benefits from having software architecture as a major acquisition driver. Specifying architecture and architecture evaluations as part of the acquisition can provide early visibility into critical design decisions. These design decisions drive the entire system and software development effort, are the hardest to get right, and have the most profound and lasting effect on the system. Software architecture evaluations can mitigate many of the technical risks associated with an acquisition and improve the likelihood that a system will satisfy its desired system-quality attributes. The potential benefits of adopting a common architecture for battle-management software are improved warfighting capabilities, better utilization of crews, reduction in operational costs, and significant acquisition-cost avoidance.

3.3 Avionics Software Product Line Development—Wendy Roll, The Boeing Company

In 1996, Boeing launched an initiative to enhance the reusability of mission-computing software across several of its platforms. This initiative targeted a product line approach to developing avionics and cockpit display software for several of the military fighter aircraft Boeing was producing at the time. The products of this approach would be installed as part of the standard maintenance and upgrade cycles for the platforms.

The project goals were to develop reusable application components and a reusable architecture framework. The architectural goals were to contain change in the logical and physical architectures by isolating avionics system specifics, to support both single- and multi-CPU systems, and to maximize reuse by maximizing the "pluggability" of the software components. The software bottom-line goal was real-time performance.

The resulting architecture was based on two architectural patterns: Layers and Model/View/Controller [Buschmann 96]. The layered architectural pattern limited system dependencies by partitioning the architecture into layers of components. For example, one layer includes components that provide access to hardware while another layer includes components that perform more abstract system functions such as radar processing or weapons control. The Model/View/Controller pattern isolated the core-domain representation from the user interface. The model contains the core functionality and data. The view and controller handle the user interface. The Model/View/Controller pattern enabled changes to the user interface without modifying the core functionality. This was vital since the platforms in the product line all had different display requirements.

The components are medium grained (approximately 1000 source lines of code [SLOC]) with hundreds of component types per platform. The components integrate multiple objects behind

a façade to provide unified coherent services to other parts of the software. For flexibility, components were designed as instances of patterns that make them configurable, distributable (across 4-10 processors), or capable of real-time levels of service. The components were developed by a common components group and configured to meet its needs by project-specific groups.

These patterns are combined to create a top-level architecture consisting of four layers: infrastructure services, model, operator, and configurator. The infrastructure services layer abstracts and encapsulates an execution environment that is based on the common object request broker architecture (CORBA) and consists of hardware, runtime, and operating-system mechanisms. The model layer implements the information model and core functionality of the application domain. The operator layer implements the user interface. The configurator layer implements the capability to configure and integrate the components for a complete system. Each of these layers may be further decomposed into additional layers. To achieve performance requirements, layering is non-strict, permitting the software in higher layers to bypass intermediate layers if needed to directly access lower-layer components.

The product line architecture was developed by the Core Architecture Team. This team was composed of representatives from the common software group (who manages common components), a representative (the lead architect) from each of the product groups, and a product line lead architect. This group worked together through an iterative process to determine architectural goals, components, and architectural issues. The forces that drove the architecture included legacy requirements, new technologies, and constraints—especially affordability.

The product line was contractor driven rather than customer driven (i.e., Boeing rather than its customers drove the product line effort). The effort was internally managed and funded. Advantages of the contractor-driven approach included the iterative development of the architecture and systems, and of the controls on the scope and architecture. Challenges included the lack of shared requirements among the programs, conflicts in production plan schedules, and the lack of incentives to customer program managers.

Although the product line effort was considered a success, there were a number of challenges and lessons learned. Up-front costs, changes in technology, and the different groups involved made it difficult to measure the benefits of this approach. The amount of reuse for each platform was directly related to the amount of shared requirements. Since the product line was contractor driven, there was limited control over those requirements.

Scheduling the stovepipe efforts (i.e., different programs with different schedules and priorities) also impacted the product line effort. Affordability was the key to this effort, but the promises of lower cost may not be enough to convince people/programs to work together and coordinate efforts.

The future requires commitment to the product line at every level. Both customers and contractors must buy into the approach. Development and upgrade efforts need to be coordinated and some sort of cost-sharing agreement needs to be in place. Current contracts are already in place with no previsions for cost sharing. It is difficult to introduce the research-and-development-funded product line approach into a program already under contract.

3.4 The Variation Point Model: A Graphical Representation of Variation Points for Use on a Family of Systems—Diana Webber, Scitor Corporation

This Variation Point Model (VPM) is based on work with the CCT[7] project of the NRO and was later refined as part of a PhD thesis[8] at George Mason University. The model addresses the problem of how to create a common core or common components from a legacy system.

In a sample problem, a set of common core components is identified from a legacy reference architecture for reuse in a different target architecture. Both the common core component and the target-system component are large-grained deployable units conforming to a set of defined interfaces.

While the overall problem includes breaking the legacy software into useful chunks for reuse in the target system, this talk focuses on how to gain leverage from the reusable chunks through variation points. A *variation point* identifies one or more locations where a variation will occur and identifies the mechanisms to use the variation point. The variation points provide a mechanism to enable multiple products in a product line to be tailored for uniqueness.

Figure 2 shows an intuitive illustration of how variation points are applied to common core components.

[7] For background information on the CCT, see Section 3.5 on page 27.

[8] While the original use of variation points was derived from project experiences, this presentation and the dissertation both use examples from a sample Automated Teller Machine (ATM) application. Examples presented in this summary represent that application.

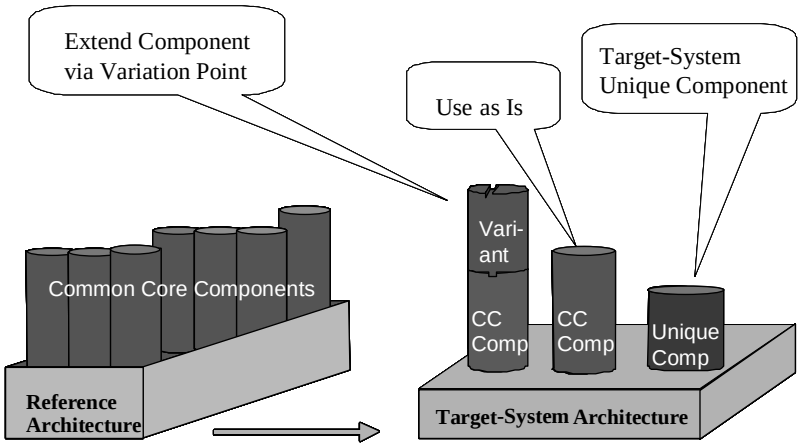


Figure 2: The Common Core Concept

The common core components from the reference architecture on the left are needed for reuse in the target-system architecture on the right. The notional target-system architecture illustrates three cases: use of a common core component as is, a unique target-system component, and a common core component that is extended with variation points. The capability of systematically defining and managing variation points is a significant requirement for product line success, especially for product lines with large numbers of products.

The VPM defines a set of detailed steps for defining variation points from four different views:

1. requirement-variation point
2. component-variation point
3. static-variation point
4. dynamic-variation point

For the requirement-variation-point view, the requirements are structured to specify variations. The component-variation-point view allocates the variation points to components. For the static and dynamic views, there are three types of variation points:

1. inheritance, which allows a reuser to overwrite or add parameters and methods
2. callback, which allows the reuser to extend the functionality of a component by registering a callback
3. parameterization, which allows a reuser to insert values for parameters in a component

Each of these views is discussed further below. For purposes of this discussion, the process of creating them involves three phases: the analysis phase, the architecture phase, and the design phase.

3.4.1 Requirement-Variation-Point View

The requirement-variation-point view is created during the analysis phase. This enables variation points to be specified in the requirements, either through enumerated “shall” statements or through use cases. For instance in the ATM example, requirement-variation points can specify the need for a choice among multiple languages (e.g., English or French) to guide the user.

3.4.2 Component-Variation-Point View

The component-variation-point view is created during the architecture phase and is focused at the component level. In this view the variation point is identified along with the components or classes that the variation point impacts. The component-variation point tells the user where the variation points are, whether they are mandatory, and whether the variation point impacts multiple classes.

3.4.3 Static-Variation-Point View

The static-variation-point view is created during the design phase and is focused at the class level using static analysis techniques. The VPM has a set of defined steps to enable variation points from the static-point view using inheritance, callback, and parameterization variation points. For example there are nine steps to building a static-variation-point view using inheritance:

1. Construct the class being specialized.
2. Show which package the class belongs to, to help the reuser find the class.
3. Determine which attributes and methods the subclass will overwrite.
4. Tag the parts that are to be overwritten.
5. Put the variation-point tag at the top of the list of attributes or methods if an attribute or method is to be added.
6. Use the Universal Modeling Language (UML) inheritance symbol to show inheritance to the subclass.
7. Use the constraint on the inheritance relationship to show that this inheritance is for reuse only.
8. Use the word *variant* in the subclass stereotype so that it can be easily detected as a variant by the reuser.
9. Once the static-variation-point view is complete, update the component-variation-point view if necessary.

