Product Line Systems Program

Linda Northrop
Director, Product Line Systems

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Product Line Systems Program

Our mission:

• create, mature, apply, and transition technology and practices

• to effect widespread, architecture-centric development and evolution, verifiable and predictable software construction, and product line practice

• on systems at all scales throughout the global software community.

Strategic Relevance

This work directly enables

• predictable product qualities during design, evolution, and construction

• dramatically reduced cost for software-intensive systems through reduced test and integration time and economies of scale
Our Portfolio of Work

Software Architecture

(Software Architecture Technology Initiative)

Predictable Software Construction

(Predictable Assembly from Certifiable Code Initiative)

Software Product Lines

(Product Line Practice Initiative)

Ultra-Large-Scale Systems
Program Technical Themes

Product-centric
Quality attribute focus
Architecture importance
Predictability
Efficiency
Business and mission goals
Stakeholder involvement
Automated support
Our Customers and Collaborators

ABB
Boeing
Daimler Chrysler
Caterpillar
Federal Express
Foliage
General Dynamics Viz
Gereral Motors
Intuit
NCR
Northrop Grumman
Ortho Clinical Diagnostics
Pitney Bowes
Raytheon
RIM
Robert Bosch Co.
Siemens
Unisys
Visteon
LLNL
FAA
NASA: JSC, KSC, JPL
NASA: Goddard
NRO: CCT
JNIC
DMSO

Philips
Lucent
AT&T
Hewlett Packard
Thomson-CSF
Ericsson
Schlumberger
Nokia
Telesoft S.p.A.
Boeing
CelsiusTech
Avaya
Fraunhofer
IBM
Microsoft
Motorola
Cummins, Inc.
General Motors
Lockheed Martin
Salion, Inc.
MarketMaker
Argon Engineering
Agilent

US Army: ASA(ALT), Aviation, TAPO, BC, FBCB2, CECOM, ATSC, FCS, AMTS, WinT, IBS,
US Navy: Navair, DDX, OAET, CLIP
US Air Force: F-22, JMPS, ESC
Today’s Presentation

Software Architecture
  (Software Architecture Technology Initiative)
Predictable Software Construction
  (Predictable Assembly from Certifiable Code Initiative)
Software Product Lines
  (Product Line Practice Initiative)
Ultra-Large-Scale Systems
Universal Business Goals

- High quality
- Quick time to market
- Market agility
- Product alignment
- Low cost production
- Low cost maintenance
- Mass customization
- Mind share

require

IMPROVED EFFICIENCY AND PRODUCTIVITY
Software Strategies Are Needed

Business and Mission Goals

System (Software) Strategies

- Process Improvement
- Improved Architecture Practices
- process quality
- product quality
Focus: Software Architecture

From our experience:
The quality and longevity of a software-intensive system is largely determined by its architecture.

Many large system and software failures point to
- inadequate software architecture education and practices
- the lack of any real software architecture evaluation early in the life cycle

Risk mitigation early in the life cycle is key.
- Mid-course correction is possible before great investment.
- Risks don’t become problems that have to be addressed during integration and test.
Software Architecture Value Proposition

Using architecture-centric practices throughout the software development lifecycle and throughout the lifetime of a software-intensive product leads to

- early identification of important product qualities resulting in higher contract win rates
- early identification and mitigation of design risks resulting in fewer downstream, costly problems
- cost savings in integration and test
- predictable product quality supporting the achievement of business and mission goals, which translates into competitive advantage
- cost-effective product evolution
What Is a Software Architecture?

“The software architecture of a program or computing system is the structure or structures of the system, which comprise the software elements, the externally visible properties of those elements, and the relationships among them.”

Why Is Software Architecture Important?

The right architecture paves the way for system success. The wrong architecture usually spells some form of disaster.

Represents *earliest* design decisions

- hardest to change
- most critical to get right
- communication vehicle among stakeholders

*First* design artifact addressing

- performance
- modifiability
- reliability
- security

Key to systematic *reuse*

- transferable, reusable abstraction
SEI Software Architecture Technology (SAT) Initiative’s Focus

Ensure that business and mission goals are predictably achieved by using effective software architecture practices throughout the development lifecycle.

“Axioms” Guiding Our Work

- Software architecture is the bridge between business and mission goals and a software-intensive system.
- Quality attribute requirements drive software architecture design.
- Software architecture drives software development throughout the life cycle.

