# Second DoD Product Line Practice Workshop Report

John Bergey
Grady Campbell
Paul Clements
Sholom Cohen
Lawrence Jones
Robert Krut
Linda Northrop
Dennis Smith

*October 1999*

TECHNICAL REPORT
CMU/SEI-99-TR-015
ESC-TR-99-015

blank page (to be thrown out immediately before production)

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

# Second DoD Product Line Practice Workshop Report

CMU/SEI-99-TR-015
ESC-TR-99-015

John Bergey
Grady Campbell
Paul Clements
Sholom Cohen
Lawrence Jones
Robert Krut
Linda Northrop
Dennis Smith

*October 1999*

**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

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.

# Table of Contents

# List of Figures

# Abstract

The Software Engineering Institute (SEI) held the Second Department of Defense (DoD) Product Line Practice Workshop in March 1999. The workshop was a hands-on meeting to identify industry-wide best practices in software product lines; to share DoD product line practices, experience, and issues; and to discuss ways in which the current gap between commercial best practice and DoD practice can be bridged.  This report synthesizes the workshop presentations and discussions.

# 1 Introduction

## 1.1 Why Product Line Practice?

Historically, software engineers have designed software systems for functionality and performance. A single-system mentality prevailed. Little attention was paid to the consequences of a design in the production of multiple software-intensive products or their long-term sustainment. Large software development, acquisition, and reengineering efforts undertaken with this single-system mentality perpetuate a pattern of large investment, long product cycles, system integration problems, and a lack of predictable quality. Each product involves vast investments in requirements analysis, architecture and design, documentation, prototyping, process and method definition, tools, training, implementation, and testing, with little carried forward to future products.

An increasing number of organizations are realizing that they can no longer afford to develop or to acquire multiple software products one product at a time. They have explicit needs to achieve large-scale productivity gains, improve time to market, maintain market presence, compensate for an inability to hire, and leverage existing resources.  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.

A *product line* is defined to be a group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission. It is most economical to build a software product line from a common set of assets.[1] In fact, the products in a software product line can best be leveraged when they share a common architecture that is used to structure components from which the products are built.  This software architecture[2] capitalizes on commonalities in the implementation of the line of products, supports the needed variation among the products, and provides the structural robustness, which makes the derivation of individual software products from software assets economically viable. The architecture and the components are central to the set of core assets used to construct and evolve the products in the product line.

[1] A software asset is a description of a partial solution (such as a component or design document) or knowledge (such as a requirements database or test procedures) that engineers use to build or modify software products [Withey 96].

[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].

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. 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 a technical initiative, 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 the multi-use of software assets so that these organizations can produce and maintain similar systems of predictable quality and at a lower cost. One of the strategies for reaching this goal involves direct interaction with and nurturing of the community interested in product line practice.

This transition strategy has been executed in part by a series of product line workshops organized by the SEI. Three of these workshops, in December 1996, November 1997, and December 1998, brought together international groups of leading practitioners from industry to codify industry-wide best practices in product lines. The results of these workshops are documented in SEI reports [Bass 97, Bass 98b, Bass 99]. 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 practice and commercial product line practices were discusses and documented [Bergey 98]. At this workshop 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 framework for product line practice. The SEI's Product Line Practice Framework is the first formal attempt to codify comprehensive information about successful product lines. The framework describes the foundational product line concepts and identifies the essential elements and practices that an organization should 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 categorized according to software engineering, technical management, and organizational management. These categories do not represent job titles, but rather disciplines. The framework is a living document that is evolving as experience with product line practice grows. Version 2 of the framework was made available on the SEI Web site in August 1999 [Clements 99].

## 1.2 About the Workshop

The SEI held the second in a series of two-day DoD Product Line Practice Workshops in March 1999 to achieve the following goals:

- identify industry-wide best practices in software product lines
- share DoD product line practices, experience, and issues
- discuss ways in which the current gap between commercial best practice and DoD practice can be bridged

The workshop participants were sent a draft copy of Version 2 of the SEI's Product Line Practice Framework 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 acquirer or DoD system contractors. Together we elucidated and discussed the issues that form the backbone of this report.

The workshop participants included

- Captain Jeffrey B. Benesh, U.S. Air Force
- John Bergey, Product Line Systems Program, Software Engineering Institute
- Loring Berhnardt, MITRE Corporation
- David Bristow, ITT Systems
- Grady Campbell, Product Line Systems Program, Software Engineering Institute
- Robert Chadbroune, TASC
- Kris Christianson, Joint National Test Facility
- Paul Clements, Product Line Systems Program, Software Engineering Institute
- Sholom Cohen, Product Line Systems Program, Software Engineering Institute
- Mark Dehlin, West Virginia High Technology Consortium (WVHTC) Foundation
- David DeKing, United Defense LP, Program Management Office for Crusader
- Joel Dorenfeld, Joint National Test Facility
- James Estep, West Virginia High Technology Consortium (WVHTC) Foundation
- Matthew Fisher, Product Line Systems Program, Software Engineering Institute
- Russell Graves, MITRE Corporation
- Major Scott Jacobsen, USASOC / Technology Applications Program Office (TAPO)
- Jeffrey Jensen, Joint National Test Facility
- Jon Johnson, TRW Strategic Systems
- Larry Jones, Product Line Systems Program, Software Engineering Institute

- Bradley Kettner, Joint National Test Facility
- Bob Krut, Product Line Systems Program, Software Engineering Institute
- Michael Marks, Rockwell Collins
- Brian Miller, Rockwell Collins
- Robert S. Miller, Altair Aerospace Corporation
- Lt. Col. George Newberry, U.S. Air Force
- Linda Northrop, Director, Product Line Systems Program, Software Engineering Institute
- Dean Oswald, Software Technology Inc.
- Major Frances S. Pacello, USASOC / Technology Applications Program Office (TAPO)
- Major LeRoy Pedone, U.S.  Air Force Integrated Space Command and Control
- Dr. Rami R. Razouk, The Aerospace Corporation
- Charles J. Ryan, Senior Member of Technical Staff, Software Engineering Institute
- Scott Sage, Altair Aerospace Corporation
- David Sharp, Boeing
- Jeffrey A. Shaw, Raytheon Systems Company
- Major Christopher Shotts, Joint National Test Facility
- Barry Smith, U.S. Air Force Electronic Systems Center (ESC)
- Dennis Smith, Product Line Systems Program, Software Engineering Institute
- Steven Woods, Product Line Systems Program, Software Engineering Institute
- Thomas E. Wylie, USASOC / Technology Applications Program Office (TAPO)
- Larry Yung, U.S. Army, Program Management Office for Crusader
- Paul Zamule, USASOC / Technology Applications Program Office (TAPO)

To properly set the context, the workshop began with two presentations by SEI technical leaders of the product line work.  They characterized the current state of product line practice by describing the industry's best product line practices, the current contents of the SEI Product Line Practice Framework, and preliminary guidelines on developing a product line concept of operations.  Representatives from five of the participating DoD organizations then made presentations explaining their organization's product line experience, describing how some of the practice areas in the framework are addressed.

Following the presentations, the participants divided into five working groups. Three of these groups were to explore DoD product line practices in software engineering, technical management, and organizational management.  They were asked to provide general comments on their category, especially as it is represented in the framework, and then select from among the practice areas identified in the framework for their category, and describe how each selected practice area "gets done" within the DoD community.  The explicit practice areas, defining and implementing an acquisition strategy and Product line concept of operations, were each explored by one of the two remaining groups. Some groups used the framework format to structure and document their discussion. Others were more free form. In addition, the software engineering group was challenged to consider whether or not Domain Analysis should continue to be a separate practice area in the framework or should be included as a specific practice in the practice are Requirements Elicitation, Analysis, and Tracking.

The workshop concluded with the working groups presenting their results to the entire group, followed by a verbal evaluation of the workshop.

## 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 or initiating product line practices in their own organizations. Acquisition managers and technical software managers should also benefit from the information in this report.

The report is organized into six main sections that parallel the workshop format:

1. Introduction
2. State of Product Line Practice: Digest of SEI Overview Presentations
3. DoD Product Line Experiences: Digest of DoD Presentations
4. Product Line Practices: Working Group Reports
5. Feedback From Workshop Participants
6. Summary

The section following this introduction, *State of Product Line Practice: Digest of SEI Overview Presentations*, summarizes the two SEI presentations that set the context for the workshop. The next section, *DoD Product Line Experiences: Digest of DoD Presentations*, summarizes the product line experience of five of the participating organizations. Section 4 is composed of the five working group reports on selected practices in software engineering, technical management, organizational management; and on defining and implementing an acquisition strategy, and product line concept of operations, respectively. Each of the working group reports reflects the interests, experiences, and style of the individual group. The emphasis and completeness of the information varies by group and by practice. The practices discussed are important in their very selection. Section 5 lists some specific recommendations from the workshop participants relative to the framework and to the workshop. The summary in Section 6 recaps the major themes and suggests future directions. Additionally, there is an Appendix providing a glossary of terms.

# 2 State of Product Line Practice: Digest of SEI Overview Presentations

## 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 product line, the state of commercial product line practice, the relevance of product lines to DoD, the SEI Product Line Practice Initiative, and the SEI Product Line Practice Framework

She then distilled the results of the SEI's Third Product Line Practice Workshop held in December 1998. She uncovered the issues and solutions shared by experts from five commercial organizations with real-world experience in developing and fielding software product lines.

Sholom Cohen then presented the SEI work on a guidance document for creating a concept of operations for product lines. This document has great potential for application within the DoD community. The workshop participants' feedback was very helpful.

## 2.2 Overview: SEI Product Line Practice Framework, Linda M. Northrop - SEI

### 2.2.1 What Is a Product Line?

A product line is a group of products sharing a common, managed set of features that satisfy specific needs of a selected market or mission. This definition has been around a long time in the manufacturing world. For example, a telecommunications company may offer a number of cellular phones that share a similar market strategy and an application domain, thus making up a product line. It is well understood in the world of manufacturing that when you have a product line you take economic advantage of the common features of the products in the product line when you are building those products. Common product designs and parts are used. Assembly lines and automated tool support are set up.

It is only within the last ten to fifteen years that organizations developing software have taken advantage of the commonality among similar systems and have taken a product line approach. The most economical way to approach software product lines is to build a common architecture that is shared by the products in the product line and that is used to structure the components from which the products are built.

The architecture and components are central to the set of core assets[3] used to construct and evolve the products in the product line. In other words, a software product line can best be leveraged by managing it as a product family (as it has traditionally been called in the field of computer science). A product family is a set of related systems built from a common set of assets. For example, if the product line of cellular phones is built from a common architecture and set of common components, it is managed as a product family. When we refer to a product line, we always mean a software product line built as a product family. This particular use of terminology is not nearly as important to us as the underlying concepts involved, namely, the use of a common asset base in the production of a set of related products.

Product line practice is therefore the systematic use of software assets to modify, assemble, instantiate, or generate the multiple products that constitute a product line. 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

### 2.2.2 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 market or field at the right time. Time 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 their on-board ship command and control systems in the mid 1980s [Brownsword 96]. Their efforts resulted in a product line they call Ship System 2000 that now spans 12 classes of ships, from surface vessels to submarines, and has fielded more than

[3] Some organizations refer to the core asset base that is reused on systems in a product line as a *platform*.

50 ship systems from the same architecture and set of components. Among many other benefits that CelsiusTech has enjoyed with this product line is a reversal in the hardware-to-software cost ratio, 35:65 to 60:20, that now favors the software.

A number of other companies have shown similar success using a product line approach. Hewlett Packard, who like CelsiusTech has been using a product line approach for the past ten years, has collected substantial metrics showing two-to seven-times cycle time improvements with product lines. On one project they were able to ship five times the number of products, that were four times as complex, 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 four times cycle time improvement with 80% reuse. Cummins engine Co. uses a product line approach for the engine control software for their diesel family and site 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), 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 to continue their health in the market, to maintain market presence, to sustain unprecedented growth (especially poignant given today’s employment market) 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 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 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 from DoD Directive 5000.1 and DoD Regulation 5000.2-R have 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 what the commercially proven practices are that 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 others within the DoD that are attracted to product line concepts that 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. This workshop is a part of the planned activities of the SEI's Product Line Practice Initiative, which is attempting to bridge this gap.

### 2.2.4 The SEI Product Line Practice Initiative

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

One of the ways in which the SEI executes these strategies is to directly collaborate with organizations on product line efforts. The National Reconnaissance Office, 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 transition.

Widespread transition has been effected through a series of workshops and presentations, targeted at well-defined audiences. The SEI has hosted three workshops for commercial leaders [Bass 97, Bass 98b, Bass 99], organized seven workshops for researchers and technologists, and, with this workshop, two for the DoD community [Bergey 98]. Presentations have been given at countless government, commercial, and technical forums.