The other types of variation points are defined in similar detail.

3.4.4 Dynamic-Variation-Point View

The dynamic-variation-point view is created during the design phase and focused at the class level during the code execution. After building the static-variation-point view, there should be one dynamic-variation-point view for every static-variation-point view. The types of variation points (inheritance, callback, and parameterization) for the dynamic-variation-point view will correspond to those of the static view.

3.4.5 Conclusion

The VPM allows a developer to communicate variation points with other developers. It also enables reusers to see where the variation points are located, to locate any mandatory variation points, and to build a variant from a variation point.

A survey is currently underway to evaluate the effectiveness of the variation point model with the CCT. Preliminary results suggest that the VPM saves 10% of the reuser's effort.

3.5 A Business Case for Strategic Reuse in the NRO—Sholom Cohen, SEI, John Ohlinger, NRO

The NRO is charged with overhead reconnaissance to support U.S. intelligence and warfighting needs. The NRO invests heavily in the acquisition and operation of space-based systems to meet this mission. In April 2000, the Acquisition Steering Group of the NRO chartered a team to examine software reuse as a money-saving strategy. The team, consisting of members from the NRO, the SEI, and the Aerospace Corporation, investigated the NRO situation and created a business case in support of strategic software reuse. This presentation provides some background on the NRO context and an overview of the NRO business case.

The NRO was motivated to investigate strategic software reuse in part due to the recently successful CCT program. This program, initiated in 1997 and completed in February 2000, created a common software architecture and set of reusable software components to support satellite ground command and control systems. A domain analysis of three products, the Distributed Command and Control System (DCCS), Standard Satellite Control Segment (SSCS), and MALTA, revealed that commonality in the command and control infrastructure components ranged from 49% to 89% among the three systems. The CCT assets were developed and then used to develop an operational system. In the portions of the development where CCT reuse was applied, benefits included

- Discrepancy reports were reduced by 90%.
- Productivity was improved by six times.
- Software builds were completed in weeks (instead of months).

- Integration time was reduced.
- Performance requirements were met.

The CCT experience demonstrated that strategic reuse could work in the NRO context. A business case was necessary to help determine if a broader program of strategic software reuse would be worthwhile.

The study team crafted a business case structured in five sections:

1. Current State
2. Preferred State: Strategic Reuse Through Software Product Lines
3. Business Justification for a Software Product Line Approach at the NRO
4. Strategies to Get the NRO to the Preferred State
5. Conclusions

3.5.1 Current State

This section contains a description of the current situation relative to NRO software systems—what is referred to as the *current state*. After a description of the "facts of life" (realities of the situation), the characteristics of the current state are classified as strengths or weaknesses. Examples of strengths include

- The NRO has extensive experience with space-based systems.
- The NRO has proven processes for fielding operational systems.
- Program managers have full authority/responsibility.

Examples of weaknesses include

- Acquisitions are typically independent (stovepiped).
- The acquisition process is not well suited for systematic reuse.
- There is little contractor incentive beyond opportunistic reuse.
- Contractors have a high degree of autonomy.

3.5.2 Preferred State: Strategic Reuse Through Software Product Lines

This section of the business case describes the preferred state—a situation that would admit the facts of life, build on strengths, and address weaknesses. The general characteristics of the preferred state include

- lower development costs
- lower costs for transition/readiness
- lower maintenance costs
- lower operations costs
- improved quality
- the ability to maintain the current level of performance and reliable operations

The business case contends that strategic reuse through *software product lines* will effect a situation that delivers exactly these characteristics. This section describes what is necessary to achieve success with software product lines and also includes the benefits and risks associated with the approach.

3.5.3 Business Justification for a Software Product Line Approach at the NRO

This section is the heart of the business case. It includes the business justification for using a product line approach for NRO software development. It compares the current, non-product line approach of the NRO (one that uses historical cost data) to a product line approach (one that uses CCT cost data).

Key assumptions for the basic cost calculations include[9]

- There will be an average of two new systems per year over five years in the product line.
- The degree of reuse (DOR) was assumed to be 25% (i.e., 25% of a new system may be derived from existing assets).
- The cost of reuse (COR) was assumed to be 25% (i.e., the cost of reusing an asset is 25% of the cost of creating that asset from scratch).
- The costs of reuse include those of
  - asset development
  - asset sustainment
  - product development and sustainment using the assets

[9] These are conservative assumptions. A sensitivity analysis is possible by adjusting the parameters associated with them.

Using the historical data and applying the assumptions, traditional single-system development was compared to three reuse scenarios:

- Scenario 1: Build assets as the basis for the future development of products. Restrict the sustainment of the assets to routine maintenance and responses to discrepancy reports.
- Scenario 2: Build assets as in Scenario 1, but invest in further refinements to the assets. Maintenance activities go beyond the bare minimums of Scenario 1. (This scenario has the effect of decreasing the COR from Scenario 1.)
- Scenario 3: Expand the features covered by the assets into new domains beyond those of Scenarios 1 and 2. (This scenario has the effect of increasing the DOR over Scenarios 1 and 2.)

Cost comparisons in millions of dollars are summarized in Table 2. The projected cost savings range from $101M to $220M.

Table 2: Costs and Savings Projections Over Five Years

Scenario	**Initial Asset-Development Cost**	**Annual Maintenance**	**Total Asset-Cost Projections (over five years)**	**Project-Cost Avoidance**
1	$16M	$1.6M	$23.2M	$101M
2	$16M	$3.2M	$30.4M	$127M
3	$32M	$3.2M (years 1-3) $6.4M (years 4 & 5)	$54.4M	$220M

A trend chart such as Figure 3 illustrates even more dramatic results from Scenario 1. The numbers along the vertical axis represent millions of dollars, and the numbers along the horizontal axis represent years. With the conservative assumptions of a DOR of 25% and a COR of 25%, the gap between product line costs and stovepipe costs grows dramatically. Furthermore, these savings will come sooner if more than two systems per year are produced.

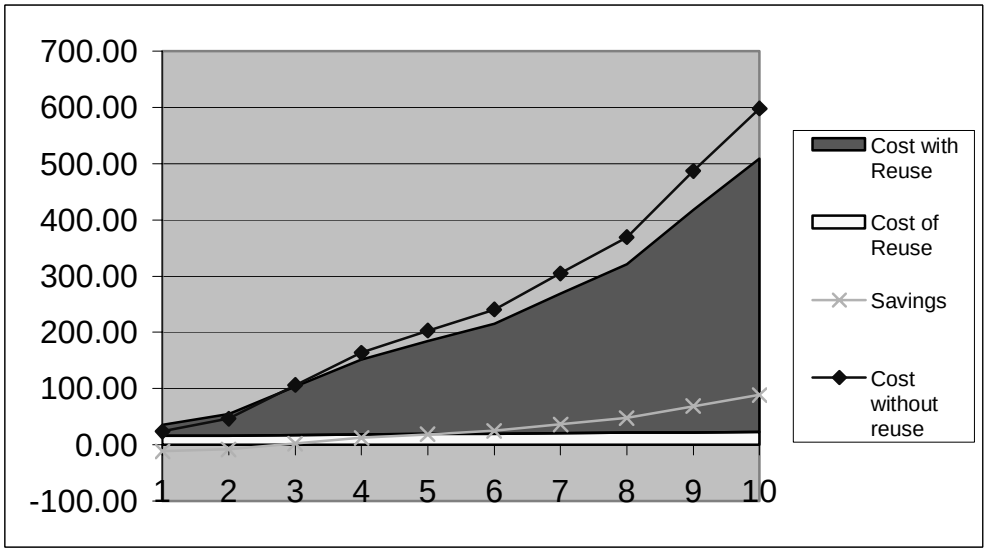


Figure 3: Cost Comparisons Under Scenario 1

Figure 4 illustrates the cost savings over time for the three scenarios. (Again, the numbers along the vertical axis represent millions of dollars, and the numbers along the horizontal axis represent years.) Scenario 2 shows increased savings due to a decreased COR. Scenario 3 shows the greatest eventual savings reflecting an increased DOR. However, there is a dip when the expenses for domain expansion come into play, between systems 4 and 7. The organization must be prepared to absorb this negative return until it recovers the cost of new asset development.