Earliest work focused on the second axiom leading to the Architecture Tradeoff Analysis Method® (ATAM ®) for architecture evaluation.
Conceptual Flow of the ATAM®

Business Drivers → Quality Attributes → Scenarios

Software Architecture → Architectural Approaches → Architectural Decisions

Analysis

impacts → distilled into

Risk Themes → Tradeoffs → Sensitivity Points → Non-Risks → Risks
Architecture-Centric Activities

Architecture-centric activities include the following:

• creating the **business case** for the system
• understanding the **requirements**
• creating and/or selecting the architecture
• documenting and communicating the architecture
• analyzing or evaluating the architecture
• setting up the appropriate **tests and measures** against the architecture
• **implementing** the system based on the architecture
• ensuring that the implementation **conforms** to the architecture
• evolving the architecture so that it **continues to meet business and product goals**
ATAM® Led to the Development of Other Methods and Techniques

What if the quality requirements are not well-understood?
Quality Attribute Workshop (QAW)

What if there’s no architecture?
Attribute Driven Design (ADD)

What if I don’t know my system’s architecture?
Architecture Reconstruction using ARMIN

Our scenarios tend to be incomplete or ambiguous.

What are some of the most important questions to ask?
Quality Attribute Tactics

Which risks should I work on first?
Cost Benefit Analysis Method (CBAM)

What information should be included in my architecture documentation?
Views and Beyond Approach (VaB)
Characteristics of SEI Methods

- are explicitly focused on quality attributes
- directly link to business and mission goals
- explicitly involve system stakeholders
- are grounded in state-of-the-art quality attribute models and reasoning frameworks
- are documented for practitioner consumption
- are applicable to DoD challenges and DoD systems

QAW
ADD
Views and Beyond
ATAM
CBAM
ARMIN
ATAM® Led to the Development of Other Methods and Techniques

What if the quality requirements are not well-understood?

Business / Mission Context

What if there's no architecture?

System Context

What information should be included in my architecture documentation?

Technology Context

What if I don't know my system's architecture?

Architecture Reconstruction using ARMIN

Quality Attribute Workshop (QAW)

Attribute Driven Design (ADD)

Views and Beyond Approach (VaB)

What are some of the most important questions to ask?

Organization Context

Which risks should I work on first?

Attribute Tactics

Attribute General Scenarios

Cost Benefit Analysis Method (CBAM)

What if there's no architecture?

What if the quality requirements are not well-understood?

Our scenarios tend to be incomplete or ambiguous.

Risks

Sensitivity Points

Non-Risks

Tradeoffs

Analysis

Scenarios

Architectural Decisions

Architectural Approaches

Software Architecture

Business Drivers

Quality Attributes

Distilled into

What information should be included in my architecture documentation?

What are some of the most important questions to ask?

Which risks should I work on first?
Architecture Evolution

Problem

• The architecture of a software intensive system must continually evolve to ensure consistency between the system and its mission and business goals.
  – “Tactical evolution” focuses on change over a short time horizon to ensure system consistency with current goals.
  – “Strategic evolution” focuses on change over a long time horizon to manage uncertainty in future goals, exploit future opportunities, and defend against future risks.

Approach

• Explore design space using quality attribute tactics, patterns, and tradeoff analysis.
• Use ideas from economic such as real options, utility theory, combinatorial optimization, and release planning.
Architecture Competence

Problem

- Organizations need help in measuring and improving the architecture competence of their individuals and teams in order to effectively use and benefit from architecture-centric software engineering practices.

Approach

- Determine factors contributing to architecture competence based on surveys, exemplar practices and our experience
- Develop assessment and improvement instruments based on those factors
- Explore relevant models such as those from
  - Organizational coordination mechanisms
  - Human performance model
  - Organization learning
System ATAM

Problem

- Severe integration and runtime problems arise due to inconsistencies in how quality attributes are addressed in system and software architectures.
- This is further exacerbated in a System of Systems (SoS) context where major system and software elements are developed concurrently.

Approach

- Make minor enhancements to the ATAM for use on system architectures.
- Develop a method to perform a "first pass" identification of inconsistencies between constituent systems of SoSs by using mission threads augmented with quality attribute concerns.
Architecture-Related Technology

Problem

- Prevailing technology and technology trends can both enable and be inimical to sound architecture practices.
- Guidance is needed.
- Architecture practices are often labor intensive and error prone.
- Automated support can help.