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

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

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

### 2.2.5 Product Line Practice Framework

We are capturing those essential activities and practices in the SEI Product Line Practice Framework. The framework is a web-based, evolving document that is designed to address both development and acquisition contexts. The framework is primarily targeted at members of organizations who are in a position to make or influence decisions regarding the adoption of product line practices.[4]

As depicted in Figure 1, core asset development and acquisition are distinguished from product development and acquisition using these assets with the understanding that management orchestrates, tracks, and coordinates both sets of activities. The arrows signify the high degree of iteration involved.

[4] At the time of this workshop, Version 1.0 of the framework was available on the SEI Web site. Workshop participants were sent a draft of Version 2.0. At the time this report is being written, Version 2.0 is available at http://www.sei.cmu.edu/plp/framework.html.

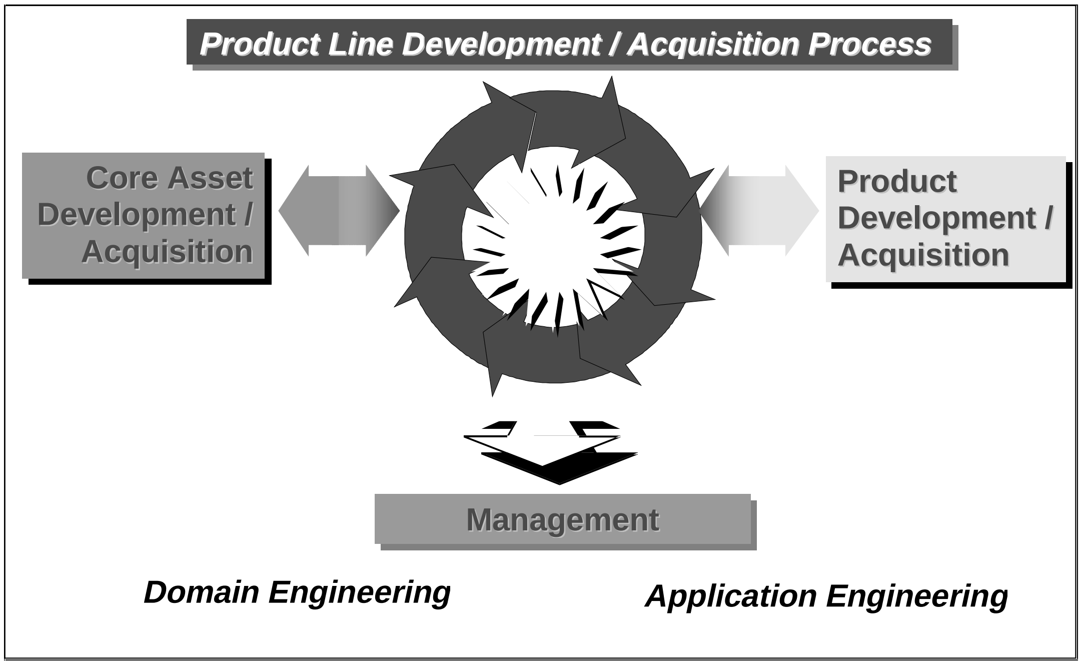
Product Line Development / Acquisition Process
Core Asset Development / Acquisition
Product Development / Acquisition
Management
Domain Engineering
Application Engineering

*Figure 1: Product Line Practice Essential Activities*

On the left side of the figure, the critical core assets involved are the architecture and components. Inputs to the development and acquisition of core assets are: product constraints found by analyzing the similarities and differences of current and projected products; production constraints such as might be found in a technical architecture; a production strategy for the assets; an inventory of pre-existing assets; 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.

On the right side of the figure, individual products are developed or acquired 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, the 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 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. 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/acquisition
- how this practice area is applied to product development/acquisition
- specific practices in this practice area
- risks in this practice area
- references

#### 2.2.5.1 Software Engineering

The software engineering practice areas include

- Domain Analysis
- Architecture Exploration and Definition
- Architecture Evaluation
- Mining Existing Assets
- Component Development
- Testing
- Requirements Elicitation, Analysis,and Tracking
- COTS Utilization
- Software System Integration

#### 2.2.5.2 Technical Management

The technical management practice areas include

- Data Collection, Metrics, and Tracking
- Product Line Scoping
- Configuration Management
- Process Modeling and Implementation
- Planning and Tracking
- Make, Buy, Mine, Outsource Analysis
- Technical Risk Management

#### 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
- Funding
- Market Analysis
- Developing and Implementing an Acquisition Strategy
- Operations
- Training
- Customer and Supplier Interface Management
- Technology Forecasting
- Launching and Institutionalizing a Product Line[5]
- Organizational Risk Management

### 2.2.6 Framework Future Direction

The SEI Product Line Practice 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 are incorporating that feedback into Version 2.0. We are encouraged to learn that fourteen organizations have reported to us their use of the framework in their software product line efforts. The community response has been most favorable.

Future versions of the framework will build upon the current foundation and incorporate feedback and our experience, will complete the other practice are descriptions, will describe a small number of product line scenarios, and will provide a list of frequently asked questions and answers.

To further support product line practice transition, the SEI is producing generic product line artifacts, case studies, technical reports, and instructional products, and is beginning to pilot product line technical probes in individual organizations to determine product line readiness.

[5] At the time of this workshop we considered this to be two separate practice areas: Launching a Product Line and Product Line Institutionalization. Owing in part to workshop input and also to subsequent investigation and discussion, the two were combined in the published Version 2.0.

### 2.2.7 Third Product Line Practice Workshop Highlights

#### 2.2.7.1 Introduction

In December 1998, the SEI conducted the third in its series of workshops on product lines. The goals of this workshop were to

- share information and issues about product lines
- stimulate the growth of a network of interest in software product lines
- populate the SEI framework with proven practices
- point out gaps where experience is not properly reflected in the framework

Representatives from the following organizations were present:

- Cummins Engine Company
- Thomson-CSF
- Raytheon Missile Systems
- Boeing
- Telesoft S.p.A.

This workshop had a different character than our earlier workshops for commercial leaders. In the earlier workshops, participants had product lines and told us how they did it. In this workshop, the participants were launching product line efforts; they described why and how.

The workshop featured presentations by representatives from each of the participating organizations, followed by working groups that focused on product line practices in the areas of software engineering, technical management, and organizational management. The results of this workshop were incorporated into the SEI's Product Line Practice Framework.

This summary first synthesizes the common themes of the presentations as well as some heuristics that were shared. Next, the discussions of each of the working groups are summarized. A much more complete discussion of this workshop can be found in the corresponding workshop report [Bass 99].

#### 2.2.7.2 Common Themes

The importance of architecture was apparent in each presentation. Every speaker presented the architecture for his or her product line, underlining its central role. The creation of the right product line architecture was considered by all to be a crucial and difficult step to achieve. Several organizations are using the Unified Modeling Language (UML) as an architectural description vehicle. There were many references to deficiencies in UML that needed to be accommodated.

Discussion of organizational structure was also a common theme. Each of them adopted the separate-group model for producing core assets. In this structure, a dedicated group builds core assets, and products are built using these core assets by separate product groups. One variation on this model was a "lightweight organizational structure" that was put in place while the organization is transitioning to a product line.

Product line scoping was an important practice area for all participants. Three of the organizations represented at this workshop had to make an explicit decision about whether their business would be best served by a small group of tightly focused product lines, or a single, somewhat more generic product line. Significantly, all three organizations decided in favor of a single product line, suggesting that the benefits of product lines are faster when the scope is set somewhat broadly.

Other themes common to the presentations included the following:

- The use of use cases is the primary mechanism for capturing requirements and domain models. Two participants included an explicit domain analysis.
- The judicious use of small prototypes and demonstration projects can help work out the bugs in setting up a product line production capability in an organization.

### 2.2.7.3 Pearls

Each presentation contained one or more "pearls" of knowledge.[6] Among them are the following:

- Develop a "lightweight organizational structure" and "lightweight processes" to serve during product line adoption.
- Explicitly catalog the commonalities and variabilities among the products in the product line.
- Comprehensively catalog the requirements of the products in the product line.
- Explicitly choose architectural styles and patterns.
- Don't establish the asset base as common across all applications.
- Recognize automatic regression testing as a critical capability when fielding a core asset base.
- Recognize the importance of requirements traceability in the core asset base.
- If needed, actually stop the efforts on stovepiped systems to initiate product line.

### 2.2.7.4 Working Group: Software Engineering

The software engineering working group focused on two practice areas from the framework: requirements elicitation, analysis, and management; and component development.

[6] These heuristics are more completely explained in the *Third Product Line Workshop Report* [Bass 99].

In the area of requirements elicitation, analysis and management, four questions were explored:

1. How are requirements used?
2. What kind of requirements are important?
3. What kind of development contexts are there and do they have an impact on requirements?
4. What is an example of a product line requirement?

Requirements in product lines are used to: distinguish between those requirements that will be satisfied by software assets and those that will be satisfied by specific products; provide engineers with instructions as to their task; and provide product customers with a traceability baseline so that they can verify that their requirements have been met. It was agreed that requirements include both behavioral and quality-related requirements, at both run-time and development time. The group focused on behavioral requirements, excluding performance, modifiability, and other qualities. Two development contexts were discussed: green-field (from scratch) and plowed-field (systems exist that fit within at least the initial scope of the product line). It was concluded that a different requirements process would be required for each of these two contexts. It was agreed that product line requirements necessitate some type of abstraction that generalizes requirements that are common to all products and describes how the particulars of each general requirement can vary from product to product. This abstract representation manages the complexity of product line requirements. However, the development of the abstract representation has a cost that is best managed in a green-field development. The abstract representation can be used both for development specifications and for traceability.

The group considered two aspects of component development practices: (1) the process for developing components, and (2) techniques used to manage variation. Insofar as possible, components should be developed to be intelligent about their environment. That is, they should, either during initialization or as a portion of the build process, have the flexibility to adapt to different environmental conditions such as using different network protocols or different methods for being invoked or gaining input data. Once the rules for writing components have been regularized, a guide describing the rules and the process for developing components would be written. Three techniques were identified to manage variation: (1) parameterization, (2) inheritance and delegation, and (3) call backs.

#### 2.2.7.5 Working Group: Technical Management

The technical management working group focused on the process modeling and implementation and technical risk management practice areas. Because product lines represent a new way of doing business, those involved in developing or acquiring them require process guidance in the form of templates and examples. There is a need to define the following:

- the main practices and activities of product line development
- the flow among these practices and activities
- descriptions of how activities actually transform inputs into specific deliverables and products

Product line process modeling must be managed flexibly, cutting across organizational units. The process definition should initially be more lightweight and have greater degrees of freedom. As an organization is successful with a product line approach, the process model can—become more formal. This group identified suggested practices for process modeling as well as inherent risks in this area.

As with any complex software development or acquisition activity, risk management is crucial for implementing a product line approach. There are well-established approaches and guidelines for performing risk management that should be a good starting point. The group explored how these approaches can be used for product lines and identified specific practices for technical risk management in a product line context.

#### 2.2.7.6 Working Group: Organizational Management Practices

This working group attempted to clarify the nature of organizational management and the composition of its product line practice areas. The group began by characterizing the timelines that organizations follow in instituting a product line approach. They identified two organizational management timelines to consider: (1) the enterprise timeline that charts how an organization institutes a product line approach and (2) the product line timeline that charts how a business unit adopts a product line approach for a particular product line business. The group then discussed the primary milestones of each timeline.

They then determined that the practice areas in the organizational management category should help an organization address

- the business (products and customer, now and in the future)
- the approach (product line practices that will benefit the business and how to transition to them)
- managing the approach (the management practices needed to ensure the success of product line efforts)

The group then discussed concerns that need to be addressed when launching a product line and characterized each concern from the perspective of a single product line effort and from the perspective of the broader business enterprise of multiple product lines.

Infrastructure planning for product lines was also an organizational management practice area that was discussed. The purpose of infrastructure planning is to determine what facilities are needed to support the product line approach and to plan their acquisition or development.

Finally, the group identified four goals for managers of product lines:

1. Communicate product line strategy and vision.
2. Set product line objectives.
3. Create rewards and incentives associated with the product line objectives.
4. Show commitment.

## 2.3 Product Line Concept of Operations, Sholom Cohen - SEI

Sholom Cohen presented his work on a guidance document for creating a concept of operations (CONOPS) for product lines. The purpose of a product line CONOPS is to describe, at a high level, key organizational considerations that are necessary to implement a product line approach. Thus, an organization's CONOPS serves as an important communications vehicle to identify the significant changes the organization will experience. These changes include such things as organizational structure, roles and responsibilities, and processes.

The CONOPS guidelines document was created to help an organization formulate its own specific CONOPS. Because "one size" does not fit all organizations, the guidelines are more than a simple template. They also includes a series of key questions that will help the CONOPS creator apply the guidelines appropriately.  Additionally, examples based on actual experience are included.

As described in the guidelines, the CONOPS will include the following sections:

- Overview: basic concepts and definitions
- Approach: how products will be fielded
- Background: the history and motivations leading to the product line decision
- Organization: the new organizational elements and interactions
- Technical: processes for creating and sustaining core assets and products
- Recommendations: a set of initial issues and actions to be taken.