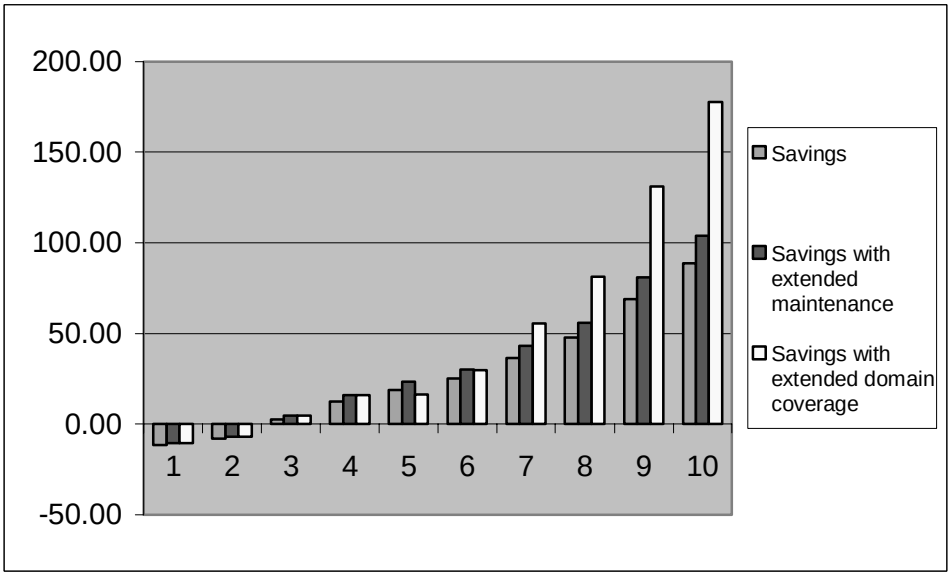


Figure 4: Cost Projections for Three Scenarios

In addition to these cost savings, the business case identifies certain non-monetary benefits. A strategic software reuse approach would

- Put the NRO back in front on technology.
- Allow the NRO to apply limited technical resources on the harder mission-unique problems.
- Reduce the perception of overlapping systems and of "re-inventing the wheel."
- Allow the adoption of best commercial practices.
- Align with top-level, government-technology directives.
- Align with the Space Object Technology Group (SOTG) and other standards work.
- Give the program manager more flexibility.

3.5.4 Strategies to Get the NRO to the Preferred State

This section of the business case recommends some strategies to get the NRO to the preferred state—a state where software product line practices have been institutionalized and the benefits of strategic software reuse are realized. The four broad strategies to pursue when implementing the chosen scenarios are to

- Fund and acquire core assets.
- Identify potential users and provide incentives for them to develop systems from core assets.
- Provide the infrastructure to sustain the effort.
- Fund the sustainment of core assets and products.

Naturally, the refinement and creation of detailed plans will be necessary to implement these strategies.

3.5.5 Conclusions

This section of the business case gives the conclusions, which include benefits and issues associated with implementing the product line strategies.

The bottom line is that the business-case exercise was a success. In March 2001, the Acquisition Steering Group accepted the business-case recommendation to proceed. The team is currently working on an implementation plan.

4 Software Product Line Practices: Working-Group Reports

Following the plenary presentations, workshop participants were divided among four working groups. Each working group discussed the companion in general and performed a more detailed analysis of three selected practice areas. The style and format of reporting varied among the groups. The following sections contain reports from these four working groups. Section 5 concludes with a summary of the general comments that surfaced during the working-group sessions.

4.1 Working Group 1—Customer Interface Management, Developing and Implementing an Acquisition Strategy, Testing

In addition to the general discussion of acquisition and the companion, this group addressed the practice areas of

- Customer Interface Management (built on the current draft of the companion)
- Developing and Implementing an Acquisition Strategy (built on the current draft of the companion)
- Testing (new input not in the current draft of the companion)

Each practice area was discussed in terms of its

- general description
- aspects that are related to single-system acquisition
- aspects that are related to product line acquisition
- acquisition-specific risks
- related practice areas

4.1.1 Customer Interface Management

Next the group discussed the “Customer Interface Management” practice area. First a general discussion was held on the practice area and the current companion draft. The group emphasized that the definition of the human interface assumes even greater importance in a product line because of the broad range of potential products, as well as the possibility of variation

points. The practice will also need to clearly define the roles of acquirer, customer, and end user, along with the relationships between these roles.

Several issues about users were discussed. Because a product line can span multiple organizations and products, the coordination of Cost as an Independent Variable (CAIV) issues with the customer and the prioritization of multiple change requests for core assets need to be coordinated. These processes are potentially more complex and lengthier for product lines. The group also suggested that users might be strong advocates and that strategies for harnessing this potential should be explored.

4.1.1.1 Description

The description was derived from the current draft of the companion.

> *"Customer interface management refers to the exchanges and relationships of the organization with the customer and other stakeholders. Specific individuals within the product line organization units are typically designated as customer points of contact. In addition, marketers will likely interface with customers. Protocols for dealing with customers will be set by organizational goals, standards, and procedures. The major customer-management issues that need to be addressed are*
>
> - *What will the customer see from the product line?*
> - *What are the ground rules for conducting business?*
>
> *Exchanges in communication should respond to the needs of the customers, but not be driven by customer demands.*
>
> *This practice area is closely coupled with "Requirements Engineering." Both practice areas are needed to ensure that the right requirements as well as the complete set of requirements are being addressed and managed."*

4.1.1.2 Aspects Related to Single-System Acquisition

In addressing customer interface issues for a single-system acquisition, the group underscored the importance of customer involvement through the entire acquisition and deployment life cycle. Next the discussion shifted to product line acquisition.

4.1.1.3 Aspects Related to Product Line Acquisition

The group reiterated the importance of making explicit in the request for proposal (RFP) and throughout the acquisition life cycle, how the user/system interface gets accomplished in the context of multiple products, and the potential of multiple human interface requirements for a product line. This is especially important because managing the customer interface is more complex when multiple organizations are involved.

Additional points on customer interface management for a product line acquisition include the following:

- The potential for sharing system-administration costs across users should be explored.
- A product line places constraints on users, and these constraints need to be spelled out clearly in the operational concept.
- Because of the central significance of the product line architecture, it is important to train the customer/acquirer in the use of architecture-evaluation techniques.
- How to incorporate new technology developments and Pre-Planned Product Improvements (P^3Is) into the acquisition should be considered.

4.1.1.4 Acquisition-Specific Risks

Customer interface risks include the following:

- Customers may focus on bells and whistles rather than the core assets and products of the product line.
- Customers could dictate a solution or operational concept based on current products. To mitigate this risk, they can be educated about the new potential that a product line offers.
- Changes in the funding profile, scope, organization, requirements, technology, and doctrine can all impact the product line, specific products, and the role of the customer.
- Users' expectations may be misaligned or unrealistic. To mitigate this risk, these expectations need to be actively managed on a continuous basis.
- The needs of different end users may diverge over time.
- There is a political risk that users may be sold on other solutions.

4.1.1.5 Related Practice Areas

The related practice area is "Requirements Engineering."

4.1.2 Developing and Implementing an Acquisition Strategy

Next the group discussed the "Developing and Implementing an Acquisition Strategy" practice area, using the current draft as a baseline. During the discussion, the group emphasized that a sound acquisition strategy often focuses on the identification and mitigation of risks and that acquisition decisions can be viewed as risk-mitigation strategies. The group suggested that the practice area should explicitly consider how risk mitigation impacts the acquisition strategy.

The discussion also addressed the fact that an acquisition strategy touches just about every other practice area. Clear entry and exit conditions for this practice area, and clear points of interaction with other practice areas need to be established.

4.1.2.1 Description

The description was initially derived from the companion draft. The group added a reference to the need for a mitigation strategy to mitigate risks and a paragraph that emphasizes the coordination of multiple projects.

The description statement was revised as follows:

> *"This practice area is concerned with organizations that elect to commission the development of core assets or products, or to commission the development of an entire product line.*
>
> *Since the DoD is primarily an acquirer of products and services, as opposed to being a developer (or occasional acquirer), choosing the right acquisition strategy is a key consideration in pursuing a product line initiative. An effective acquisition strategy is needed to mitigate the risks of acquiring technical products and services (from external sources) and integrating them into an organization's product line operations.*
>
> *Choosing to adopt a product line approach is a strategic, long-term decision. It involves coordinating multiple project efforts for the long-range acquisition of a family of related software systems. As a result, developing a suitable acquisition strategy is more critical and complex than that for a traditional system acquisition."*

4.1.2.2 Aspects Related to Single-System Acquisition

The discussion underscored the need for risk mitigation to be a major driver of the acquisition strategy and suggested that this emphasis be addressed explicitly. Other factors related to risk in developing an acquisition strategy include

- Mission-capability evaluations should be based on risks/structure.
- Evaluation criteria can be based on risks.
- The process for the management of risks needs to be addressed explicitly.
- Decisions about contract type can be influenced by the potential risks.