Approach

- Scrutinize technology and technology trends through the lens of architecture-centric development and provide guidance and support
  - SOA, from a quality attribute point of view
  - impact of open source on architecture and vice versa
- Identify technology gaps related to architecture practices and provide guidance and build prototype tools
  - reconstruction and conformance technology (with PACC)
  - ArchE, an architectural design assistant
Transition

Foster widespread awareness
- Books
- Reports
- Presentations
- SATURN, ATAM Lead, DoD, and Educators Workshops

Enable others
- Course licensing
- Certificate Programs
- ATAM Lead Evaluator Certification
- ArchE

Ensure practicability
- Methods
- Case studies
- Acquisition guidelines
- Technology investigation

Assist others
- Teaching
- Applying methods and techniques
- Providing expertise

Transition Products and Services
# Certificate Program Course Matrix

## Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Software Architecture Professional</th>
<th>ATAM® Evaluator</th>
<th>ATAM® Lead Evaluator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Architecture: Principles and Practice</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Documenting Software Architectures</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Software Architecture Design and Analysis</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Software Product Lines</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAM® Evaluator Training</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ATAM® Leader Training</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>ATAM® Observation</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Architecture Tradeoff Analysis Method® (ATAM®)
Associated Texts

Software Architecture in Practice, 2nd Edition

Documenting Software Architectures: Views and Beyond

Evaluating Software Architectures: Methods and Case Studies

Software Product Lines: Practices and Patterns
SAT Accomplishments

- Individuals from more than 400 different companies have taken courses in the SEI Software Architecture Curriculum.
- The SEI ATAM® was used to uncover risks in major DoD systems; for example, Win-T, ABCS, DDX, FBCB2, and FCS
- Teams at Raytheon, Boeing, and Robert Bosch GmbH are routinely conducting architecture evaluations using the ATAM.
- Representatives from ten U.S. Army programs reported that ATAM® evaluations resulted in reduced risk in schedule and cost, improved documentation and communication, and a higher quality product for the warfighter.
- The SEI books on software architecture have sold over 110,000 copies.
New Challenges

**Challenge**: Developing empirical and theoretical foundations for our competence assessment and improvement instruments.

**Our Research**: Applying the foundations of organizational learning and organizational coordination to the field of software architecture.

**Challenge**: Developing techniques to manage the uncertainty associated with strategic architecture evolution.

**Our Research**: Applying models and techniques from economics (such as real options and utility theory), release planning and product marketing (such as conjoint analysis) to the field of software architecture.

**Challenge**: Extend the concepts of software architecture to ultra-large-scale systems.

**Our Research**: Potentially develop new notions of architecture based on ideas from mechanism design and emergent behavior in complex (biological) systems.
Today’s Presentation

Software Architecture
   *(Software Architecture Technology Initiative)*
Predictable Software Construction
   *(Predictable Assembly from Certifiable Code Initiative)*
Software Product Lines
   *(Product Line Practice Initiative)*
Ultra-Large-Scale Systems
Software Strategies are Needed

Business and Mission Goals

System (Software) Strategies

- Process Improvement
  - Process Quality
  - Improved Architecture Practices
  - Improved Construction Practices

- Product Quality
The PACC Initiative

Predictable: runtime properties of an assembly
Assembly: easy but controlled integration
Certifiable: runtime properties of software
Code: executable code

It’s PACC!
PACC Focus and Axioms

Focus:
To introduce into routine practice the use of programming techniques that result in systems that exhibit quantifiably predictable runtime qualities.

Axioms:
• Runtime quality attributes are ultimately implemented in code, and code is ground truth for which qualities are implemented.
• Smart constraints improve the scale, confidence, and automation of quality attribute analysis of software.
• Automation is key to reducing the cost and increasing confidence that programs verifiably meet quality requirements.
PACC Value Proposition

There is **economic value in objective evidence** that code implementations satisfy quality attribute requirements

- This value can be realized in several ways, including: reduced quality assurance cost, reduced acquisition risk, optimized system designs.
- Evidence becomes stronger, and yields greater value, as it moves nearer to code implementation.

Independent of economic value propositions, there is a **pressing need** to establish a **sound basis for trusted software**

- DoD and US Industry increasingly depends on code with unknown provenance, from the global market, and from open source code.