The guidelines also identify key questions that will help in formulating and tailoring the CONOPS contents. These questions are the following:

- Who is the audience for the CONOPS?
- What constitutes the product line?
- Who is/will be the product line champion?
- What are the key organizational elements?

- How will the product line be introduced?
- Is there an acquisition/contractor relationship for core assets? For the products created from the core assets?

As noted, there will be variation among actual CONOPS. Some of the parameters that will influence the CONOPS contents include

- product attributes: size, maturity, mission or market
- asset procurement: in house, commissioned, or off-the-shelf?
- assets categories: models, architecture, components, etc.
- asset development: green field, legacy, evolutionary
- use of assets: complete/partial system, frequency of use in product development, longevity of product line
- control of assets: internally, internal and external, available for external use
- acquisition: number of contractors, mixed contractor/acquirer, architecture provided to vendors

Finally, the examples provided are drawn from the Air Force, Navy, National Reconnaissance Office, and a composite of several commercial organizations.

As of the time of this workshop, the CONOPS guidelines were in draft form. A Technical Report, *Guidelines for Developing a Product Line Concept of Operations*, has since been published and is available on the SEI Web site [Cohen 99].

# 3 DoD Product Line Experiences: Digest of DoD Presentations

## 3.1 Introduction

The following four presentations of DoD product line experiences were selected:

- Avionics Software Product Line Development - David Sharp, The Boeing Company
- Crusader – A System for the 21st Century - Larry Yung, U.S. Army Crusader and David DeKing, UDLP
- Missile Defense Space Tool (MDST) - Brad Kettner, Joint National Test Facility
- Raytheon Satellite Ground Station Command and Control Product Lines - Jeffrey A. Shaw, Raytheon Systems Company

## 3.2 Avionics Software Product Line Development - David Sharp, Boeing

Boeing has initiated a product line approach for the development of reusable operational flight program (OFP) software [Sharp 98]. Targeted reusers are Boeing's developers of avionics software for fighter aircraft, including the F-15, the FA-18, and the AV-8.

This initiative was chartered to support reuse of OFP software across multiple fighter aircraft platforms and to create a supporting architecture based on commercial hardware, software, standards, and practices. Substantial emphasis has been given to developing a goal-guided, pattern-based architecture, and pattern-based components supporting object-oriented development. Goals were to localize changes and system differences to particular components, to ensure portability and distributability, and to maximize reuse across products. However, an overriding goal was to support real-time performance requirements. The result is a layered architecture that isolates changes in mission computing hardware and runtime system, in avionics subsystems (radar, cockpit controls and displays, etc.), and in system requirements, to specific layers and components.

Organizationally, architecture and components development is the responsibility of a Common Software Group. Oversight and guidance is provided by a Core Architecture Team consisting of the lead architect and layer leaders from the Core Software Group and the lead architects from the various Boeing avionics software product engineering groups.

The architecture has been designed based on the principles of two widely known patterns: Layers and Model-View-Controller [Gamma 95]. The Layers pattern guides partitioning of the architecture into distinct sets of components, each set being characterized by a focus on particular concerns. For example, one layer includes components that provide access to hardware while another layer includes components that perform specific system functions such as radar processing or weapons control. The Model-View-Controller pattern is based on the insight that user interfaces often present different views of the same information. Furthermore, the general information model of a system is inherently more stable than the form and content of particular views presented to users.

Components are of medium granularity, integrating multiple objects behind a façade to provide unified coherent services to other parts of the software. For flexibility, components are designed as instances of patterns that make them configurable (based on the Abstract Factory pattern), distributable (across 4-10 processors), or capable of real-time levels of service. The façade aspect of a component unifies concurrency control, distribution, persistence, configuration, and execution dependencies for a set of contained interrelated objects.

These patterns are combined to create a top-level architecture consisting of four layers: configurator, operator, model, and infrastructure services. The infrastructure services layer abstracts and encapsulates a common object request broker architecture (CORBA)- based execution environment consisting 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 a capability to configure and integrate 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 software in higher layers to bypass intermediate layers if needed to directly access lower layer components.

## 3.3 Crusader – A System for the 21st Century, Larry Yung – U.S. Army Crusader; Dave DeKing – UDLP

Larry Yung and Dave DeKing provided a joint presentation of the Army Crusader Program. This program is responsible for development of a new line of highly mobile, fully automated howitzer system. This system consists of two platforms: a self-propelled howitzer (SPH) and a re-supply vehicle (RSV). The two platforms are designed to mate together in the field to form a single fighting unit. Both vehicles are being developed from scratch, under the same management and contractor control, with reuse requirements within and across the two vehicles.

Combined, the SPH and RSV address five key areas:

- firepower – cooled cannons for continuous fires, 10-12 rounds per minute out to a range of 40-50 km, enhanced accuracy with a projectile tracking sensor
- mobility – 1500 HP to meet and exceed M1/M2, first driven-by-wire ground combat vehicle, better ride quality than M1/M2
- automation – fully automated resupply, ammunition handling, aiming, loading, and firing
- information dominance – crew cockpit enables information dominated warfare through mission planning, situation awareness, and decision aids
- survivability – separate crew and weapons stations, composite armor, and ballistic and non-ballistic protection

The RSV and SPH are highly automated to meet performance requirements. Software is responsible for setting the howitzer angles, loading, and firing the charge. A multifunction yoke will control the mechanical functions of the vehicle. Both vehicles will employ a common man/machine interface. A major subsystem of the man/machine interface is a vehicle control panel used to drive the vehicle home in case of software failure. The Crusader "cockpit" software presents the maps, navigation data, etc. in graphical form to the operator. Terrain analysis is provided for determining the best routes.

GPS will supply position and navigation information to the RSV and SPH. Electronics will come from other programs and the reuse of software from advanced field artillery tactical data system (AFATDS). The mass memory unit is the same unit on the Bradley program.

The program is currently at the preliminary design review (PDR) phase. Prototype systems are being developed that will evolve into the actual systems. The RSV chassis is scheduled for delivery in June 1999 followed by prototype software and the SPH in 2000. Delivery of the first fully equipped unit is expected by 2005.

The Crusader program has decided to abandon the more traditional software partitioning approaches because of expected hardware and software changes. The Crusader program will employ a Real-Time Common Operating Systems (RTCOE) layer to separate hardware and software. This middleware should reduce software dependencies on hardware, defining computer software configuration items (CSCIs) on physical boundaries, and hard coding communication into the application code. Isolating application software from specific hardware implementations will allow for hardware upgrades/additions, operating system upgrades, and isolate compiler dependencies. The Crusader program will employ up to 30 CSCIs. Testing and integration of the CSCIs is expected to cost more yet be more reliable. All software development teams will use the same structure when developing software. Hardware developers will use the same real-time system. So that all applications can accommodate new hardware, the program will use 2 processors connected by a system bus.

The Crusader program is taking an object-oriented approach and has worked with users to develop use cases and scenarios. Use cases describe key system functions that are triggered by a user or external stimulus. Scenarios are single instances of a use case. They represent a path through the use case that elicit some system behavior. The identified use cases were common to both vehicles. These include: initialize and configure, conduct deployment, conduct re-supply, conduct self defense, conduct training, conduct maintenance, aid decisions, communicate, and operate equipment. The only use case not common to both vehicles was conduct fire mission.

The Crusader program did not start out as a product line development effort. However, as the effort gathered momentum, the Crusader program realized that the development of the SPH and RSV could be the foundation of a new product line. Therefore, the program began an investigation into a potential product line that would be based on the SPH/RSV development. A 5-layer common architecture has been proposed.[7] Layer 1 would provide common utilities for low-level interfaces and reusable components. Layer 2 would provide common services for a distributed computing environment. This includes interface, peripheral, and distributed services as well as common types and abstractions. For the Crusader program, each processor would include the common utilities and services. Layer 3 would represent the "virtual artillery vehicle." Initially, this layer would provide specific crusader mobility, survivability, re-supply, armament, and battlefield software. Layer 4 would be for artillery vehicle command and control and embedded support. Layer 5 is for user and external system visibility. This layer represents all external user interfaces and off-board test and training.

The current vision is that many other programs will be interested in using the common utilities and common services layer. Possible mission (layer 3) software may be reused or leveraged off of in the future. Layer 3 would also be tailorable[8] and scalable. Although not part of this effort, there may exist potential reuse for the M1 or Abrams/Bradley platforms.[9] Commonality across M1 or Abrams/Bradley currently has undesirable legacy applications but as these systems evolve, they may use this configuration and components. This would provide a common infrastructure between them so they can share applications.

## 3.4 Missile Defense Space Tool (MDST) - Brad Kettner, Joint National Test Facility

The Joint National Test Facility (JNTF) provides modeling and simulation, analysis, testing, war gaming, and exercise support to the DoD, joint, individual service, and international acquisition and war fighting communities. JNTF has developed a pilot implementation of the product line approach through the implementation of Missile Defense Space Tool (MDST). The MDST system represents a merger between a current system, Portable Space Model (PSM), and the new capabilities represented by the Space Based Infra Red System Wargaming Model (SBIRS).

[7] Lowest level implies closest to the hardware.
[8] Tailorability would be fixed at run-time.
[9] It should be noted that neither the Bradley nor M1 are object-oriented developments.

The current PSM simulation provides a simulation of the current Display Support Program (DSP) satellites in geostationary orbit. First launched in 1970 and still operational today, DSP’s primary mission is to detect strategic rockets in their boost phase and to report them to the National Command Authority. The emerging SBIRS system is being developed to meet the evolving theater threats in the post-Cold War era. It will integrate space assets in multiple orbit configurations with a consolidated ground segment to provide more effective integration of data and better information.

SBIRS/Wargaming Model (SBIRS/WM) represents an approved simulation of the emerging SBIRS system. However, in 1998, it did not yet have certification to model the older DSP satellites. On the other hand PSM has the required DSP certification. But PSM did not have the capability to model the SBIRS system.

JNTF made the decision to merge the capabilities into a new system, called MDST. Two factors led to this decision:

1. A working group at JNTF, with the SEI as a catalyst, had been looking for a pilot to test the feasibility of product line concepts at JNTF on a small scale.
2. JNTF wanted to participate in the Expeditionary Forces Experiment 1998 (EFX 98), which required SBIRS/WM to add a DSP capability to properly represent the emerging SBIRS system.

MDST was conceived as a cost-effective way to develop and inject a valid SBIRS model in wargames and exercises. The concept behind MDST was to develop a flexible framework for the still evolving and changing SBIRS. As the system evolves, new models can be quickly incorporated with minimal impact to the user community, and at a lower cost than developing a new simulation.

By reusing common components, the initial technical release of the new product to meet EFX’98 requirements took approximately 3 weeks (the coordination took 3 months) at a cost of $110,000 using 3-5 part time people. Without this reuse, the JNTF would not have been able to participate in EFX exercises with a SBIRS model for at least 12 months. In addition, if the certified message capability had been developed from scratch and added to SBIRS/WM, the total cost would have been at least $150,000 greater. If a new DSP model had to be written and added to SBIRS/WM, it could have taken up to several years to develop at a cost of $400,000. These data refer only to the costs of meeting the EFX’98 requirement. The longer-term savings of reusing common components for future simulations are in the range of $5 million to $8 million.

The pilot project demonstrated three types of results for JNTF:

1. that the approach of managing software assets as a set of common components is viable, and its implications should be studied in greater detail

2. that horizontal leverage within JNTF organizations can be effective in pursuing common goals
3. that cooperative arrangements between JNTF and external organizations, such as those that enabled MDST to become successful, can potentially save significant time and resources

Once the initial capability was demonstrated, plans were made to develop MDST in multiple phases to accommodate short-term requirements such as EFX 98, as well as longer-term product enhancements such as incorporating upgraded or new models. The MDST is currently being developed to evaluate a product line approach for software development at the JNTF. The goal is to develop a tool set that can be easily modified to meet the needs of a specific product.

The results of the pilot also suggested a potential change in the JNTF business model. Traditionally, JNTF builds models for specific situations. The models are custom tailored for a particular need, and not reused. The MDST approach builds a framework into which specific models can be integrated. By building, maintaining and integrating frameworks, JNTF can potentially react to new opportunities more quickly and more cost effectively. Using this new approach, a product is defined as the application of a set of core assets (such as MDST or personnel) brought together to meet a specific requirement or market application. An example of a possible product at the JNTF is a wargame or exercise, not necessarily the simulation used to support these exercises.

### 3.4.1 Conclusions and Next Steps

The technical accomplishment of MDST was modest. However, to accomplish the goal, it was necessary for a number of different organizations including JNTF, Space Missile Center (SMC), and Space Warfare Center (SWC) to work together. This combination of organizational resources demonstrated that the reuse of common assets is possible across organizational boundaries, and that significant gains are possible.