The group also recommended that a reference be added on how to factor the CAIV concept into the acquisition strategy.

4.1.2.3 Aspects Related to Product Line Acquisition

The discussion on product line acquisition continued to focus on the themes of risk and the need to involve multiple organizations in the acquisition process. Points relative to risk include

- Risk is a major focus of an acquisition strategy, and a product line approach has additional risks to manage because of multiple target systems.
- There is a need to identify risks and manage them explicitly as part of an acquisition strategy.

Issues on the involvement of multiple acquiring organizations include

- the need to get participation in the development of the business case and production plan among the acquiring organizations, to share these with each organization, and to keep the documents currents
- the need to coordinate with each organization throughout the long period of time that may be required to create a product line and to maintain the buy-in of each group
- the implications of multiple organizations on sustainment decisions

Other acquisition-strategy issues related to product line acquisition include

- The production plan, product line architecture, and business case need to be considered as drivers for the acquisition strategy.
- If different target systems have major variations, the acquisition strategy needs to be sensitive to requirements variability.
- Variation points built into the core assets of the product line can help to anticipate variability in target systems.

- The acquisition strategy needs to address how to protect the government from potential sole-source sustainment as it considers future products within the product line.
- A product line can often be developed incrementally. In this case, an incremental acquisition strategy needs to be developed.
- In order to promote greater innovation and to leave potential options open, it is necessary to share the business case with potential contractors during the pre-RFP phase.

4.1.2.4 Acquisition-Specific Risks

A number of potential risks were identified for product line acquisitions including

- There may be an increased potential for protests. It is important to factor the potential for a protest carefully into the acquisition strategy.
- There is a risk that multiple organizations will initially sign up to participate in the product line acquisition, but that their interests will diverge, and some will eventually drop out.
- There is a risk in owning and maintaining core assets.
- There are risks in providing core assets as government-furnished information (GFI).
- Because of the different approach required by a product line acquisition and the fact that there are not many contracting officers who are familiar with it, there is an increased risk of violating the Federal Acquisition Regulations (FAR).
- If contracting and legal organizations are not involved closely enough through the entire acquisition process, there is the risk that procedural or legal errors may occur.
- If the RFP does not allow for enough innovation, the resulting product line will not fulfill the product line vision.
- Protracted negotiations may increase the time for getting an RFP issued.
- There may be multiple versions of assets across different organizations.
- If the scope of the acquisition is not clearly understood, the product line effort may be too broad to implement, or it may be too narrow and not very useful.
- The sponsor may change, leading to a potential period of limbo, or a scaling back or canceling of the product line approach. A mitigating strategy would be to investigate whether a product line approach can be binding for a new program manger (PM). However, such an approach would have a new set of risks, which would need to be fully explored.

4.1.2.5 Related Practice Areas

As discussed in the introduction to this section, the "Developing and Implementing an Acquisition Strategy" practice area is linked to a number of other practice areas. Several were specifically brought up during the group's discussions. Although these are listed here, the group did not make an attempt to develop a systematic list.

- Funding
- Customer Interface Management
- Achieving the Right Organizational Structure
- Organizational Risk Management

4.1.3 Testing

Since the "Testing" practice area is not currently included in this draft of the companion, this discussion represents new input. The group used the "Testing" practice area from the framework as a reference point and discussed testing for a product line from an acquisition perspective.

4.1.3.1 Description

A general discussion was first held on issues that would fit into the description section and drive down throughout the practice area. The points of that discussion included:

- All testing artifacts (including test plans, test cases, and test scripts) should be listed in an RFP as documents and deliverables to be approved.
- The levels of testing need to be listed in an RFP and monitored during contract execution, including unit, subsystem, system deployment, and regression testing.
- The Test and Evaluation Master Plan (TEMP) spells out the roles and responsibilities of all parties during testing.
- The role of testing relative to other life-cycle activities needs to be defined.

4.1.3.2 Aspects Related to Single-System Acquisition

The role of testing in a single-system acquisition is crucial. Since the acquisition organization has an oversight role rather than a development role, it needs to monitor the testing process effectively. Testing-related issues that need to be addressed in the RFP include

- development of a high-level test plan
- description of the test process and how to monitor it
- costing testing separately in the cost proposal
- assurance that testing addresses both functional requirements and quality attributes

- use of testing metrics such as the analysis of defects
- architecture evaluation prior to testing
- tests for conformance to the Defense Information Infrastructure Common Operating Environment (DII COE) and JTA interoperability testing

4.1.3.3 Aspects Related to Product Line Acquisition

Testing issues become more complex in a product line acquisition. Because of the number of potential systems and the interplay of technical and organizational issues, testability for a product line acquisition needs to be addressed early. Agreements need to be in place on sharing all test artifacts when multiple organizations are involved in a product line acquisition. Because system testing spans multiple products, the responsibilities for testing each product need to be made clear.

From an acquisition perspective, the RFP needs to specify that a test plan be developed and that a test process will be followed. There needs to be specific ways to monitor the process and to evaluate test artifacts. The test plan for the product line needs to address a set of issues and to make artifacts visible for evaluation including

- verification and validation of the product line architecture
- testing of all core assets
- evaluation of reusable test assets
- interactions among all core assets
- testing of variation points
- version compatibility of tools, such as configuration-management tools
- performance testing of specific products

4.1.3.4 Acquisition-Specific Risks

The acquisition-specific risks related to testing include

- inadequate verification of architecture and core assets
- inadequate requirements that do not allow appropriate testing
- inadequate schedule and budget in which resources for testing are cut
- lack of appropriate analysis of the impact of external policies (such as spiral development, evolutionary acquisition, or CAIV) on testing
- inability to obtain the appropriate level of user involvement in the testing and acceptance of product line products
- not getting approval for key test documents
- not getting or not making decisions about the resources required for adequate testing

4.1.3.5 Related Practice Areas

The related practice areas are

- Data Collection, Metrics, and Tracking
- Technical Risk Management
- Component Development
- Customer Interface Management

4.2 Working Group 2—Architecture Definition, Product Line Scoping, Understanding Relevant Domains

This working group discussed the acquisition companion in general and three practice areas in particular:

1. Architecture Definition
2. Product Line Scoping
3. Understanding Relevant Domains

For each practice area, the group discussed its

- aspects that are related to single-system acquisition
- aspects that are related to product line acquisition
- specific practices
- risks

4.2.1 Architecture Definition

This practice area was not included in the draft of the companion. However, the group felt that it was so important it should be included in version 1. Thus, what follows are original suggestions about the contents of the practice area. In cases where the answers were not evident, the important questions were at least identified.

4.2.1.1 Aspects Related to Single-System Acquisition

The group identified these aspects:

- specifying the skill set for an architect or an architecture team
  - The acquirer must specify the requirements for appropriate competence on the contractor team.
  - The acquirer must know how to recognize and evaluate the skills in proposals.

- maintaining corporate knowledge of the architecture
  - How do you maintain this critical corporate knowledge?
  - It may be necessary to require a knowledge-dissemination plan.
- knowing when the architecture is “good enough”
  - How do you know this?
  - Planning and prototyping for architecting before design must be specified.
  - This has an impact on the funding and staffing plans and delivery schedule.

4.2.1.2 Aspects Related to Product Line Acquisition

All of the elements of single-system acquisition are relevant here. However, they are more complex and even more important to address in architecture definition than in other practice areas.

4.2.1.3 Specific Practices

Participants shared the following specific practices that had been applied in their work experience:

- maintenance of the Core Architecture Team: Continuity of architectural knowledge was maintained through this team. Thus, the team’s composition, size, and term of assignment were planned carefully.
- architecture apprentice program: Related to the above practice was a program to specifically “grow” new architects. Less experienced staff were trained in “how to be an architect” and in the architecture of the product lines. These people, who could eventually serve on the core team, performed risk mitigation against the loss of staff.

4.2.1.4 Risks

The primary risk was seen as the loss of corporate architecture knowledge. The specific practices above suggest two successful risk-mitigation strategies.

4.2.2 Product Line Scoping

4.2.2.1 Aspects Related to Single-System Acquisition

In an acquisition context, the responsibility for scoping is in the domain of the acquirer. While scoping a single domain might be relatively easy, scoping across domains is difficult. This is due to the need for interfacing with stakeholders from different organizations with potentially conflicting views. Also communicating technical issues across domains is more difficult. It may be necessary to augment the acquisition staff with technical expertise to perform a scoping investigation.