The **cost of quality and its evidence** can be substantially **reduced through automation**

- Automation magnifies the impact of analysis theory: how much economic value has accrued from Java's strong type system?
Working the Architecture/Code Seam

PACC will enable organizations to
• develop classes of systems
• that predictably satisfy
• with objective evidence
• quality attribute requirements

PACC links architecture and code.
• using construction rules that
• are automatically enforced
• and are informed by theories
• that are sound and effective
• and provide confirmable results
PACC and Component Technology

Architect
Quality by design
- component and connector view
- analyzable design patterns

Prediction Enabled Component Technology

Quality by construction
- strict code abstractions
- automated checking

Developer

PACC 2002-2007 exploited strong assumptions about programming models based on "components"

Beginning in 2008 we begin relaxing these assumptions to reach a broader audience
Today: Verifiable and Predictable Code

Component model + Software Architecture is effective, but:

- Assuming use of component technology is a strong a priori constraint on PACC customers.

Relaxing the assumption of a strong component model in code:

- continues emphasis on programming technology
- preserves emphasis on objective evidence of code behavior
- opens new avenues for PACC to deal with legacy software
Producing Trusted Binaries

Problem:
- Obtaining justifiable trust in the behavior of code without having to trust code suppliers or their development processes.

Approach:
- Automated program verification with software model checking

Foundations
- Automated predicate abstraction, abstraction refinement, SAT solvers, L*, ...
- Proof-carrying code and certifying model checking

Contributions
- Carry proof from design to executable
- Reduce trusted computing base: model checker, code generator, and compiler
PACC
Transition Products

KEY:
- Ongoing
- Early Stages
- Not Yet Started

- Transition Products
  - Web Site
  - Clinic
  - Handbook
  - Tutorials and Course
  - Case Studies
  - Workshops and Conferences
  - Papers, Technical Reports, and Presentations
  - Documentation, Methods and Practices
  - Adaptation Tools & Guidelines
  - Starter Kit

Software Engineering Institute | Carnegie Mellon
Product Line Systems Program
Linda Northrop
© 2008 Carnegie Mellon University
Transition: PACC Starter Kit and Testbed

Problem

• Getting the power of PACC into the hands of practicing software developers.

PSK provides real and compelling examples:

• Robot Command and Control
• Multichannel Audio Mixing

PSK demonstrates integrated concepts:

• Prediction with objective evidence (real time analysis, model checking)
• Pin component technology, real-time threads, event queues, statecharts, code generation, reasoning frameworks, Eclipse

Reduces learning curve

• Hands-on Tutorial at 2008 International Conference on Software Engineering
• PACC Educators’ Workshop, Summer 08

Examples from PACC Testbed & Starter Kit
Industrial Cases from 2002-2006

**Industrial Robotics**

1. Safe 3\(^{rd}\)-party software extension of hard real-time robot controller

2. Using model checking to find deep bugs in robot controller middleware

**Substation Automation**

3. Component-level assembly of substation controllers with predictable performance

4. Evolving a legacy substation controller to safe, predictable 3\(^{rd}\) party configuration
Recent Results: Predictable Assembly in Legacy Grid Code

Problem
- Highly configurable controller to support safe 3rd party configuration
- Legacy code not designed for predictability
- Millions of lines of code

Results
- Used the PACC Starter Kit to predict latency for specific configurations
- Identified architecture and coding changes to improve predictability
- Showed how performance models can be extracted from code
- Developed confidence intervals to provide evidence of system performance

<table>
<thead>
<tr>
<th>Thread</th>
<th>Pop (ρ)</th>
<th>Con (γ)</th>
<th>Ub (MRE)</th>
<th>Con (γ)</th>
<th>Ub (MRE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tApp1</td>
<td>99%</td>
<td>95</td>
<td>8.75%</td>
<td>99%</td>
<td>9.42%</td>
</tr>
<tr>
<td>tApp2</td>
<td>99%</td>
<td>95</td>
<td>6.03%</td>
<td>99%</td>
<td>6.44%</td>
</tr>
</tbody>
</table>

Confidence Intervals
New Challenges

Challenge: Developing practical and theoretical foundations for predicting the behavior of legacy software and potential modifications.

Our Research: Creating and applying program analysis technologies and best practices (e.g., static analysis, model checking, and run-time monitoring) to existing code.

Challenge: Developing an objective basis for trusting code developed by third-parties.