On March 8, 1999, MDST was accredited by HQ United States Space Command (USSPACECOM)/J3J7 as accurate and appropriate for use in joint exercises and training events to simulate the operations of the Tactical Event System (TES) in providing warning of theater ballistic missile launches to a Commander in Chief/Joint Tactical Forces (CINC/JTF) level training audience. MDST is the USSPACECOM model of choice to simulate theater missile warning operations in constructive campaign simulations. This marked the achievement of one of MDST’s major goals, and marked the acceptance of this tool by a member of the war fighter community.

## 3.5 Raytheon Satellite Ground Station Command and Control Product Lines, Jeffrey A. Shaw – Raytheon Systems Company

Jeffrey Shaw described the Control Channel Toolkit (CCT) and Raytheon Eclipse—two satellite-ground-station, command and control product lines. Shaw was the program manager from the contractor side for CCT, which was developed under the auspices of the National Reconnaissance Office (NRO). Shaw is now the lead on a Raytheon product line called Eclipse, where CCT is being leveraged to provide Raytheon-specific ground station command and control software for commercial satellite operation.

The driving requirements for the CCT program were to reduce life cycle costs and development risks, promote interoperability, and provide flexibility. Flexibility translates into accommodating multiple implementation contractors and enabling the integration of COTS and existing legacy assets. CCT includes a reference architecture, extendable software components, and documentation to support the development of ground-based spacecraft command and control systems.

### 3.5.1 Domain Engineering

In the early stages of the CCT, an array of government and commercial systems was examined for domain requirements: Distributed Command and Control System (DCCS), MALTA, and Standard Spacecraft Control System (SSCS). The objective was to develop a tool kit, once, and have reuser programs share development and maintenance costs. The effort began by performing a domain analysis of these systems. Life cycle costs for these systems were also analyzed. Based on a domain engineering requirements analysis, cost estimates were derived for implementing the candidate set of systems using a product line approach. The commonality of requirements across these systems ranged from a low of 49% to a high of 89% and the cost benefit analysis showed significant life cycle cost savings.

CCT emphasized reengineering of existing DCCS modules, developing an open-systems architecture, using standardized application program interfaces (APIs) and common plug-and-play components with well-defined interfaces, and integrating customized components to meet mission-unique requirements.  In general, CCT represents greater than 50% of the solution while the difference is mission-specific reuse or development.

A typical CCT “product line reuser” begins by examining the CCT reference architecture and then tailoring/targeting it to define a systems architecture that is compatible with the desired system features and system quality attributes. Reusers then use the system architecture to implement their system using CCT components and custom code they develop.

### 3.5.2 Development Process

The CCT development was structured into six overlapping increments with an Integrated Product Team (IPT) responsible for each. In the CCT product line environment, the work is broken down into five areas: domain and architecture engineering, component engineering, application engineering, test engineering, and sustainment engineering. Since the architecture drives everything, the first increment addressed the domain and architecture engineering task; this involved scoping the product line. Each of the other increments constituted a set of core capabilities for a satellite command and control system.

The systematic reuse approach adopted by CCT was process-driven and took advantage of what was available—legacy assets, proven algorithms, and COTS products. Value-added processes included conducting frequent working Technical Interchange Meetings (TIMs), promoting processes that provided an engineering product focus, and implementing metrics to track and evaluate progress.  IPTs were used to promote community and stakeholder involvement and avoid long, drawn-out Preliminary Design Reviews (PDRs) and Critical Design Reviews (CDRs). Added benefits of using this incremental, evolutionary approach were increased system visibility, improved documentation, and increased understanding. Other value-added CCT innovations included providing documentation online as well as community interaction via a CCT Web site.

Test engineering adopted a component-level test approach consisting of four levels: informal unit test (level 0), formal component test (level 1), informal application test based on use case analysis (level 2), and formal system test (level 3). By structuring the requirements to map to test cases, the requirements, in effect, defined the test strategy and drove the detailed test procedures. CCT Performance Validation has many facets. The overriding consideration was to demonstrate the nimbleness of the product line reference architecture and validate that reusers do not have to standardize early on COTS products.

### 3.5.3 Issues

Technical, cultural, and organizational issues continue to surface in the process of moving forward with the CCT and Eclipse product lines. The technical issues concern scaling the CCT toolkit up to the system level, reconciling competing technologies, and developing training. While the CCT tool kit is a significant technical achievement, the problems that prove most challenging are organizational and cultural problems rather than technical problems.

### 3.5.4 Commercial Spin-Off

Eclipse represents the first commercial reuser of CCT. CCT and Eclipse are conceptually the same. Raytheon is bringing both its government and commercial systems together through the CCT reference architecture. Eclipse is building from the CCT toolkit and is implementing a system architecture using the CCT processes and tools. With Eclipse, the software development paradigm for ground-based satellite command and control systems will involve seamlessly integrating Raytheon common software components with COTS software and satellite-specific software. This includes support for major industry standards for open architectures (e.g., Ethernet LAN, TCP/IP Protocol, VME Bus) and a database-driven architecture to accommodate new user requirements.

Raytheon has over 40 ground-based command and control systems installed worldwide. This represents 30 years of experience in spacecraft command and control systems. Eclipse and CCT permit Raytheon to use a product line approach to leverage 30 years of experience and corporate “know-how” in developing command and control systems.

# 4 Product Line Practices: Working Group Reports

The following sections contain reports from the working groups. These working groups covered software engineering practices, technical management practices, organizational management practices, acquisition management practices, and product line concept of operations.

## 4.1 Software Engineering Practices

The software engineering breakout group decided to explore the Architecture Definition and Exploration practice area as well as issues concerning the evolution of a product line over time. Before tackling these topics, however, the group took a few minutes to discuss the question put to them in the plenary session: Should domain analysis be a separately identified practice area in the framework, or is its work subsumed in the Requirements Elicitation, Analysis, and Management and the Product Line Scoping practice areas?

### 4.1.1 Domain Analysis

The group identified two roles associated with domain analysis in a product line context:

- analyzing specific legacy products to see if they are usable in the envisioned product line: Domain analysis provides a prescribed approach for cataloguing assets in a selected area and determining their potential appropriateness for consideration in the product line by comparing their functional capabilities to those envisioned for the product line.
- analyzing the general product concept for feasibility: By analyzing legacy products in the domain(s) related to the product line, domain analysis aids in understanding the structure and state of the domain(s) from which the product line will draw its resources. Legacy products represent the history of a domain or set of domains; the product line to be built represents one part of its future, and the former informs the latter. Domain analysis represents a tool for methodically cataloguing what has come before.

Interestingly, domain analysis serves both of these roles in a single-system context as well. The difference in a product line setting is the “multiplicity” mentality that is required—that is, considering the effects of the domain analysis results on the envisioned family of systems, as opposed to just one. A third role was this one:

- maintaining domain requirements in the face of changing market needs and technologies: As the product line matures and evolves, the domain models (the output of the domain analysis) must evolve as well so that they remain viable assets for the product line' planning function. Also, as new product requirements (which are noted to be common or generally useful across more than one product) are promoted to product-line-wide requirements reflected in the product line scope, the domain models must accommodate these changes.

The sense of the group was that domain analysis is a key activity in a product line and that the framework must highlight it prominently to make sure that readers understand its importance. Keeping it a separate practice area would be one way of maintaining its prominence. Featuring it conspicuously in some of the frequently asked questions would be another. However, the group acknowledged that domain analysis is more often recommended than performed, at least in any comprehensive explicit manner, and that all of the roles allocated to it are in fact consistent with those under the Product Line Scoping practice area and the Requirements Elicitation, Analysis, and Management practice area. Under those conditions, making it a specific practice under those practice areas seemed to be a reasonable approach.

### 4.1.2 Architecture Definition and Exploration

Architecture remains the linchpin software engineering area of product line practice, occupying the key role of central core asset on which all derivative products are based. Therefore, it is not surprising that the software engineering working group wanted to devote some of its discussion time to the issues surrounding the exploration and definition of architectures.

#### 4.1.2.1 What is a Reference Architecture?

First, a question was posed: What exactly is a reference architecture? The term has no formal definition, but is used loosely in many publications discussing product lines and reuse. More than one speaker at this workshop also used the term without definition.

A reference architecture is an architectural style and/or design pattern for a specific domain or product line. It is insufficient for building systems in that a style or pattern by its nature requires the binding of unresolved issues. In addition, an architectural style represents a set of constraints on architectures of individual products to be built as part of the product line, and therefore expresses the commonality among the architectures of the different products. The group envisioned the "incompleteness" in the reference architecture as perhaps not accounting for certain components that might be in the end products (or allowing for different components to occupy the "slots" defined in the reference architecture). This is as opposed to a sort of incompleteness where variation points in the architecture are completely described but left unbound until an individual product architecture is specified.

The notion that a reference architecture was insufficient, however, was controversial within the group—some of our members felt that this left the notion too vague to be of much value. These group members were much more comfortable with the notion that a reference architecture specifies completely the set of product architectures, although by using well-defined variabilities allows for different products to have different end architectures. For example, a reference architecture might specify that there might or might not be a component A present in the architecture, but if it is present there is a data flow connection between it and component B. Thus, some products might include component A and some might not, but the reference architecture completely describes both cases. A reference architecture under the "insufficient to build from" notion in the previous paragraph would simply omit component A from its description. Thus, under this more rigorous interpretation, a reference architecture is the same as any final product's architecture, except that it describes the variation points explicitly whereas the product architectures have those variations bound.

Is a reference architecture the same as a "product line architecture?" The group felt that it was, and that both terms were not needed.

The group agreed that a product line might in fact feature more than one architecture in its asset base. For example, different architectures might "wire together" the same set of components in a different way (using, say, different interconnection mechanisms) in order to achieve higher performance or greater reliability. In this case, we could consider these architectures either as truly separate entities, or as different branches off of some (possibly hypothetical) "master" architecture that admitted both candidate architectures as possible variations. Which alternative is to be preferred is a matter of convenience in the matter of keeping the core asset base up to date and in planning for future evolution of either or both architectures. In any case  the group was adamant in insisting that the core asset base include at least one architecture — else the asset base is nothing more than a component library, which is a paradigm for reuse that has largely failed.

In summary, the working group concluded that a reference architecture (when the term is applied in a product line context) is a partial or incomplete specification for a specific product in the product line—an "abstract architecture" establishing constraints on implementation, and a set of constraints on the architecture of the products in the product line.

#### 4.1.2.2 Role of Patterns and Styles

The group felt that architectural patterns and styles should play an important role in the exploration and definition of product line architectures, and for all the familiar reasons: known quality attributes associated with catalogued styles, shorter learning time, availability of associated infrastructure (such as test cases and test scaffolding), easier analysis, etc.

The choice of architectural style is fundamental to defining a product line architecture. Group members felt that a domain model for the product line area would hint at appropriate styles, based on their use in earlier similar systems. By identifying the entities of interest in a problem area, a domain model may also suggest styles or patterns that are best suited for representing those entities and their manipulation in a system.

They felt that design patterns were a finer-grained version of architectural styles, and hence occupied a later place in the design cycle, as implementation strategies were chosen for individual components. What are needed are domain-specific styles and patterns and more reliable (quality-based) style catalogs.

And how are architectures created for product lines in practice today? Layering is by far the most common style used, but “layering” can in fact mean any of a dozen or more things. In some layering schemes, usage is restricted to adjacent layers; in others, only the next lower layer; in still others, any lower layer. Also, layers do not necessarily dictate any finer-grained decomposition of components within layers. In fact, layering only speaks to allowed usage, and is not necessarily the result of a decomposition of the system into modules. It may be the case, for instance, that the programs of a single module reside in different layers: there is no architectural rule that requires modules and layers to be identical. Thus, layering (while useful) is only a start at the architecture exploration process, and the choice to use layers does not relieve the architect of the responsibility to make other partitioning decisions and additional style/pattern choices.

#### 4.1.2.3 Documenting Variation Points

An architecture for a product line differs from an architecture for a single system that evolves over time because it must support simultaneous variations as opposed to sequential ones. Hence, the specification of product line variation points, and variability mechanisms to support those variation points, are both key aspects of a product line architecture.

To let our group discuss the handling of variation points from an informed point of view, we asked Jeff Shaw to speak to our group about how variation points were handled by Raytheon in the CCT architecture. Shaw explained to us that variation points are part of the product line documentation beginning at the requirements stage. Each functional component in the CCT architecture is documented with inserted statements about their required variability. Further, there are use cases that explicitly “exercise” the variability capabilities needed to support the CCT product line. Finally, a special document called a Reuse Guide has been written that describes (among other things) how the builder of an individual product in the product line should go about his or her job in the context of the variation points that are included and the variability mechanisms that the architecture provides. Shaw shared with us that they have not yet solved the problems of navigation and traceability to their satisfaction, but have made strides to tie each variability mechanism in the architecture to a requirement or use case that calls for it.