4.2.2.2 Aspects Related to Product Line Acquisition

The group mentioned several additional issues that arose when scoping a product line acquisition:

- A study of quality attributes is useful to help determine the scope. Since quality requirements often pull in different directions (e.g., response time versus security), it is useful to determine the acceptable ranges of quality attributes to determine the feasibility of coexistence.
- The technical suitability of scoping is not the whole story. The alignment of organizations that may share products in a product line is equally important.
- When scoping a product line you should, "Think big and start small."
- A scoping plan is important and should address
  - alignment with windows of opportunity among potential participants
  - criteria for specifying which systems to consider
  - criteria for specifying how to rule a system in or out
  - the possible compromises that would bring the projects into scope (This ties to requirements engineering.)
- The separation of concerns must be strongly enforced. (This ties to architecture definition.)

4.2.2.3 Specific Practices

Participants shared the following specific practices that had been applied in their work experience:

- Plan to evolve your scope. This can be particularly successful if you start with at least three systems. These initial systems should stretch across different dimensions of the product line design space.
- Consider two types of scoping and handle them differently.
  - Some candidates are "no brainers." However you should still use your brain somewhat to confirm a match (or mismatch) to requirements.
  - Other candidates require the exploration of boundaries. To do this, you need a preliminary architecture to test the ideas. Consider that the following three activities should be executed iteratively: scoping, requirements engineering, architecture definition and evaluation.
- Plan to "de-scope" if necessary. This is a risk-mitigation activity that can allow you to respond to funding or schedule crunches.

4.2.2.4 Risks

The group identified two main risks:

1. Business pressures might try to force the scope of the product line to be too large. This can happen in a situation with a push for standardization where such standardization compromises stakeholders' needs.
2. You might "back into" a product line approach. Usually in this case, the organization has not used the rigor or discipline needed to succeed.

4.2.3 Understanding Relevant Domains

As with "Architecture Definition," this practice area was not included in the draft of the companion. However, the group wanted to provide input for a future version of the companion.

4.2.3.1 Aspects Related to Single-System Acquisition

The group noted that this practice area had a strong relationship to the "Product Line Scoping" practice area and felt that a team of stakeholders (including both acquirers and suppliers) should be formed to represent the breadth and depth of the domains being considered. Since the supplier must have technical-domain knowledge, there must be a means for specifying this, selecting a supplier with the appropriate qualifications, and making sure that this knowledge is maintained.

4.2.3.2 Aspects Related to Product Line Acquisition

The main point of the discussion was, "How should the domain understanding be captured and maintained for the benefit of the acquirer?" The framework says that a domain model is not required, but how can the maintenance of this knowledge be assured over the life of the product line when suppliers might change? In discussing whether or not to require a domain model as a deliverable, these points were raised:

- A domain model is important to make the acquirer less dependent on a sole-source supplier.
- There is no standard notation for documenting domain knowledge.
- Even when you have a domain model, it is typically difficult to achieve common understanding even within the team that developed it.
- There would be a large investment necessary to create the extensions and tools needed to support the model.

The bottom line was the group felt that a deliverable domain model was necessary, but in what form and at what cost?

4.2.3.3 Specific Practices

While the preceding discussion raised some difficult questions, the group had some specific practices to provide some answers:

- Use a domain model as a top-level architecture. It is possible to represent a domain model that is the equivalent of a top-level architecture. This approach could use the same modeling techniques and support tools, which has been successful in addressing interoperability concerns.
- Use a product-domain representation. It is possible to create a domain model that represents the space of products in the product line and which areas of the domain they address. This might fall short of modeling the entire domain but can address its practical aspects.

4.2.3.4 Risks

The most obvious risk was the lack of sufficient domain understanding. Consideration of this risk was a sobering experience. Since domain knowledge is essential to product line success and it is difficult to represent and maintain over the life of the product line, where does that leave an acquirer? Risk mitigations might include similar strategies for maintaining architecture knowledge.

4.3 Working Group 3—Requirements Engineering, Developing and Communicating a Business Case, Technology Forecasting

This working group discussed the companion in general and three practice areas in particular:

- Requirements Engineering
- Developing and Communicating a Business Case
- Technology Forecasting

The discussion of the "Requirements Engineering" practice area led to the discussion of the other practice areas. The main question to address was, "What's different about the requirements engineering of product lines for an acquisition organization?"

4.3.1 Global Companion and Acquisition Issues

The initial discussion looked at global issues concerning the companion. Suggestions for issues to address include

- providing a clear delineation among the concerns of single-system acquisitions, acquisitions of product line assets, and acquisitions of products built from the assets
- providing additional examples and case studies
- assisting the acquirer's understanding of how variability affects the practice areas
- providing a discussion of how to orchestrate the product line essential activities and how the practice areas contribute.

4.3.2 Requirements Engineering

Obtaining user input for requirements is critical. An operational concept (CONOPS[10]) should be developed to describe how the product line organization should function. The CONOPS should serve as a guide for requirements development by providing an understanding of how the assets and systems will be used. This CONOPS may include scenarios for normal use or exceptional use. There must be sufficient, appropriate documentation with the assets to allow potential users to determine whether the assets will meet their needs.

Requirements engineering should emphasize the decision process for defining product line requirements in terms of

- functionality
- flexibility through variation points to handle varying needs and changes
- performance
- configuration management
- sustainability

Verification should be an important requirements engineering practice for the developer. The acquirer must ensure that verification is performed on core assets and on products that use those assets. Without adequate verification, the assets will never be considered safe for use on new systems.

[10] The traditional view of a CONOPS is that it represents the user's view of how a system will operate in its intended environment. A product line CONOPS describes the business processes and organizational structures for fielding and sustaining a product line and how products will be developed using the core assets.

Specific to requirements engineering issues, the group discussed the coverage of assets. This requirement for assets may constitute a superset of assets required for building a product or may only offer a subset for development. While the determination of coverage is a scoping issue, the acquirers must take into account the distinction between using assets for parts of a system versus using the asset base for the whole product. Requirements engineering also varies depending on whether you are developing the core assets, building products in the product line, or buying an instance of the product line from an outside group.

There is a distinction in requirements engineering between using the assets and building them. The requirements should address the entire set of core assets and also indicate how the rest of the system is built around the core. While the budget and the knowledge of what can be successfully built may drive product line scoping, the requirements must cover the entire suite of products, defining what's in and out of scope for the assets.

The group discussed three alternative approaches for arriving at the requirements for a product line:

1. Develop single-product requirements with a goal of spinning off common asset requirements. Sharing assets can be accomplished through this approach rather than through interoperability, as is currently required. One approach may be to work from a single system with the knowledge of future systems and to go back and identify what assets from the single system will become core assets for future systems.
2. Develop asset requirements in terms of the needs for systems the organization knows it will be building in the near term. This approach seeks the union of requirements used to build individual products. A contractor can then build assets to satisfy these requirements.
3. Develop requirements about common capabilities, variation, and architecture drivers in order to satisfy the needs of all potential asset users. Keen insight is required for systems not yet built and for future requirements. There is a practical limit on the time horizon for how well this can be accomplished.

The group further discussed requirements engineering issues in light of cyclic, evolutionary, development processes. Under a cyclic development approach, product line requirements evolve in anticipation of the product requirements. The scope may evolve as well, broadening to cover new domains or new areas within a domain. Using this approach there would be less up-front investment in asset requirements, because development would occur incrementally.

Requirements evolution that is too rapid is a significant risk for product lines. If product-specific requirements evolve too rapidly, assets cannot keep up the pace. If the core requirements evolve too rapidly, it is difficult to incorporate the assets into products. Technology forecasting can mitigate some of these effects. (The "Technology Forecasting" practice area is covered in Section 4.3.4.)

An evolutionary approach must be tied to plans for system delivery. Where there are simultaneous deliveries of systems, there must be more up-front work to identify the commonalities and plan for variability. The sequential delivery of systems in the product line allows a more evolutionary approach and may support developing requirements via mining requirements from each delivery.

An evolutionary approach may lead to too many dependencies for the program manager to control effectively. The combination of multiple programs with evolving system requirements as well as an evolving asset base creates dependencies between programs and between the assets and the product line, impacting the programs that depend on these releases. For example, the DII COE, a collection of COTS and government-furnished equipment (GFE) assets, places some restrictions on building systems. Because of COTS tools in the DII COE, the programs using it must contend with many dependencies. Sometimes, good software engineering practices can mitigate some of these evolutionary effects. For example, CCT assets were designed for late binding to a specific operating system or object request broker (ORB).