Our Research: Linking the theories needed to reason about program behavior to implementation constructs and providing objective evidence of program behavior (e.g., statistical measures or proof objects).

Challenge: Developing scalable technologies for reasoning about program behavior.

Our Research: Identifying "smart constraints" that simplify automated analyses and developing a technique for integrating static analysis and model checking technologies.
Today’s Presentation

Software Architecture
   *(Software Architecture Technology Initiative)*
Predictable Software Construction
   *(Predictable Assembly from Certifiable Code Initiative)*
Software Product Lines
   *(Product Line Practice Initiative)*
Ultra-Large-Scale Systems
Few Systems Are Unique

Most organizations produce families of similar systems, differentiated by features.

A reuse strategy makes sense.

Traditional reuse strategies have had little economic benefit.
A Proven Solution

SOFTWARE PRODUCT LINES
Software Product Lines Value Proposition

The systematic use of software product line practices results in significant organizational benefits including

- increased quality
  - by as much as 10x
- decreased cost
  - by as much as 60%
- decreased labor needs
  - by as much as 87%
- decreased time to market (to field, to launch...)
  - by as much as 98%
- ability to move into new markets
  - in months, not years
What Is A Software Product Line?

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

Product lines
• take economic advantage of commonality
• bound variation
How Do Product Lines Help?

Product lines amortize the investment in these and other core assets:

• requirements and requirements analysis
• domain model
• software architecture and design
• performance engineering
• documentation
• test plans, test cases, and test data
• people: their knowledge and skills
• processes, methods, and tools
• budgets, schedules, and work plans
• components and services

PRODUCT LINES = STRATEGIC REUSE
Widespread Application - 1

akvasmart
Feed control and farm management software

BOEING
Bold Stroke Avionics

E-COM Technology Ltd.
Medical imaging workstations

HP
Firmware for computer peripherals

Lucent Technologies
5ESS telecommunications switch

ABB
Asea Brown Boveri
Gas turbines, train control, semantic graphics framework

Dialect
Internet payment gateway infrastructure products

ERICSSON
AXE family of telecommunications switches

LG
Elevator control systems

NOKIA
Mobile phones, mobile browsers, telecom products for public, private and cellular networks

AXIS
Computer printer servers, storage servers, network camera and scanner servers

DNV
Customized solutions for transportation industries

GM
Software for engines, transmissions and controllers

LSI LOGIC
RAID controller firmware for disk storage units

NASA
Interferometer product line
Widespread Application - 2

**PHILIPS**
High-end televisions,
PKI telecommunications switching system,
diagnostic imaging equipment

**RICOH**
Office appliances

**SALION**
Revenue acquisition management systems

**BOSCH**
Automotive gasoline systems

**SIEMENS**
Software for viewing and quantifying radiological images

**TELVENT**
Industrial supervisory control and business process management systems

**SYMBIAN**
EPOC operating system

**NAVSEA**
Test range facilities

**U.S. ARMY**
Command and control simulator for Army fire support

**MOTOROLA**
Pagers product line
SEI Product Line Practice (PLP) Initiative’s Focus

Capitalize on the commonality across scores of families of similar systems to predictably and efficiently achieve business and mission goals by using effective software product line practices.

“Axioms” Guiding Our Work

• A product line approach is a proven strategy for exploiting the commonality among similar systems and achieving significant cost, schedule, agility, and quality benefits.
• Software product lines require specific technical and management practices.
• An architecture-centric approach is pivotal to software product lines.
The SEI Framework For Software Product Line Practice℠

The SEI Framework for Software Product Line Practice is a conceptual framework that describes the essential activities and twenty-nine practice areas necessary for successful software product lines. The Framework, originally conceived in 1998, is evolving based on the experience and information provided by the community.

Version 4.0 – in *Software Product Lines: Practices and Patterns*

Version 5.0 – [www.sei.cmu.edu/productlines/framework.html](http://www.sei.cmu.edu/productlines/framework.html)
# PRACTICE AREAS