Architectural mechanisms for variability include

- parameterization, or the use of compile-time parameters to define a family of components, the binding of which happens at individual-product-build-time
- polymorphism and inheritance, the object-oriented world’s way of tailoring objects based on abstract schema

- providing interfaces that are abstract (generic) with respect to specific implementation choices (such as the choice of algorithms to carry out a computation), and then providing one or more implementations that bind such choices in different ways but which are all faithful to the interface
- varying the number of components (e.g., servers), or adding/deleting certain components to/ from end products

### 4.1.3 Evolution of a Product Line

How do product lines evolve over time? What are the stimuli to evolution? How is evolution managed? How does evolution of a product line differ from evolution of a single system? These and other questions occupied the bulk of this working group's time.

Change is inevitable. As customers use members of the product line, they will levy new requirements. New technologies or new market conditions will impel the product line owners to update their software to stay current or retain market share. These forces for change compel a product line choice on how to respond:

- Ignore it: turn down a customer's request, decline the marketers' entreaties to modernize, etc.
- Adopt the change, and force the upgrade on all of the current product line customers, as well as new ones.
- Adopt the change partially, by splitting the product line: let one of the resulting sets of products go their own way, by not maintaining those products as part of the supported product family. Alternatively, the "orphaned" set of products could be formed into its own product line, supported by a slightly different set of core assets.
- Adopt the change, and generalize the core assets in such a way as to keep the affect set of products in the family, adding variability in such a way as to let unaffected customers continue using their products unaware of the change.

Which choice is made depends, of course, on both technical factors and market factors. Technical factors include how involved the change is to make, and the technical feasibility of generalizing the product line to let both the change and the pre-change products be nodes of a new variation point's decision tree. Market factors include the applicability of the change, the importance of not alienating certain customer groups, the organization's market ambitions, the costs involved, and so forth.

However the choice is resolved, the relevant procedures should be captured in a product line concept of operations (CONOPS). That document should lay out the standard practices for making such a decision, and then the practices for upgrading the core assets and (if so chosen) back-fitting deployed products to be consistent with the new versions of those assets.

Resolving procedural issues is critical to the survival of the product line. The goal is to preserve the integrity of the architectural vision (by preserving the integrity of the architecture itself). This is accomplished by making sure that every product continues to use the product line architecture and the assets in the core asset base only in approved variations. “Clone and own,” the practice of checking out a software component and freely changing it to meet the needs of a single product, is a major threat to the long-term survival of a product line. A concept of operations established ownership of the core assets, and the organizational mechanisms to be put in place to manage the usage and evolution of those assets. In the case of a product line acquisition situation, in which a large customer (such as the U. S. government) has commissioned the product line and the initial set of products, the concept of operations must also account for the customer’s role in levying evolutionary changes on the product line.

Aspects of the evolution issue that are unique to product lines can be summed up in the problems associated with the simultaneous support of multiple products, as opposed to the support of a single product that, while varying over its lifetime, exists in only one version at any moment. Single-system evolution issues, which are not particularly well-understood or codified in their own right, are greatly magnified. New product line customers receive the latest version, but earlier versions must be supported. Customers’ evolutionary pressures often conflict with each other. Changes to a product in the field must be folded back in to the product family in a way that does not disrupt the operations of customers unaffected by such changes.

Evolutionary issues associated with the core asset base involve the relation between the core assets (more precisely, the *versions* of the core assets) that were used to field particular products (or versions thereof). Changing a core software component is theoretically less expensive than changing all of the systems that are presumably affected by such a change; when possible, this is the preferred approach to evolution. This situation is made much more tenable by managing the requirements for the product line as a unified entity rather than on a cases-by-case (product-by-product) basis. Product-line-wide requirements evolution can be facilitated by having a user’s group of the products’ major customers, or by showcasing new requirements to the customer base as a whole rather than just those customers whose needs precipitated the changes. The goal behind core asset evolution is the reduction in the proliferation of product versions. Proliferation occurs when core assets are split into almost-alike versions. More desirable is to design components for which a split is not necessary, but handled as a pre-planned variation point.

An issue of core asset evolution with which an organization must come to terms is the granularity of new releases: should new components be released individually, or should the entire core asset base be re-released from time to time? Boeing opted for the large-grained approach. They felt it was

- good for core asset developers
- easier for configuration management
- easier for re-creating products for trouble fixes

It was, however, harder for product developers, since unwanted changes are forced on them along with desired changes.

The product line evolution issue applies to the product side of the equation only indirectly. If the products are all built faithfully using the core assets, then the issue does not apply—unless a product branches out of the product line to go its own way based on the unique requirements of its own customer base. But in that case, it isn't part of the product family anyway, and again the issue does not apply.

But evolution may occur in that part of a product that is not built from the core assets. This may lead to later product line updates, and product-specific components may need to be synchronized with core asset versions.

Risks associated with product line evolution include

- proliferation of alternate distinct versions of products and/or core assets
- abandonment of the product line by customers (especially since a change that benefits one customer may be detrimental to another)
- overload of pace of change. This could come from
  - asset utilization (as-is versus tailored)
  - bureaucracy
  - lack of expertise
  - cost
  - cumbersome or low-integrity architecture
  - component proliferation

## 4.2 Technical Management Practices

The technical management practices working group concentrated its discussion on the following two practice areas:

- Planning and Tracking
- Make, Buy, Mine, Outsource Analysis

### 4.2.1 Planning and Tracking

In the area of planning and tracking, the working group focused specifically on planning. They viewed tracking as an area that is related closely to data collection and metrics, which already is a practice area defined in the framework.

Planning was distinguished from the development of the business case, which has a planning aspect. The group viewed the business case, along with a statement of work, as providing an entry criteria for product line planning. Given this context, planning was addressed according to the overall outline of the framework's practice area template.

The group focused much more on what had to be addressed in plans then in planning per se. Also, it is clear that there must be a number of plans at different organizational levels to support a product line effort. The group did not tease those out.

Standard practices for software planning are widely available. The working group viewed such practices as broadly relevant for product line planning and therefore focused on areas where product line practice requires either a different focus or a greater emphasis.

#### 4.2.1.1 Aspects Peculiar to Product Lines

A prerequisite for the development product line plans is a thorough understanding of the implications of the product line approach, including an understanding of organizational management, technical management, and software engineering issues related to product lines. To achieve such understanding, one must either have experience in developing product lines or simulate experience by extensive reading about the basic principles and lessons learned from product line efforts. The SEI's Product Line Practice Framework was noted as an excellent source to help short circuit learning. In fact, the framework can be very helpful in pointing out areas that might require some improvement activity and therefore an improvement plan that will be executed prior to the actual product line effort.

In addition to the standard issues required for the planning of single software products, product line practice requires the development of plans at an organizational management level that address a number of issues in greater detail, such as

- organizational structure and decision making authority for core asset and product teams
- sponsorship
- incentives to product teams for reusing core assets and for contributing to the development of potential new core assets
- adequate resources for the maintenance of core assets
- factors that lead to a stovepipe mentality
- ownership of core assets
- resources for the acquisition and integration of necessary tools.

There are also plans needed for the technical work and the technical management, for example, a work plan for defining the architecture, a work plan for testing the core assets, plan for setting up the product line configuration management process, a technical risk management plan, etc. A checklist can be developed from the software engineering practice areas in the framework. Each of these practice areas should have an associated work plan.

### 4.2.1.2 How Applied to Core Asset Development and Product Development

Core asset development and product development were discussed together. There must be a plan that ensures that an overall strategy for the creation of core assets is developed. Core assets may be developed using either a top-down or a bottom-up approach. For example, an organization may start by developing a product line architecture and a set of core assets and then building products based on these assets. On the other hand there may be more of a bottom up approach that starts with products and uses these products as inputs to the core assets.

There must be plans that address the following issues related to the development of core assets:

- the development of a product line architecture
- the identification of core assets
- the development of interface definitions
- the identification of current assets that are available and their potential for reuse
- the definition of the configuration management process
- the development of interface definitions between components and policies for enforcing them

There must also be plans to support product development that address the following:

- maintenance and enhancements to products
- the movement of people between core asset and product teams
- the assembly of products from components

### 4.2.1.3 Specific Practices

The group did not identify any specific planning practices but did point out the need to establish standards that are based upon corresponding plans.

These standards include the following:

- the software development process to be used for core assets[10]
- guidelines for the development of appropriate documentation
- guidelines for mining assets

[10] The process here and the guidelines in subsequent bullets are referred to in the framework as individual *attached processes* for the core assets.

- guidelines for the assembly of products from components[11]
- guidelines for making tradeoffs between requirements
- guidelines for managing commonality and variability

#### 4.2.1.4 Risks

There is a risk that if an organization's product line stakeholders are not involved in developing plans for the product line effort, there will likely not be the buy-in necessary for success.

There are risks associated with the degree of specificity provided in the plans associated with the product line effort. If plans are too vague, there will not be sufficient rigor to ensure that practices are routinely executed and sufficiently coordinated. On the other hand if the plans are too specific or too rigid, there may not be enough flexibility to accommodate the changing dynamics of the organization's mission or business or the evolving maturity of the asset base.

### 4.2.2 Make, Buy, Mine, Outsource Analysis

The group next discussed issues related to deciding whether to make, buy, mine, or outsource the development of software assets. The same analytical approaches for making these decisions in a single-system software development environment ought to be employed. However, product lines introduce a new set of factors that need to be considered since the software assets have to be sufficiently general to support the multiple products in the product line and since the decisions that are made affect more than one system.  This latter aspect can be an advantage. The cost of implementing the decision can actually be amortized over multiple products so the cheapest route need not necessarily have to be selected.

The nature of the interfaces and how one will accommodate the required variability plays an important role in make/buy/mine/outsource analysis for product lines, as do the type of wrappers that could be used for legacy assets and their associated costs.

Other issues that were discussed include the following:

- The outcome of the make/buy/mine/outsource decisions can be considered to be both an input to core asset development as well as a constraint on it.
- If outsourcing is chosen, the interfaces need to be rigorously specified, thus providing additional constraints on the vendor.
- The integration of COTS components can have unpredictable results and side effects. This can lead to additional costs.

[11] Such guidelines are captured in the *production plan,* as described in the framework.

- There is a need for more up-front analysis of components and their interactions since the impact of these interactions will be eventually played out in multiple products.
- It is useful to build a mini business case for the decision associated with each asset

Risks in making make/buy/mine/outsource decisions include the following:

- The integration of COTS components and interactions between them can involve hidden costs.

  (This risk can be mitigated by up front analyses of the COTS components and their interactions.)
- Failure to adequately analyze the scope of the product line or the critical requirements can undermine individual asset decisions.
- Hidden interactions among components can produce unpredictable consequences.

## 4.3 Organizational Management Practices

The working group that focused on organizational management practices devoted most of their time to the Launching a Product Line practice area, and briefly touched on the Funding practice area.

### 4.3.1 Launching a Product Line

Launching a product line involves not just technical changes in an organization, but non-technical changes at both the enterprise and the business unit or program level. In order to effect such comprehensive change, it is essential to have high-level management buy-in, a business case, and advocacy at all levels.

The group felt that organizations should start with the following pivotal pieces:

- an explicit articulation of the need for change (usually tied to business or mission goals)
- definition of the product line goals, strategies, and vision
- a plan to garner advocacy

In order to define goals and strategies, the organization must analyze its own culture and embrace realistic strategies that can be effected within that culture. Many strategies can be potentially appropriate, but some are simply not viable in particular organizational cultures. The goals and strategies will drive the product line rollout or adoption plan, will define success, and should become the chief product line communication piece.

In securing advocacy for the product line effort, the DoD organization should start with the senior level since funding is needed to resource the organizational change. However, grassroots support is going to be needed; the “troops” should not be ignored. A bold communication campaign should be assembled early and should exploit the negative experiences of the past to convince the staff that a new approach is needed.

#### 4.3.1.1 The Need for a Plan

The group determined that there is a strong need for a product line launch or adoption plan. This plan should include the product line goals, vision, and strategies, identify the activities needed to execute the strategies, and stipulate the roles and responsibilities of those who are responsible for putting the plan into action.

Planning is iterative and so the plan should be built in increments. Action items that identify work needed to complete the plan should be documented and tracked. The logical relationships of the activities in the plan should be examined before attaching a duration or schedule estimate.

The planning group should be small and limited to the group who will actually be responsible for implementing the plan. Action groups or IPTs can later be established to execute major sections of the plan.

Since a significant part of any technology change is overcoming the natural resistance to change that is manifested in cultural/people issues and barriers, the adoption plan should include specific activities to work the people issues. The following activities were identified by the working group:

- Conduct a product line assessment to understand the current organizational strengths and weaknesses relative to product lines.
- Identify those individuals most experienced in the domain.
- Identify those in the organization who are experts in critical technologies; e.g., architecture, domain analysis, configuration management, process definition, component development, etc.
- Prepare a high-level pitch to convince the staff that a product line approach is in their best interest and that it is a way to gain/keep their professional competitive edge.
- Work a strategy for mining assets that embraces both the assets and the folks with asset expertise.
- Provide bonus awards for papers that are written on the new approach.
- Assign briefings to subordinates to help them feel a sense of ownership in the product line effort.
- Spread the championship role among a group with the talent and motivation to succeed in product lines.