The group discussed the usefulness of Integrated Product Teams (IPTs) for performing requirements engineering activities. The IPT should set the asset requirements, requirements for transitioning from assets to products, and requirements evolution.

Finally, the group discussed the difficulties associated with performance requirements. Variations to accommodate different needs are a feature of asset bases that cover a superset of the requirements for any one system. However for a subset asset base, there is not the same level of guarantee for meeting performance or other quality factors. How do you define performance at the core level when performance will vary as variation points are added? In an extreme case, performance parameters may not be known at all, for example, when a COTS component is used. Performance requirements may require an organization to build an asset from scratch to have a necessary understanding of its performance. Possibly the asset base can include tools to support a performance assessment with variation points highlighting potential tradeoffs. The asset base should also support the expression of performance requirements in more concrete terms. For example, rather than specifying “minimize latency” as a requirement, express the requirement in terms of the range of response rates provided by the assets. For performance requirements, the companion should include specific recommendations to guide acquisition groups in enhancing the usability of assets. The companion may achieve this by enumerating the types of support that the asset base should provide to prospective users.

4.3.2.1 Risks

There are a number of risk areas associated with requirements engineering:

- A critical concern is the synchronization of asset (or COTS) availability with the production system's needs. If the assets are not ready when a product is under development or requirements have evolved since creating the asset base, the value of the asset base declines.
- What happens when an organization loses business or the core-asset groups do not market themselves well enough? The asset group may go out of business and leave dependent programs without support. Then what happens to those who rely on the assets group? Flexible requirements for the target system may allow for the substitution of alternative assets, but the target-system analysis must include the cost effectiveness of waiving certain requirements. A product line perspective supports better up-front planning for target systems and the ability to define the impact of change.
- What happens when assets become obsolete and are no longer maintained? Market changes or the development of other COTS products may make assets obsolete. How do you determine which legacy assets should be maintained as part of the core assets? Some requirements remain unchanged during the life of a product, and the asset base can continue to support them. However, if those same requirements in other products of the product line evolve, the asset base may be required to maintain assets for only a single user and build new assets for the rest. This increases the costs of reuse and strains the financial return. A requirements-review board must assess the impact and business case of maintaining legacy assets in the face of evolutionary requirements.

4.3.3 Developing and Communicating a Business Case

The group identified several issues associated with this practice area:

- It is important to make a distinction among the concerns of single-system acquisitions, the acquisitions of product line assets, and the acquisitions of products produced from the assets.
- It is important to know cost drivers that contribute to the decision process. What are the cost drivers for single-system acquisitions, and do they vary from those leading to product line decisions? Are the cost drivers the same for all product lines, or will specific markets or mission areas lead to different sets of drivers?
- There are important issues associated with the use of government off-the-shelf (GOTS) components. The companion should expand on the risks, costs, capability, and quality of GOTS components, and contrast their use in single-system versus product line development.
- Cost will remain the primary factor in selling product line concepts to the DoD. To cover other benefits of product lines, the companion could offer a broader set of issues for evaluating proposals. *Best-value contracting* may be a term to use for generating the additional metrics and evaluating results that can deliver lower costs, improved quality, dependable schedules, and enhanced maintainability. In the O&M world, it is important to include the cost of developing and evolving assets over time. A five-year business case

does not go out far enough for the DoD. A life-cycle business case for product lines may need to span 20 years to cover O&M activities adequately.

- A business case is basically cost, schedule, performance, and supportability. Therefore the business case for product lines should consider issues other than the development and use of assets. A business case may be made in terms of justifying a new process for building assets or products. The justification would be better value for your money. Single-system business cases may be made for justifying a requirement or justifying why a legacy system no longer meets the requirement. Will the business case for a product line explore similar issues? The business case should also include scheduling information, which accounts for dependencies with other programs.
- The audience for the business-case practice and its products is very broad. In the DoD the business case may address multiple services and the benefits that cross branches. The audience may include Congress because of its control on budgetary authority. The contractor community at large must also be convinced. They must buy into the business case and believe in it. Their business-case drivers will often be different from those of the government.
- The business case may emphasize flexibility in contracting so that third-party developers can be brought in and the DoD is not locked into a proprietary solution. Currently, some Air Force groups are building their own in-house solutions on top of general-purpose office tools and are not as dependent on contractor support as they have been in the past. The DoD needs a policy allowing supplier selection based on government needs rather than the need for competition. The practice area should emphasize building business cases that reflect the need to grow and evolve, as opposed to today's contracting that too often presents a static situation that cannot effectively respond to change.

4.3.4 Technology Forecasting Issues

Today's contracting approaches do not allow work on requirements that are not specified in the contract. While it is well known that systems will evolve and that planning for this is a must, the current acquisition process seems at odds with this reality. Technology forecasting should be a factor in all contracts to match the mission to the technology, subject to the budget and other constraints and what is technically possible. While we know that technology will evolve, we don't know exactly how. Too often, the contract locks in a technical solution that is obsolete by the time the system is developed. The practice area for the companion should address contracting issues and how to delay technology decisions when they may pose impediments to successful evolution.

A related issue is technology insertion. The broader impact of product lines may be to anticipate the needs of future systems, yet legacy systems must also be brought up to speed to take advantage of product line innovations. While technology forecasting makes a valuable contribution to the decision process, without the ability to insert that technology, potential benefits will be lost.

Most times, you cannot predict how requirements will change. How do you justify your budget based on the unknown? While the forecast can identify trends, the companion must

tell organizations which methods take advantage of the cost savings or other benefits of technological advances. In theory, product lines should save money by inserting technology developed at one place into multiple places, without multiple acquisition and development efforts.

The time allotment for the breakout groups ended without fully discussing the "Technology Forecasting" practice area. This subject should be covered in more detail at future workshops.

4.4 Working Group 4—Architecture Evaluation, Funding, Software System Integration

This working group discussed three companion practice areas:

- Architecture Evaluation
- Funding
- Software System Integration

The group not only provided insight into good practices for the DoD but also gave valuable feedback on the companion itself. Because the feedback on the companion is primarily useful to the authors of this report, much of this information was placed in the appendix.

4.4.1 Architecture Evaluation

Acquisition managers need to understand how architecture evaluations can be used in a product line acquisition. Some of the points associated with architecture evaluations include the following:

- Architecture evaluations can be used to identify and resolve tradeoffs in an acquisition. This is with respect both to a single system (or product) from the product line and to the product line as a whole.
- The requirements for architecture evaluations must be included in an RFP or contract. A related concern is the need to specify the requirement for an architecture in an RFP. A difficult issue is how to ensure that an architecture has the appropriate form and content to permit an evaluation.
- Architecture evaluations must focus on critical operational requirements. Care is required to ensure that the focus does not drift to secondary requirements.

4.4.2 Funding

With regards to funding product line acquisitions, the group agreed that existing DoD processes and practices, on the surface, appear somewhat hostile for adopting and funding a product line approach. Yet in the DoD, there are workable exceptions—creative, legal ways to initiate and sustain a product line approach.

There are important interconnections with the "Developing and Communicating a Business Case" practice area. Obtaining funding requires the development of a business case that identifies appropriate cost data to substantiate the business case (with appropriate funding-strategy considerations). That is, an effective business case is a necessary prerequisite to obtaining funding for a product line. Analogously, a feasible funding strategy is an important element of the business case.

The working group discussed ways to facilitate product line acceptance and implementation within the DoD, particularly in terms of how to obtain funding for product line efforts:

- Change DoD policies and regulations to explicitly provide for the use of a product line approach in appropriate situations.
- Change the FAR to provide for the evaluation of product line opportunities as part of acquisition checklists.
- Put in place a reward system to stimulate the government's adoption of a product line approach when appropriate.
- Create a position of "product line advocate" in DoD program organizations, with the responsibility for promoting the consideration, adoption, and use of product line practices.

4.4.3 Software System Integration

The "Software System Integration" practice area was not included in the draft companion. As a result, discussions in this area only raised several issues to consider:

- The acquirer should have insight into the quality of a developer's production plan that incorporates the integration aspects of components for a product line. An open issue is how to evaluate this quality.
- This practice area is related to the "Component Development" practice area. The acquirer should have insight into how to evaluate and improve the quality level of a component to ensure that it meets suitable criteria for its use and integration into the product line.
- Relative to integration aspects of product lines, the roles and responsibilities of the acquirer can vary considerably depending upon the acquirer's overall role and responsibility with respect to acquiring and managing a product line. For example, should the acquirer assume responsibility for the integration aspects of a product line, perhaps implemented through an IPT?