<table>
<thead>
<tr>
<th>Software Engineering</th>
<th>Technical Management</th>
<th>Organizational Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture Definition</td>
<td>Configuration Management</td>
<td>Building a Business Case</td>
</tr>
<tr>
<td>Architecture Evaluation</td>
<td>Make/Buy/Mine/Commission Analysis</td>
<td>Customer Interface Management</td>
</tr>
<tr>
<td>Component Development</td>
<td>Measurement and Tracking</td>
<td>Developing an Acquisition Strategy</td>
</tr>
<tr>
<td>Mining Existing Assets</td>
<td>Process Discipline</td>
<td>Funding</td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>Scoping</td>
<td>Launching and Institutionalizing</td>
</tr>
<tr>
<td>Software System Integration</td>
<td>Technical Planning</td>
<td>Market Analysis</td>
</tr>
<tr>
<td>Testing</td>
<td>Technical Risk Management</td>
<td>Operations</td>
</tr>
<tr>
<td>Understanding Relevant Domains</td>
<td>Tool Support</td>
<td>Organizational Planning</td>
</tr>
<tr>
<td>Using Externally Available Software</td>
<td>Key: New Name and Substantial Change</td>
<td>Organizational Risk Management</td>
</tr>
<tr>
<td></td>
<td>Substantial Change</td>
<td>Structuring the Organization</td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>Technology Forecasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Training</td>
</tr>
</tbody>
</table>
PLP: Transition

Enable others
- Certificate Programs
- Course licensing
- PLTP Leader Certification

Foster Widespread Awareness
- Books
- Reports, articles, papers
- Five-course curriculum
- Executive seminar
- Conferences
- Workshops
- Website

Ensure practicability
- Methods
- Patterns
- Case studies
- Adoption Roadmap
- Acquisition Companion

Assist others
- Product Line Technical Probe
- Product Line Quick Look
- Practice-specific workshops
- Planning workshops

Transition Products

Transition

PLP: Transition

Enable others
- Certificate Programs
- Course licensing
- PLTP Leader Certification

Foster Widespread Awareness
- Books
- Reports, articles, papers
- Five-course curriculum
- Executive seminar
- Conferences
- Workshops
- Website

Ensure practicability
- Methods
- Patterns
- Case studies
- Adoption Roadmap
- Acquisition Companion

Assist others
- Product Line Technical Probe
- Product Line Quick Look
- Practice-specific workshops
- Planning workshops

Transition Products
## Certificate Program Course Matrix

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Software Product Line Professional</th>
<th>PLTP Team Member</th>
<th>PLTP Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Product Lines</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Adopting Software Product Lines</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Developing Software Product Lines</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PLTP Team Training</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PLTP Leader Training</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PLTP Lead Observation</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Summary of SEI Contributions

Models and Guidance
- A Framework for Software Product Line PracticeSM
- Software Product Line Acquisition: A Companion to A Framework for Software Product Line Practice
- Product line practice patterns
- Product line adoption roadmap
- Pedagogical product line

Methods and Technology
- product line analysis
- architecture definition, documentation, evaluation (ATAM®), and recovery
- mining assets
- production planning
- Structured Intuitive Model for Product Line Economics (SIMPLE)
- Product Line Technical ProbeSM (PLTPSM)
- Product Line Quick Look (PLQL)
- Interactive workshops in product line measurement, variability management, product line management
- Prediction-enabled component technology

Book
Software Product Lines: Practices and Patterns

Curriculum and Certificate Programs
- Five courses and three certificate programs
- Product Line Executive Seminar

Conferences and Workshops
- SPLC 1, SPLC2, SPLC 2004; SPLC 2006; Workshops 1997 - 2005; Army Product Line Workshop 2007

Technical Reports, publications, and Web site
New Challenges

*Challenge:* Automating all or part of the product line production process.

*Our Research:*
- Use of aspect-oriented programming to support product lines
- Product line production, including automated derivation

*Challenge:* Combining a software product line approach with new technologies and contexts
- System of systems
- Service-oriented architectures
- Open source
- Globalization
- Predictable assembly
- Ultra-large scale systems

*Our Research:* adapting software product line concepts to exploit new technologies and serve new contexts
The Total Picture

Business and Mission Goals

System (Software) Strategies

Process Improvement

Improved Architecture Practices

Software Product Lines

Improved Construction Practices

Process and product quality

process quality

process quality
Today’s Presentation

Software Architecture
   (*Software Architecture Technology Initiative*)
Predictable Software Construction
   (*Predictable Assembly from Certifiable Code Initiative*)
Software Product Lines
   (*Product Line Practice Initiative*)
Ultra-Large-Scale Systems
Trend Toward Increasing Scale-1

- Enormous web service and computing infrastructure
- Supply chain systems
- Software-based engineering systems
Trend Toward Increasing Scale-2