#### 4.3.1.2 Launching Approaches

The group discussed two approaches for launching a product line: start with a pilot and start with a real product line

#### 4.3.1.2.1 Product Line Pilots

One approach for launching a product line effort is to start with a pilot or experimental product line. A pilot may actually be needed to provide sufficient data for the business case needed to justify a full-scale product line effort. Pilot is also a good way to test product line feasibility within the organizational culture.

A pilot approach is best when

- there is a problem but it is unclear that product lines are a good solution
- there is insufficient buy-in for a full-scale product line effort
- the organization lacks a cooperative culture
- there are limited resources to undertake a product line effort
- the feasibility of a product line approach has not been proven for the problem domain that the organization is focused on
- it is not the right time in the DoD budget cycle to fund a full-scale product line effort
- the organization is large and the pilot effort will be easily staffed

Once a pilot approach has been chosen, criteria should be developed in order to select an appropriate pilot. The working group identified the following criteria for the product line pilot. The pilot

- shouldn't be on a critical path
- should be important to the mission or business
- shouldn't require people who are already schedule-constrained
- should be tightly scoped and constrained, as any good experiment should be
- should have a pre-existing advocacy group

It is important to identify the goals of the pilot and appropriate measures associated with those goals. Data should be collected during the pilot implementation so that the measures can be tracked. The results of the pilot should be analyzed and then folded into the business case that would be used to justify a full-scale product line effort. A lessons learned document or pilot case study should be written so that the full-scale product line effort can exploit the successful pilot practices and address the pilot shortcomings.

#### 4.3.1.2.2 Product Line Launches Without Piloting

If a full-scale product line effort is going to be implemented without first running a pilot, an incremental approach should be chosen and a complete inventory of potential assets should be taken. These assets can be architecture, components, process definitions, tools, etc.

The product line effort should begin with a scoping activity that will yield a description of the products that are to be supported by the product line (the product line scope). Then a core asset production plan should be created that includes the following activities:

- needs assessment of methodology and tools
- identification of a common set of automated tools and development environments and provisions for site-wide licenses
- definition of a common methodology that selects a composition approach, a generative approach or a hybrid approach.
- definition and enforcement of common processes

Processes should be defined that are associated with the core assets and will be used as part of the production plan that specifies how the core assets will be used to build products. These processes should be lightweight initially and be made more “industrial strength” as they are applied, analyzed, and improvements are identified. The group discussed the pendulum pattern related to process definition that frequently shows up in organizations. The processes are light on details at first and then they are refined to include a great deal of specificity and detail. The organization gets overwhelmed with so much detail; the process definitions go through a distillation until a happy medium is reached. Stable processes are actually a good measure of success for the product line effort.

An organization launching a full-scale product line effort will likely need to do some organizational restructuring. It is important early on to identify roles and responsibilities. The working group discussed two communication approaches: (1) a group that rotates between core asset development and product development, and (2) a configuration review board that addresses modifications and extensions and communicates the information back to separate core asset and product groups. Action teams or IPTs that cross boundaries are also options.

Adequately staffing the effort is another primary concern. Architecture talent is a must. It may be necessary to buy the requisite architecture expertise. Some members of the working group had good experiences with this tactic. Include some developers on the architect team if they have the latent talent to grow into an architect’s role. Architecture training is critical.

It is also essential to identify who the ultimate product users are (in terms of roles and names) so that they can be involved as early as possible. Establishing a users group is also a good way to get the users to be come product line advocates.

A training plan for product developers will need to be produced. There must be training on the core assets (architecture, requirements, components, test plans, etc.) and on the product plan (the method, processes, and tools to be used to create products from the core assets).

Core assets can be mined from existing assets. Often mining isn't optional since it is necessary to preserve some of the DoD components for certification purposes. One approach to mining is to define the product line architecture first, then test the fit of individual legacy pieces, and determine the degree of effort required to make them conform. "Ugly" assets can be wrapped. The key is to keep the interfaces in the product line architecture stable. It is important to keep the architecture separate from the implementation and to spend sufficient time on architecture. There is always a tendency to rush to code. Sometimes it helps to create an architectural view that bridges the gap to people's previous frame of reference (e.g., a list of CSCIs mapped to class categories to help government and developers understand the architecture from their previous frame of reference. Another approach is to begin to deal with the legacy architecture, build variation points, and then incorporate the legacy components as appropriate. This approach works with local changes; in the process, the architecture often loses whatever integrity it may have had.

#### 4.3.1.3 Launching Risks

The following are risks related to launching a product line:

- Resisters can often undermine the product line effort. There is a need to engage the old hand architects from previous stovepipe systems though they may find the new approach difficult to accept. Committed managers must be enlisted into the team. Relationships at every level should be cultivated. The middle management layer is usually the most problematic. Get technology change training or hire a change consultant if problems emerge that can't be mitigated any other way.
- Senior management can get too enthusiastic too early and then be disappointed and lose interest when the results aren't immediate. There is a need to work to guarantee that management has reasonable expectations. A strong leader should head the core asset development effort so that management cannot brow beat the effort. It often helps to appeal to non-biased external consultants.
- There will be some recalcitrant people who won't ever get on board. The fact is all won't be enthusiastic immediately or ever. Some may have to move on to another organization.
- A planning group that is too large will never reach consensus and launch activities will be retarded.
- People don't have time or the desire to work on a pilot; as a result, the needed domain expertise may be missing. A product line effort, pilot or otherwise, will fail without sufficient domain expertise. The proper resources must be negotiated with management before the pilot is initiated.
- There is a temptation to grow the scope of the pilot such that successful completion of the pilot is at risk. Get agreement on the pilot scope and expectations up front and stick to the agreement.

### 4.3.2 Funding a Product Line

The working group briefly discussed some of the aspects related to funding a product line effort. There is the up front expense of core asset development. The architecture, components, requirement, tools, plans, and processes, all must be developed. The evolution of the core assets—their care and feeding—must also be funded. What are the funding options?

- Research and development funds can be used. Such funds are usually easier to arrange but do not usually provide a stable funding mechanism since research is usually the first area to be cut in times of economic crunch.
- Products can be taxed to support the core asset development and sustainment. Robust assets and a firm business case would be needed to convince the product units to pay the tax.
- The same customer can pay all—both the asset and product development and evolution.
- High-level management funds can be secured so that the cost is built into the margin. In the DoD, the program manager (PM)/program executive officer (PEO) is the right level to secure a program objectives memorandum (POM) for product lines, not projects.

## 4.4 Acquisition Management Practices

### 4.4.1 Defining and Implementing an Acquisition Strategy

The working group that focused on acquisition strategy noted that describing an acquisition strategy in the context of the Product Line Practice Framework is complicated by at least two factors. One factor is the scope of the acquisition. At a very high level (program manager level), an acquisition strategy describes a roadmap for program acquisition from initiation through post-deployment support. At a lower level, an acquisition strategy describes a contracting strategy for purchasing specific hardware or software. In between these extremes, other acquisition strategies may be necessary for acquiring systems or sub-systems comprising the product line. The other complicating factor is the interaction of this practice area with other practice areas in the framework. Among the many possible interactions, the group noted a heavy interdependence with the following practice areas:

- Product Line Scoping
- Building and Communicating a Business Case
- Funding
- Achieving the Right Organizational Structure
- Make, Buy, Mine, Outsource Analysis
- Customer and Supplier Interface Management

In the face of these complications, the group determined some simplifying assumptions. Future work could relax these assumptions and examine the effects. The assumptions made were the following:

- The acquisition would take place in a United Stated DoD environment. Important implications of this assumption are the related DoD governing regulations and the need to maintain fair competition.
- A business case has been established that justifies that a product line approach is appropriate.
- The product line has been appropriately scoped.
- The acquisition organization’s size, structure, and level within the government hierarchy has been taken into account.

Given these assumptions, the group then defined the term *acquisition strategy* and identified and explored some alternative product line acquisition strategies. At its essence, an acquisition strategy is a description of a process to attain key objectives and support specific business goals through the acquisition of products and services. Several possible product line acquisition strategies were then identified and explored. These strategies included the following:

1. Acquire a reference architecture.
2. Acquire a system[12] architecture.
3. Acquire a system architecture and set of components.
4. Acquire a product built using product line technology.
5. Acquire a product and some set of reusable assets.

The group’s discussions of each of these acquisition strategies for are summarized below.

## 4.4.2 Strategy 1: Acquire a Reference Architecture

### 4.4.2.1 Characteristics of This Strategy

In this case, the group considered a *reference architecture*[13] as one describing and constraining the characteristics of a *system architecture* to be built for a specific system. The ISO OSI 7-layer model for computer network protocols was given as an example. While initially keeping the acquiring organization out of the business of acquiring specific core assets and products, this strategy sets the stage for future acquisitions to build the products and assets. The group felt that this strategy provided a lot of flexibility that could support other strategies. This strategy was also seen as being less intrusive on the contractors and thus was in harmony with acquisition reform.

[12] This group used the term *system* interchangeably with the term *product*.

[13] The term reference architecture was also discussed and rejected in favor of *product line archiecture* by the software engineering working group. See Section 4.1 of this report.

#### 4.4.2.2 Risks of this Strategy

- The acquirer takes on accountability for shortcomings of the reference architecture, which may not be compatible with the acquirer's mission or talent pool. (Note that some form of this risk is applicable for most of the strategies.)
- The reference architecture might not meet quality attribute needs of all of the potential users—e.g., reliability and real-time performance requirements. This would limit its usefulness. (A mitigation strategy would be a performing an architecture evaluation.)
- Inability to foresee future requirements changes may hinder the architecture's usefulness.
- Conformance of individual system architectures to the reference architecture may be difficult or expensive to determine.

### 4.4.3 Strategy 2: Acquire a System Architecture

#### 4.4.3.1 Characteristics of this Strategy

In this strategy, the architecture for a specific system is acquired, setting the stage for future acquisitions of core assets and products of the product line that would include this system. This is a more system-specific strategy than Strategy 1. Various components, their interfaces and interactions are defined. This approach would likely be familiar to those who have written system A-specs. Because the system requirements are more strongly defined, this was seen as a more intrusive acquisition than Strategy 1.

#### 4.4.3.2 Risks of this Strategy

- Similar to the first risk identified for Strategy 1, the acquirer takes on more liability for shortcomings of the system architecture.
- The architecture may be flawed and this may not be discovered until much later without a specific system implementation to validate the quality and feasibility of the architecture.
- The architecture may not be suitable to support a product line, which will unravel the whole product line effort.

### 4.4.4 Strategy 3: Acquire a System Architecture and a Set of Components

#### 4.4.4.1 Characteristics of this Strategy

This strategy extends Strategy 2 by acquiring not only a system architecture but also a set of components. Subsequent contracts are then awarded to build systems from the architecture and components.

#### 4.4.4.2 Risks of this Strategy

As with any strategy in which the government provides assets, the acquirer takes on some liability for shortcomings of the assets. However, this strategy is seen as less risky than Strategy 2 since there is an opportunity to validate the architecture with its components.

#### 4.4.4.3 Example of this Strategy

The U.S. Army's Common Hardware/Software (CHS) system is an example of this strategy. This set of assets is based on a reference architecture for information systems. The Army requires program managers to use these assets for specific applications, such as Command, Control and Intelligence systems. The assets include a layered architecture plus components. All layers are provided except the application software layer.

### 4.4.5 Strategy 4: Acquire a Product Built Using Product Line Technology

#### 4.4.5.1 Characteristics of this Strategy

This strategy shifts the focus from the core assets to the product. In this case, the acquirer is interested in obtaining a system capability built through product line technology but without acquiring the core assets. The benefits to the acquirer are derived through the contractor's proprietary capability. The request for proposal (RFP) would require demonstration of a product line capability for bidders to be technically qualified. Assuming normal system evolution, a possible benefit is that the system would be more easily maintained. This strategy was seen as analogous to RFPs that require a certain level of process maturity for bidders to be technically qualified. A possible outcome is that as more contractors became proficient in product line technology, the government would benefit from their efficiency.

#### 4.4.5.2 Risks of this strategy

- The vision of general improvement in the contractors' capability might not be realized without incentives. Furthermore, it makes little sense for the government to incentivize development of this capability when the contractor will maintain ownership of the assets.
- There may be limited qualified bidders in a particular application domain, which will jeopardize the quality of the selection.
- The government may be subject to vendor lock-in.

### 4.4.6 Strategy 5: Acquire a Product and Some Set of Reusable Assets

#### 4.4.6.1 Characteristics of this Strategy

This strategy is an extension of Strategies 3 and 4 and addresses several of the risks identified with those strategies. In this case the quality of the architecture and components are more thoroughly demonstrated by building an actual system based on the assets. Also this strategy aligns with the natural iterative learning that takes place in establishing a product line. By proving that the assets can be used to build a system, valuable credibility for the core assets is established.