5 Summary

The Fourth DoD Product Line Practice Workshop explored the product line practices of organizations in the DoD community in light of best commercial practices in software product lines. The presentations and discussions again validated the pivotal pieces of the framework and provided valuable feedback on and content for the draft companion. Challenges and solutions within the DoD community were discussed.

The working groups focused on the treatment in the companion of specific practice areas within software engineering, technical management, and organizational management. As in the previous two such workshops, the empirical and anecdotal evidence that the workshop participants brought to the discussion significantly enhanced our current understanding of the practices and issues as they apply to the DoD. The following general comments that surfaced during the workshop added new clarity to our thinking and are certainly valuable to share with the readers of this report:

- The hardest part about adopting software product lines within the DoD is to sell the concept to the stakeholders. We are asking both System Program Offices (SPOs) and contractors to move from the business models and concepts that they are familiar with to something relatively unknown. Changes of this nature will always encounter resistance. We need to provide change agents with appropriate ammunition, guidance, and hints. We need to provide solid answers to the question, "What's in it for me?" "Evangelizing" must be carefully planned and coordinated.
- Different acquisition organizations have different organizational cultures. What works in one culture may not work in another. Important factors include
  - How actively involved is the acquirer in the technical details? Is all technical activity outsourced, or is government staff used in any capacity? If so, in what capacity?
  - Does the approach align with the acquisition organization's structures and processes? Some organizations might be better suited to an incremental approach to evolve to a product line. Others might be oriented toward new starts.
- Process maturity is an important factor.
- Incorporating existing systems into a product line is often most realistic when a window of opportunity opens for the replacement of a system.
- Informal, cooperative agreements can sometimes work better than formal agreements. This has to be watched carefully to avoid getting burned. Some things must be done more formally (e.g., activities of the core-asset team).
- More time must be spent up front envisioning future opportunities for a product line. Scoping the product line is key.

- Because product lines cut across projects and support multiple products, the companion ought to emphasize interfaces across projects and SPOs as well as the acquisition implications of working with multiple organizations.
- While many acquisitions are focused at the level of a PM, the program executive officer (PEO) may be a more appropriate target for product line acquisitions because the PEO is usually responsible for multiple products.
- Since the business and benefits of a product line extend throughout the entire life cycle of a set of products, including O&M, it is important for all practice areas to consider the entire life cycle, and not just development.
- Acquisition strategies for product lines are usually focused on incentives that the government may offer to a contractor to move in a product line direction. In addition to these incentives, government organizations should consider how to encourage the contractor to use a product line approach because it is in the contractor's best interest.
- The ownership of assets is a critical issue and needs to be an early decision of each product line acquisition.

Within the DoD there needs to be increased awareness about the DoD product line activities that may be relevant. It is critical for the DoD to think more strategically and to share information and outcomes between different areas. These outcomes could help to prevent duplication and redundant development.

In an effort to expand both the information base and the DoD community interested in software product lines, the SEI was encouraged by the participants to continue to hold similar workshops.

The results of this workshop have been incorporated into the companion, which will continue to be refined and revised as the technology matures and as we continue to receive feedback and work with the growing acquisition community championing a product line approach. If you have any comments on this report and/or are using a product line approach in the development or acquisition of software-intensive systems for the DoD and would like to participate in a future workshop, please send email to Linda Northrop at lmn@sei.cmu.edu.

References/Bibliography

[Bass 97] Bass, L.; Clements, P.; Cohen, S.; Northrop, L.; & Withey, J. *Product Line Practice Workshop Report* (CMU/SEI-97-TR-003, ADA327610). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. <http://www.sei.cmu.edu /publications/documents/97.reports/97tr003/97tr003abstract.html>.

[Bass 98a] Bass, L.; Clements, P.; & Kazman, R. *Software Architecture in Practice*. Reading, MA: Addison-Wesley Publishing Co., 1998.

[Bass 98b] Bass, L.; Chastek, G.; Clements, P.; Northrop, L.; Smith, D.; & Withey, J. *Second Product Line Practice Workshop Report* (CMU/SEI-98-TR-015, ADA354691). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. <http://www.sei.cmu.edu/publications/documents/98.reports /98tr015/98tr015abstract.html>.

[Bass 99] Bass, L.; Campbell, G.; Clements, P.; Northrop, L.; & Smith, D. *Third Product Line Practice Workshop Report* (CMU/SEI-99-TR-003, ADA361391). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. <http://www.sei.cmu.edu /publications/documents/99.reports/99tr003/99tr003abstract.html>.

[Bass 00] Bass, L.; Clements, P.; Donohoe, P.; McGregor, J.; & Northrop, L. *Fourth Product Line Practice Workshop Report* (CMU/SEI-2000-TR-002, ADA375843). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu /publications/documents/00.reports/00tr002.html>.

[Bergey 98] Bergey, J.; Clements, P.; Cohen, S.; Donohoe, P.; Jones, L.; Krut, R.; Northrop, L.; Tilley, S.; Smith, D.; & Withey, J. *DoD Product Line Practice Workshop Report* (CMU/SEI-98-TR-007, ADA346252). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. <http://www.sei.cmu.edu /publications/documents/98.reports/98tr007/98tr007abstract.html>.

[Bergey 99a] Bergey, J.; Campbell, G.; Clements, P.; Cohen, S.; Jones, L.; Krut, R.; Northrop, L.; & Smith, D. *Second DoD Product Line Practice Workshop Report* (CMU/SEI-99-TR-015, ADA375845). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. <http://www.sei.cmu.edu/publications/documents /99.reports/99tr015/99tr015abstract.html>.

[Bergey 99b] Bergey, J.; Fisher, M.; Jones, L.; & Kazman, R. *Software Architecture Evaluation with ATAM in the DoD Systems Acquisition Context* (CMU/SEI-99-TN-012, ADA377450). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. <http://www.sei.cmu.edu/publications/documents/99.reports /99tn012/99tn012abstract.html>.

[Bergey 00a] Bergey, J.; Campbell, G.; Cohen, S.; Fisher, M.; Jones, L.; Krut, R.; Northrop, L.; O'Brien, W.; Smith, D.; & Soule, A. *Third DoD Product Line Practice Workshop Report* (CMU/SEI-2000-TR-024, ADA387259). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu /publications/documents/00.reports/00tr024.html>.

[Bergey 00b] Bergey, J.; Fisher, M.; Gallagher, B.; Jones, L.; & Northrop, L. *Basic Concepts of Product Line Practice for the DoD* (CMU/SEI-2000-TN-001, ADA375859). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon, University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports /00tn001.html>.

[Bergey 01a] Bergey, J. & Goethert, W. *Developing a Product Line Acquisition Strategy for a DoD Organization: A Case Study* (CMU/SEI-2001-TN-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications /documents/01.reports/01tn021.html>.

[Bergey 01b] Bergey, J.; O'Brien, W.; & Smith, D. *Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets* (CMU/SEI-2001-TN-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu /publications/documents/01.reports/01tn013.html>.

[Bergey 01c] Bergey, J.; O'Brien, W.; & Smith, D. *DoD Software Migration Planning* (CMU/SEI-2001-TN-012). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tn012.html>.

[Bergey 01d] Bergey, J.; Campbell, G.; Cohen, S.; Fisher, M.; Gallagher, B.; Jones, L.; Northrop, L.; & Soule, *A. Software Product Line Acquisition - A Companion to a Framework for Software Product Line Practice, Version 1.0* [online]. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, August 2001. <http://www.sei.cmu.edu/plp/companion.html>.

[Brownsword 96] Brownsword, L. & Clements, P. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016, ADA315802). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. <http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.016.html>.

[Buschmann 96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; & Stal, M. *Pattern-Oriented Software Architecture: A System of Patterns*. New York, NY: Wiley, 1996.

[Chastek 01] Chastek, G.; Donohoe, P.; & Kang, K. *Product Line Analysis: A Practical Introduction* (CMU/SEI-2001-TR-001, ADA393296). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tr001.html>.

[Clements 00] Clements, P. & Northrop, L. *A Framework for Software Product Line Practice, Version 3.0* [online]. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, September 2000. <http://www.sei.cmu.edu/plp/framework.html>.

[Clements 01a] Clements, P.; Donohoe, P.; Kang, K.; McGregor, J.; & Northrop, L. *Fifth Product Line Practice Workshop Report* (CMU/SEI-2001-TR-027, ADA375843). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2001. <http://www.sei.cmu.edu/publications/documents/01.reports/01tr027.html>.

[Clements 01b] Clements, P. & Northrop, L. *Software Product Lines: Practices and Patterns.* Boston, MA: Addison-Wesley, 2001.