Healthcare Infrastructure
Trend Toward Increasing Scale-3

Homeland Security
Trend Toward Increasing Scale-4

Networked Automobiles
Trend Toward Increasing Scale-5

Saving the Environment
Increasing Scale In Military Systems

Increasingly Complex Systems
- ultra-large, network-centric, real-time, cyber-physical-social systems
  - thousands of platforms, sensors, decision nodes, weapons, and warfighters
  - connected through heterogeneous wired and wireless networks

**Goal:** *Information Superiority*
- **Transient and enduring resource constraints and failures**
- **Continuous adaptation**
  - changes in mission requirements
  - changes in operating environments
  - changes in force structure
  - perpetual systems’ evolution
  - addition of new systems
- **Sustainable - legally, technically, politically**
Ultra-Large-Scale (ULS) Systems Study

Gather leading experts to study:

- characteristics of ULS systems
- challenges and breakthroughs required
- promising research and approaches

Intended outcomes:

- ULS System Research Agenda
- program proposal
- collaborative research network

About the Effort

Funded by the Army (ASA ALT)

Staffing: 9 member SEI team
13 member expert panel

Duration: one year (04/05 -- 05/06)
Acknowledgements

Executive Summary

Part I

1. Introduction
2. Characteristics of ULS Systems
3. Challenges
4. Overview of Research Areas
5. Summary and Recommendations

Part 2

6. Detailed Description of Research Areas
   • Glossary

http://www.sei.cmu.edu/uls/
How is This Study Different

It presents an overall research agenda -- not just for new tools or a new software method or modest improvements in today’s approaches.
It is based on the challenges associated with ultra-large scale.
It focuses on the future.
It involves an interdisciplinary base.
It takes a fresh perspective on the development, deployment, operation, and evolution of software-intensive systems.

Germs of these ideas are present today in small research pockets; these efforts are currently too small to have much impact on next-generation ULS systems.
Study Conclusions

There are fundamental gaps in our current understanding of software development at the scale of ULS systems.

These gaps

• present profound impediments to the technically and economically effective achievement of the DoD goals* based on information superiority and to effective solutions for many of society’s vexing problems

• require a broad, fresh perspective and interdisciplinary, breakthrough research

We recommend

• a ULS Systems Research Agenda that includes research areas based on a fresh perspective aimed at challenges arising from increasing scale

* As stated in the Quadrennial Defense Review (QDR) Report, Feb 2006
Activity Since Report Published in May 06

There is growing community interest and research starts.
Since July 06
- more than 95,000 downloads of the report
- more than 3,000 copies of the report distributed
- more than 14 keynotes and more than 25 presentations by author team
- three press and one industry analyst interviews
- research workshops at OOPSLA 2006 and ICSE 2007
- NSF center established

New SEI Research Activities
- Mechanism design
- Implications of ULS on software architecture

Roadmap Exercise funded by Army organization (CERDEC)
New book from a non-military perspective is underway.

SMART Event
Upcoming event:
- ICSE 2008 Workshop

http://www.sei.cmu.edu/uls
Redesign of SEI ULS Systems Website

New site features include:

- Podcasts and video presentations
- ULS Systems news & events
  - RSS feed
- ULS Systems library
- Online Glossary

http://www.sei.cmu.edu/uls/
What We Learned That We Want to Share

- There is an unstoppable trend toward increasing scale in many systems important to our society.
- Scale changes everything.
- Manifestations of scale and its attendant complexity arise in many disciplines, and can be understood as a phenomenon in its own right.
- New, interdisciplinary perspective and new research in building ultra-large-scale systems is long overdue.
Working With The Product Line Systems Program

In the areas of software architecture, predictable construction, and/or software product lines, we can

- Help you solve specific problems
- Help you launch initiatives
- Help you improve your capabilities
- Conduct applied research that meets your needs
- Partner with you to create leading edge techniques, methods, and tools

In the area of ultra-large-scale systems, we would welcome your involvement as sponsors and/or as research partners.
For More Information

Linda Northrop
Director
Product Line Systems Program
Telephone: 412-268-7638
Email: lmn@sei.cmu.edu

World Wide Web:
http://www.sei.cmu.edu/architecture
http://www.sei.cmu.edu/pacc
http://www.sei.cmu.edu/productlines
http://www.sei.cmu.edu/uls/

U.S. Mail:
Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-3890

SEI Fax:
412-268-5758