There are also other variations on this strategy that were not explored, such as additionally acquiring a reference architecture or creating components mined from legacy systems. A significant variation on this strategy that was considered is to acquire an *additional* system—i.e., the first *reuse* system that reuses the core assets. This approach allows the program to reap the benefits from the investment in building the reusable assets.

#### 4.4.6.2 Risks of this Strategy

- If the architecture and components for the product are not designed for future reuse—i.e., variation points have not been provided, there will be no product line, since the variation inherent among the products in the product line will not be possible.
- This strategy is more costly and requires longer to execute than the standard acquisition for a single product . If strong commitment to the product line approach does not exist, the program may have a hard time forcing reusability into the archiecture and assets or perhaps even in carrying on to completion.
- If a struggle arises between building for reuse and the need for delivered mission capability, mission will usually win. This would undermine the product line potential.
- If there is no preliminary scoping of what products the intended product line might eventually support with the set of acquired architecture and components, it is unlikely that these assets will be suitably flexible to support other products.
- Unless there is an investment in the development of a production plan and a product line concept of operations, it is unlikely that other programs will know how to use the developed assets and will then probably not use them.

### 4.4.7 Potential Crosscutting Issues

In addition to the strategies explored during the workshop, the group felt there were other, potentially crosscutting, issues that might affect the choice of acquisition strategy. These include the following:

- Ownership and data rights. What does it make sense for the government to own?
- Government roles and responsibilities. What things should the government be responsible for? What things should the contractor be responsible for?
- How does the government handle the sustainment and evolution of core assets?
- How are program managers incentivized to use a product line approach?

## 4.5 Product Line Concept of Operations

A product line CONOPS describes, at a high level, key organizational considerations that are necessary to implement a product line approach. The CONOPS is an important communications vehicle to identify the significant changes the organization will experience in moving to the new approach. These changes include such things as organizational structure, roles and responsibilities, and processes.  The CONOPS documents the decisions that define the product line approach and the organizational structure needed to put the approach into operation. The SEI technical report, *Guidelines for Developing a Product Line Concept of Operations*, in draft form at the time of the workshop, has since been completed [Cohen 99] and will assist organizations in creating a CONOPS for a product line.

The goal of the participants in the CONOPS working group was to explore the following issues raised by the CONOPS guidelines based upon their actual product line experiences:[14]

- Who is the audience for the CONOPS?
- What constitutes the product line?
- Who is/will be the product line champion?
- What are the key organizational elements?
- How will the product line be introduced?
- Is there an acquisition/contractor relationship for core assets? For the products created from the core assets?

As pointed out in the framework, there is a significant degree of variation among product lines and product line organizations. The working group discussions were intended to surface some of those variations as represented by the specific experiences of the participants. The group included members with varying levels of experience in implementing a product line approach, ranging from experience with several different product lines to the earliest stages of initiating a first product line. Discussion of key CONOPS issues also helped the participants gain insight into their own efforts and understand the differences that exist between individual product line approaches.

The following sections provide a summary of the participants' product line experience and then address key issues that relate to their product lines for the DoD.

## 4.5.1 Product Line Experience of the Working Group

To set the context for the subsequent discussion, each participant provided a brief description of his product efforts. The descriptions are summarized as follows:

- aircrew scheduler: Across the Air Force, there are currently several independent efforts to develop an aircrew scheduler. There are also many systems using legacy schedulers. The goal of this effort is to produce an architecture and related assets for a scheduler product line whose products can be integrated into larger command and control systems.
- helicopter avionics: This effort is just getting off the ground, so to speak. Currently, several contractors support a diverse fleet of helicopters. The goal will be to produce an architecture and related assets to better support a "common cockpit" concept and the integration of new subsystems.
- wargame modeling and simulation exercises: Current legacy simulations use a diverse set of models that require significant integration efforts for running large scale exercises. A

[14] Their comments and input were used to improve the guidelines before publication.

product line proof-of-concept demonstrated that an architecture for exercises could ease the integration of diverse models and better support the needs of wargame exercises. Currently, this activity is producing a case study to capture the technical approach and lessons learned. The goal is to establish this wargame exercise architecture and production plan as a product line.

- ground-based satellite command and control: This effort is producing a complete set of assets and a demonstration application. Currently, the effort is focusing on asset (architecture and component) documentation, component integration and testing, concept of operations, and analyzing results from the demonstration system.
- commercial automotive systems: This is a domain engineering effort to move from several legacy architectures to an integrated product line architecture. The key question: what is the process? One suggested approach is to take a successful system, evolve the architecture for new systems, mine assets and roll into a product line. The reality is that legacy components generally imply some inherent "architecture" but the need to integrate has shown that a new architecture is usually required. Another problem encountered is that individual areas view themselves as having their own core assets, their own path and schedule. There is a cultural barrier to break these stovepipe developments.
- test and evaluation range exercises: This effort is currently developing a reference architecture to support a product line of exercises. The reference architecture links together legacy assets into an exercise. The goal is to evolve to a full set of product line assets that can be shared across ranges.

These brief presentations raised several issues that were further discussed within the context of the CONOPS guidelines:

- organizational
  - identifying roles and responsibilities of stakeholders; establishing organizational groups
  - adopting a common approach and schedule; breaking down cultural barriers
  - using the CONOPS to "sell" the product line approach
  - funding the domain engineering effort
- asset development
  - creating assets from on-going efforts and from legacy products
  - adopting a common data set as a product line asset
- product development
  - clarifying the role of conops as part of developing products in a product line
  - representing the "distance" between the assets and product systems

### 4.5.2 Establishing the CONOPS Audience

The CONOPS guidelines recommend that the product line organization define the stakeholder community. That stakeholder community defines the audience for the CONOPS. This part of the working group discussion addressed variations in the audience for the product line:

- aircrew scheduler: A product line architecture/component group will be developing assets for this product line. The CONOPS must also address need of the System Program

Offices (SPOs) who will be acquiring systems that use scheduler products, the organizations who will be users of those systems, and a steering group.

- helicopter avionics: The CONOPS audience will include the product line developers, local Army users as well as potential Army-wide users.
- wargame modeling and simulation exercises: The audience includes the program office (including associated contractors) and modeling and simulation groups.
- ground-based satellite command and control: Product line development is being performed by a contractor. The CONOPS audience includes the contractor, the program office, and command and control users.
- test and evaluation range exercises: The audience for this product line is primarily the range developers and range users for exercises. The collection of ranges constitute an "enterprise" that can provide range systems to support a diverse set of exercise requirements.

The group recommended that the guidelines address the range of audiences and possible needs for different stakeholders in the audience.

### 4.5.3 What Constitutes the Product Line (i.e., Scope)

The CONOPS provides specific recommendations and examples to help identify the product line and its scope. Often, the product line developers confuse the product line with the assets. It is necessary to place the product line in the context of the use of systems in the product line. In this part of the discussion, the participants described the product line boundaries:

- aircrew scheduler: The systems in this product line will be integrated into a larger command and control system. It is envisioned that the scheduler architecture and components will include: the user interface, output reports, database and communications interfaces, and a common data model. They must account for variability at the interface level.
- helicopter avionics: The product line is a common cockpit that integrates a variety of avionics and weapons systems.
- wargame modeling and simulation exercises: The product line here and in the test and evaluation range is exercises. This description is intended to address a key product line goal - the elimination of repeated integration of multiple models (or range assets) into an exercise. The wargaming product line is developing specific tools to support the production of systems in the exercise product line.
- ground-based satellite command and control: The product line here encompasses an integrated set of subsystems to support satellite command and control applications. Excluded is the man-machine interface.
- test and evaluation range exercises: The architecture will integrate product line and legacy component assets to support live exercises.

A new issue was raised here: what is the "conceptual distance" between the assets and the systems produced using the assets. For the aircrew scheduler, the assets, when assembled, will produce a complete scheduler. For the satellite command and control, major parts of a complete system are not parts of the asset base and assets themselves leave a high degree of tailoring up to the product developer.  The evolutionary nature of the range exercise product line means that early system (exercises) must do a considerable amount of development to complete a system. The reference architecture will always serve as a basis for local system architectures. The local range must still tailor assets for its own needs and to support individual range exercises. This gap between the assets and products is a technical issue, but the CONOPS must discuss the steps necessary to go from the assets to a product line system, which will  spelled out in more detail in the production plan. The guidelines should highlight the "conceptual distance" as an area of variability.

### 4.5.4 Product Line Champion Requirements

The product line champion has the vision and the authority to direct the product line.  The participants highlighted several key requirements of a champion for their organization:

- aircrew scheduler: The champion must be able to link the asset development group with the SPOs who will be acquiring systems.
- helicopter avionics: The headquarters level must champion the development.
- wargame modeling and simulation exercises: The champion must be a strong advocate of the tool-based approach for assembling exercises.  This requires the ability to work with modeling and simulation groups who until now have acted independently.
- ground-based satellite command and control: The champion must be able to relate to a diverse community of potential users and promote and provide vision for a long-term (up to ten year) approach.
- test and evaluation range exercises: The range community is diverse and well-entrenched. The champion must be able to address their concerns over control of local assets and integration into a large range enterprise.

It became evident that there are organizational idiosyncracies that must be exposed in order to succeed.

### 4.5.5 Organizational Elements

The CONOPS guidelines describe a basic set of roles including architecture and component development, product line support (environments, repository), and product developers.  There are also other roles such as oversight, change control, and user representation.  The participants discussed the roles they have identified.  Because of the different levels of maturity of the product lines, there is some variation in the roles they described:

- aircrew scheduler: The architecture and component roles have been identified.  Also the SPOs, as user representatives, and an oversight group are participating.
- helicopter avionics: There is an existing oversight organization and an architecture group just forming.
- wargame modeling and simulation exercises: Several levels of command have oversight from strategic level in the Pentagon to local command level.  There is a user representative role and a tool developer role.

- ground-based satellite command and control: The product line office has established a set of product line groups: architecture, component, test, application, and sustainment.  In addition, there are oversight groups.
- test and evaluation range exercises: The architecture role has been defined.  There is also a prototyping group that will be using the architecture in proof-of-concept demonstrations.

### 4.5.6 How the Product Line Will  Be Introduced

Each participant described their current plans for asset development and initial production of systems in the product line.  The CONOPS guidelines stress the importance of establishing the connections between the process for asset and product development:

- aircrew scheduler: The product line development group will produce an architecture based on an examination of several systems plus available off-the-shelf scheduler engines. The current operational concept is to capture assets from an on-going scheduler development. These will be modified to align with architecture for use in product line systems.
- helicopter avionics: It is too early in the development to establish specific plans. However, the need for ease in integration of new hardware boxes will drive product line introduction.
- wargame modeling and simulation exercises: The product line group has developed a proof-of-concept using a common architecture to integrate two models.  The next activity in product line introduction will be to sell the concept through a case study.
- ground-based satellite command and control: The product line will be introduced in phases.  In the first phase, now nearing completion, the product line groups are delivering assets for use in test and a limited number of operational systems.  In a sustainment phase, the assets will be used in a larger number of systems and will be enhanced to reflect the requirements of those new systems.  A final phase will see improved processes for producing new systems in the product line.
- test and evaluation range exercises: The current effort focuses on using the reference architecture in a small number of demonstration projects. If these projects show feasibility, the reference architecture will be used in actual exercises.

### 4.5.7 Acquisition/Contractor Development

The CONOPS guidelines describe acquisition and contractor relationship in terms of commissioning a product line:

- aircrew scheduler: The product line development group obtains systems components from a product development group.  It uses these components for producing product line assets.  Product line development also produces complete subsystems for use by other product development groups.

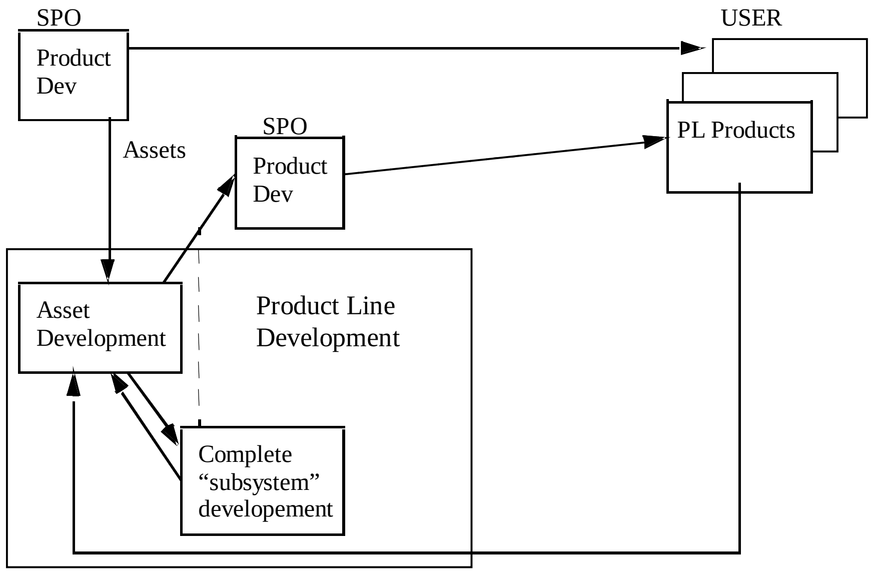

*Figure 2: Operations in Aircrew Scheduler Example*

- helicopter avionics: Too early in development.
- wargame modeling and simulation exercises: The product line group develops components and tools for use with the architecture. The tools integrate models for exercises. Models come from COTS, government off-the-shelf (GOTS) or may be developed within the product line group.