[Cohen 00] Cohen, S.; Gallagher, B.; Fisher, M.; Jones, L.; Kurt, R.; Northrop, L.; O'Brien, W.; Smith, D.; & Soule, A. *Third DoD Product Line Practice Workshop Report* (CMU/SEI-2000-TR-024, ADA387259). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications /documents/00.reports/00tr024.html>.

[Donohoe 00] Donohoe, P., ed. *Software Product Lines: Experience and Research Directions: Proceedings of the First Software Product Lines Conference (SPLC1).* Denver, CO, August 28-31, 2000. Boston, MA: Kluwer Academic Publishers, 2000.

[Kazman 00] Kazman, R.; Klein, K.; & Clements, P. *ATAM: Method for Architecture Evaluation* (CMU/SEI-2000-TR-004, ADA382629). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents /00.reports/00tr004.html>.

Appendix Feedback on the Draft Companion from Working Group 4

This working group provided excellent feedback on the draft companion. Because this information is of primary value to the authors of this report, it was placed in this appendix.

Overall Comments and Recommendations

The working group made several recommendations regarding the content and format of the companion. These recommendations concerned how to increase consistency with the framework and how to better focus the companion on acquisition concerns and an acquirer's needs and perspective.

- Remove the "Acquisition of Single Systems" section for each practice area. This section was meant to identify a common basic understanding of relevant existing acquisition practices being taken as the basis for undertaking acquisition in a product line context. Instead, it distracts from the intended focus on product lines and communicating how this focus affects acquisition practices. The intended audience (acquisition managers) does not need this level of detail unless the companion is proposing changes to existing practices.
- Make the structure of practice area sections more like that found in the framework. For example, differences in the acquisition of core assets versus the acquisition of products are not discussed separately as they are in the framework. The concern was that such differences would be overlooked if they were covered for both acquisitions in one discussion. However, discussing them separately may be largely redundant if differences are small, so the issue hinges on whether there are in fact significant differences between them.
- The companion needs to provide more detailed guidance on the practices to be used when acquiring a product line. An acquisition manager understands the acquisition practices for single systems but not how they apply to product lines. A suggested way to provide more depth was to add a "Frequently Asked Questions" section to each practice area, in order to provide additional insight specific to that practice area. This could also serve in part as a more accessible and enlightening substitute for a description of practice-area risks as provided in the framework.
- There are a large number of potential linkages among practice areas in the companion. Each practice area should identify these linkages and interdependencies and spell them out clearly.

- Statements of Objectives (SOOs) and Statements of Requirements (SORs) should be integrated for each practice area.
- Because customers and users within the DoD can be broad and diverse, DoD product line acquisitions need to consider the needs of all the potential customers of product line products. This underscores the importance of the "Operations" and "Customer Interface Management" practice areas, as well as their linkages to other practice areas.
- The assumptions for each practice area in the companion should be identified and made explicit.
- The companion needs to reflect the new interim version of DoD 5000-2R.
- A product line acquisition has implications for process improvement efforts and Capability Maturity Model® (CMM®) frameworks. These implications should be analyzed and made explicit.

Architecture Evaluation Comments and Recommendations

The "Architecture Evaluation" practice area needs to give acquisition managers an understanding of how architecture evaluations can be used in a product line acquisition. The companion should not provide a tutorial on architecture evaluation in general but instead should point readers to the framework. Suggestions on how to provide more detail specific to architecture evaluation in a product line acquisition were considered:

- Add a decision tree for this practice area to help an acquirer see and decide what type of architecture evaluation may be applicable to a particular acquisition. This should indicate also where in the acquisition process a particular evaluation method would be most applicable.
- Provide short definitions of architecture-evaluation methods in the companion focusing on their applicability in an acquisition. This should refer readers to the framework for more detailed or technical descriptions.
- Stress how architecture evaluations can be used to identify and resolve tradeoffs in an acquisition. This should be discussed with respect both to a single system (or product) in the product line and to the product line as a whole.
- Explain how an acquirer can add architecture evaluations to an RFP or contract. This practice area should reference existing technical notes that provide details of how this inclusion can be accomplished. A secondary discussion concerned how to specify a requirement for an architecture in an RFP/contract and uncertainty about how to ensure that an architecture had the appropriate form and content to permit an evaluation. Addressing this point was noted as a concern for the "Architecture Definition" practice area.
- Emphasize the need for architecture evaluations to focus on critical operational requirements. There was a concern that the discussion of architecture evaluation should focus primarily on the architecture that properly addresses those requirements, and only secondarily on whether it responds to other requirements.

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

Funding

The companion lists several funding strategies that appear to be workable within the DoD. However, the appropriate funding strategy may be somewhat unique to each particular organization. The companion should annotate this list of strategies with references to situations where each has worked, as a guide for acquisition managers needing to choose among them.

The companion should emphasize the interconnection between this practice and the "Developing and Implementing a Business Case" practice area.

Software System Integration

The "Software System Integration" practice area was not included in the draft companion. As a result, discussions in this area only raised several issues to consider:

- Discuss, in the companion and possibly in the framework, the quality of a production plan that incorporates the integration aspects of components for a product line. In turn the companion must provide guidance on how to evaluate this quality.
- In the "Component Development" practice area of the companion, discuss how to evaluate and improve the quality level of a component to ensure that it meets suitable criteria for its use and integration into the product line.
- Relative to integration aspects of product lines, the roles and responsibilities of the acquirer can vary considerably depending upon the acquirer's overall role and responsibility with respect to acquiring and managing a product line. For example, consider whether the acquirer should assume responsibility for the integration aspects of a product line, perhaps implemented through an IPT.

This discussion highlighted a key issue for the companion as a whole, establishing the nature of an acquirer's role for a product line approach.

Glossary

acquisition — the process of obtaining products and services through formal agreement

acquisition strategy — a plan of action for achieving a specific goal or result by obtaining products[11] and services through formal agreement

application engineering — an engineering process that develops software products from partial solutions or knowledge embodied in software assets

domain — an area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area

domain engineering — an engineering process that develops software assets for one or more domains

core asset — a software artifact that is used in the production of more than one product in a software product line. A core asset may be an architecture, a software component, a process model, a plan, a document, or any other useful result of building a system.

economies of scale — the condition where fewer inputs (such as effort and time) are needed to produce greater quantities of a single output

economies of scope — the condition where fewer inputs (such as effort and time) are needed to product a greater variety of options. Greater business value is achieved by jointly producing different outputs. Producing each output independently fails to leverage commonalities that affect costs. Economies of scope occur when it is less costly to combine two or more products in one production system than to produce them separately.

[11] Instead of products, the FAR use the term *supplies* to refer to tangible products and commodities.

formal agreement	a general term for the means by which an acquirer and supplier document the commitment between them. We assume that such an agreement must always be documented. This term encompasses the terms contract, memorandum of agreement, and other written agreements.
practice area	a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a software product line
product family	a group of systems built from a common set of assets
product line	a group of products sharing a common, managed set of features that satisfy the specific needs of a selected market or mission area
product line approach	a system of software production that uses software assets to modify, assemble, instantiate, or generate a line of software products
product line architecture	description of the structural properties for building a group of related systems (i.e., product line), typically the components and their interrelationships. The inherent guidelines about the use of components must capture the means for handling required variability among the systems. (This is sometimes called a reference architecture.)
product line system	a member of a software product line
production system	a system of people, functions, and assets organized to produce, distribute, and improve a family of products. Two functions included in the system are domain engineering and application engineering.
software architecture	structure or structures of the system that consist of software components, the externally visible properties of those components, and the relationships among them

software product line	a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way
system architecture	a software architecture plus its execution and development environments

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2001	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Fourth DoD Product Line Practice Workshop Report	F19628-00-C-0003

6. AUTHOR(S)

John Bergey, Sholom Cohen, Matthew Fisher, Grady Campbell, Lawrence Jones, Robert Krut, Linda Northrop, William O'Brien, Dennis Smith, Albert Soule

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2001-TR-017

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2001-017

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

The Software Engineering Institute (SEI) held the Fourth Department of Defense (DoD) Software Product Line Practice Workshop in March 2001. The workshop was a hands-on meeting to identify industry-wide best practices in software product lines; to share DoD product line practices, experiences, and issues; to discuss ways in which the current gap between commercial best practice and DoD practice can be bridged; and to obtain feedback on the first draft of the DoD Acquisition Companion to the Framework for Software Product Line Practice written by the SEI. This report synthesizes the workshop presentations and discussions.

14. SUBJECT TERMS	15. NUMBER OF PAGES
commercial product line practice, DoD product line practice, Software Product Line Practice Framework, product line workshop, software asset, software architecture, software product lines, acquisition companion, Acquisition Companion to the Framework for Software Product Line Practice	80

16. PRICE CODE

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