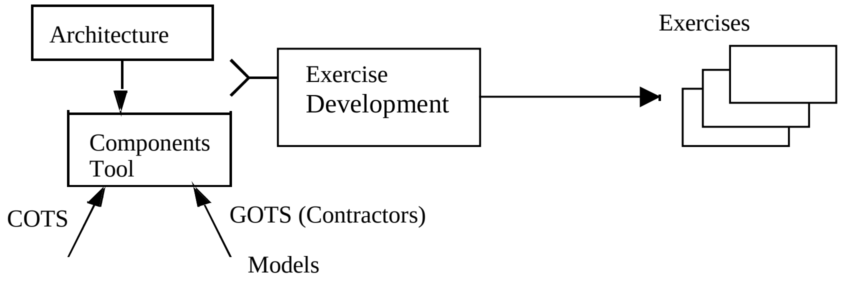

*Figure 3: Operations in Wargame Modeling and Simulation Exercises Example*

- ground-based satellite command and control: Government has commissioned a contractor to development assets. These assets will be used by contractors (possibly different from the contractor who developed the assets) to produce products.
- test and evaluation range exercises: To date, all development has been in-house within the DoD ranges. After the proof-of-concept, there may be commissioning of further asset development.

Possible development scenarios, such as those described by the participants, would help users of the guidelines establish their approach.

### 4.5.8 Challenges and Risks

The CONOPS guidelines provide a number of potential risk areas. The participants described their primary concerns:

- aircrew scheduler: ability to draw upon ongoing program for product line assets. Organizations wed to their own systems
- helicopter avionics: maintaining user satisfaction, both with new systems and with the new approach
- wargame modeling and simulation exercises: obtaining buy-in from other modeling and simulation groups
- ground-based satellite command and control: customer service for long-term support; organizations wed to their own systems
- test and evaluation range exercises: ability to support needs of a highly diverse community

### 4.5.9 Summary

The first version of the CONOPS guidelines definitely benefited from the working group discussions. These discussions raised new issues, provided insight into issues already covered, and provided feedback on areas currently covered. Moreover, they demonstrated how the guidelines get translated and applied in particular DoD organizational contexts.

The working group participants reported several positive outcomes:

- understanding the utility of the CONOPS guidelines
- raising new issues (e.g., distance between assets and products)
- help in capturing issues in fielding a product line
- identification of stakeholders not yet recognized
- need for support group role
- seeing hands-on experience in product line development
- clarification of more of the “pieces” of product lines (shareholders, roles and responsiblities, kinds of information product line must generate)
- seeing other participants/applications; able to relate their experiences to own need

# 5 Feedback From Workshop Participants

Following the working group reports, the participants were asked to provide feedback on the framework and the workshop. Evaluation forms were also completed and mailed after the workshop.

## 5.1 Comments on the Framework

Most of the workshop participants had used the framework in their product line efforts and investigations. They were very positive about the existence, the format, and the content of the framework, and the SEI's willingness to take the lead in providing such a resource. They offered the following constructive comments on the framework:

- The framework should address the role of models in product lines (principally domain models and architectural models), and how these models fit together. They suggested that the command, control, communications, computer, intelligence, surveillance and reconnaissance (C4ISR) architecture models should be mentioned, and how they relate to the concept of product lines should be made clear.
- The framework should include "patterns" for how to carry out practice steps.
- One participant shared with us a model of product lines that may not be adequately admitted by our two-sided "core asset versus product" view taken in the framework. This person's company markets two kinds of products. One kind corresponds to the kind of products we've discussed in the framework: stand-alone systems built using a base of core assets. Another class of products, however, they call COTS products. These are best thought of as components, usually not stand-alone, but filling a niche in various domains' component markets. They are bona fide products in the sense that customers buy them. The complication is this: These so-called COTS products, some of which are fairly complex, are built using the core assets in this company's asset base. The other products are built using the company's own COTS products as components. The question then becomes, are these COTS products part of the core asset base (clearly they are, with respect to the stand-alone products) or part of the product side of the product line (clearly they are, with respect to themselves)?
- The practice area definitions in the framework should be reviewed to ensure consistent format and style.
- Make the format of the practice areas more bulletized. Be sure each practice area is crisp.
- Distinguish the purpose of the framework clearly from that of the Capability Maturity Model ® (CMM®).
- Be sure to distinguish the term "practice area" from the "key practice area" concept of the CMM.

- Establish connections between the practice areas in the framework whenever appropriate.
- Consider a reorganization of the current planning and tracking practice. “Planning” can become a practice of its own, while “tracking” can be integrated with data collection and metrics.
- Maintain a clear delineation between organizational management, technical management, and software engineering practices.
- Create templates for developing product line plans and use successful plans as examples.

The framework author team addressed many of these comments in Version 2.0 of the framework and plans to address the remainder in Version 3.0.

## 5.2 Comments on the Workshop

The participants felt that the workshop was a good investment of their time. Participants commented that they came away with increased knowledge; materials that they reported using immediately in slide presentations, briefings, and practices within their organizations; important contacts within the DoD community who share product line interests; and renewed commitment to their own product line efforts. They appreciated SEI initiative in organizing and staffing the workshop.

There were also some negatives:

- not enough time for discussion
- need for broader representation (of people/efforts) to cover process.
- additional exchange with other working groups or participation in other discussions (possibly extend workshop an additional half-day)

# 6 Summary

The SEI's Second 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 validated the pivotal pieces of the SEI's Product Line Practice Framework and provided additional content for currently defined practice areas as well as those that are in the process of being defined. Challenges within the DoD community were discussed.

The working groups focused on specific practices within software engineering, technical management, and organizational management, as well as acquisition strategies for product lines, and a concept of operations for product lines. The emprical 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.

This workshop marked a turning point from the SEI perspective in that the DoD participants talked about how they were doing or going to do product lines versus how it would be impossible to do so within the DoD (a familiar lament from past DoD forums). In an effort to grow both the information base and the community interested in software product lines, the SEI was encouraged by the participants to continue to hold similar workshops and to continue reporting the workshop results to the DoD.

The results of this workshop are currently being incorporated into the framework, which will continue to be refined and revised as the technology matures and as we continue to receive feedback and to work with the growing community of software engineers 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 lmn@sei.cmu.edu.

# References

| | |
|---|---|
| **Bass 97** | Bass, Len; Clements, Paul; Cohen, Sholom; Northrop, Linda; & Withey, James. *Product Line Practice Workshop Report* (CMU/SEI-97-TR-003, ADA327610). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997. Available WWW: <URL: 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 Longman, Inc., 1998. |
| **Bass 98b** | Bass, Len; Chastek, Gary; Clements, Paul; Northrop, Linda; Smith, Dennis; & Withey, James. *Second Product Line Practice Workshop Report* (CMU/SEI-98-TR-015, ADA354691). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/98.reports/98tr015/98tr015abstract.html>. |
| **Bass 99** | Bass, Len; Campbell, Grady; Clements, Paul; Northrop, Linda; & Smith, Dennis. *Third Product Line Practice Workshop Report* (CMU/SEI-99-TR-003, ADA361391). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/99.reports/99tr003/99tr003abstract.html>. |
| **Bergey 98** | Bergey, John, et al. *DoD Product Line Practice Workshop Report* (CMU/SEI-98-TR-007, ADA346252). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/98.reports/98tr007/98tr007abstract.html>. |

| | |
|---|---|
| **Brownsword 96** | Brownsword, Lisa & Clements, Paul. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR-016, ADA315802). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available WWW: <URL: http://www.sei.cmu.edu/publications/ documents/96.reports/96.tr.016.html>. |
| **Clements 99** | Clements, Paul & Northrop, Linda. *A Framework for Software Product Line Practice, Version 2.0*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, July 1999. Available WWW: <URL: http://www.sei.cmu.edu/plp/framework.html> |
| **Cohen 99** | Cohen, Sholom. *Guidelines for Developing a Product Line Concept of Operations*, (CMU/SEI-99-TR-008, ADA367714) Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1999. Available WWW: <URL: http://www.sei.cmu.edu/publications/documents/ 99.reports/99tr008/99tr008abstract.html>. |
| **Gamma 95** | Gamma, E.; Helm, R.; Johnson, R.; & Vlissides, J.; *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995. |
| **Sharp 98** | Sharp, David C. “Reducing Avionics Software Cost Through Component-Based Product Line Development.” *Proceedings of the 17th Digital Avionics Systems Conference (DASC).* Oct. 31–Nov. 7, 1998, Bellevue, WA. New York: IEEE, 1998. |
| **Withey 96** | Withey, James. *Investment Analysis of Software Assets for Product Lines* (CMU/SEI-96-TR-010, ADA315653). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. Available WWW:<URL: http://www.sei.cmu.edu/publications/documents/ 96.reports/96.tr.010.html>. |

# Glossary

| | |
|---|---|
| **acquisition** | The process of obtaining products and services via contract |
| **acquisition strategy** | A plan of action for achieving a specific goal or result through contracting for products and services |
| **acquisition plan** | The artifact that is typically used to document the acquisition strategy |
| **application engineering** | An engineering process that develops software products from partial solutions or knowledge embodied in software assets |
| **business model** | A framework that relates the different forms of a product line approach to an organization's business context and strategy |
| **core asset** | A software artifact that is used in the production of more than one product in a product line<br><br>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. |
| **domain** | An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area |
| **domain analysis** | The process for capturing and representing information about applications in a domain, specifically, common characteristics and reasons for variability |
| **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 produce a greater variety of outputs<br><br>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. |

| | |
|---|---|
| **investment analysis** | The process of estimating the value of an investment proposal to an organization<br><br>Investment analysis involves quantifying the costs and benefits of the investment, analyzing the uncertainties, and constructing a spending strategy. This analysis links the strategic and technical merits of an investment to its financial results. |
| **platform** | Core software asset base that is reused across systems in the 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 specific needs of a selected market or mission area |
| **product line practice** | A system of software production that uses software assets to modify, assemble, instantiate, or generate a line of software products |
| **product line architecture** | A description of the structural properties for building a group of related systems (i.e., product line), typically the components and their interrelationships<br><br>The guidelines about the use of components must capture the means for handling variability discovered in the domain analysis or known to experts. (Also called a reference architecture.) |
| **product line system** | A member of a 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, which consists of software components, the externally visible properties of those components, and the relationships among them [Bass 98a] |
| **system architectures** | Software architecture plus execution and development environments |

| REPORT DOCUMENTATION PAGE | | | *Form Approved* *OMB No. 0704-0188* |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503. | | | |
| 1. AGENCY USE ONLY (LEAVE BLANK) | | 2. REPORT DATE<br>October 1999 | 3. REPORT TYPE AND DATES COVERED<br>Final |
| 4. TITLE AND SUBTITLE<br>Second DoD Product Line Practice Workshop Report | | | 5. FUNDING NUMBERS<br>C — F19628-95-C-0003 |
| 6. AUTHOR(S)<br>John Bergey, Grady Campbell, Paul Clements, Sholom Cohen, Lawrence Jones, Robert Krut, Linda Northrop, Dennis Smith | | | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Software Engineering Institute<br>Carnegie Mellon University<br>Pittsburgh, PA 15213 | | | 8. PERFORMING ORGANIZATION REPORT NUMBER<br>CMU/SEI-99-TR-015 |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>HQ ESC/DIB<br>5 Eglin Street<br>Hanscom AFB, MA 01731-2116 | | | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER<br>ESC-TR-99-015 |
| 11. SUPPLEMENTARY NOTES | | | |
| 12.A DISTRIBUTION/AVAILABILITY STATEMENT<br>Unclassified/Unlimited, DTIC, NTIS | | | 12.B DISTRIBUTION CODE |
| 13. ABSTRACT (MAXIMUM 200 WORDS)<br>The Software Engineering Institute (SEI) held the Second Department of Defense (DoD) Product Line Practice Workshop in March 1999. The workshop was a hands-on meeting to identify industry-wide best practices in software product lines; to share DoD product line practices, experience, and issues; and to discuss ways in which the current gap between commercial best practice and DoD practice can be bridged. This report synthesizes the workshop presentations and discussions. | | | |
| 14. SUBJECT TERMS<br>commercial product line practice, DoD product line practice, product line practice framework, product line workshop, software architecture, software assets, software product lines | | | 15. NUMBER OF PAGES<br>66 pp.<br>16. PRICE CODE |
| 17. SECURITY CLASSIFICATION OF REPORT<br>UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>UNCLASSIFIED | 20. LIMITATION OF ABSTRACT<br>UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102