Software Product Lines: Reuse That Makes Business Sense

Linda Northrop

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-2612
Software Engineering Institute (SEI)

Department of Defense R&D Laboratory (FFRDC)

Created in 1984

Under contract to Carnegie Mellon University

Offices in Pittsburgh, PA; Washington, DC; and Frankfurt, Germany

SEI Mission: advance software and related disciplines to ensure the development and operation of systems with predictable and improved cost, schedule, and quality.
SEI Technical Program

Networked Systems Survivability
- Survivable Systems Engineering
- Survivable Enterprise Management
- CERT Coordination Center
- Network Situational Awareness
- Practices Development and Training

Product Line Systems
- Product Line Practice
- Software Architecture Technology
- Predictable Assembly from Certifiable Components

Dynamic Systems
- Integration of Software-Intensive Systems
- Performance-Critical Systems

Software Engineering Process Management
- Capability Maturity Model Integration
- Team Software Process
- Software Engineering Measurement and Analysis

Acquisition Support

New Research
- Independent R&D
- International Process Research Consortium
- Software Engineering for Computational Science and Engineering
- Ultra-Large-Scale Systems
- Mission Success in Complex Environments
Software Product Lines: Reuse That Makes Business Sense

Linda Northrop
© 2007 Carnegie Mellon University

SEI Technical Program

Networked Systems Survivability
- Survivable Systems Engineering
- Survivable Enterprise Management
- CERT Coordination Center
- Network Situational Awareness
- Practices Development and Training

Product Line Systems
- Product Line Practice
- Software Architecture Technology
- Predictable Assembly from Certifiable Components

Dynamic Systems
- Integration of Software-Intensive Systems
- Performance-Critical Systems

Software Engineering Process Management
- Capability Maturity Model Integration
- Team Software Process
- Software Engineering Measurement and Analysis

Acquisition Support

New Research
- Independent R&D
- International Process Research Consortium
- Software Engineering for Computational Science and Engineering
- Ultra-Large-Scale Systems
- Mission Success in Complex Environments
Mission of the SEI Product Line Systems Program

The Product Line Systems (PLS) Program

- creates, matures, applies, and transitions technology and practices
- to effect widespread product line practice, architecture-centric development and evolution, and predictable construction
- throughout the global software community.

With regard to its software product line effort

- Make product line development and acquisition a low-risk, high-return proposition for all organizations.
Our Customers and Collaborators

ABB
Daimler Chrysler
Caterpillar
Foliage
Intuit
NCR
Northrop Grumman
Pitney Bowes
Raytheon
RIM
Robert Bosch Co.
Siemens
Unisys
Visteon
LLNL
FAA
NASA: JSC, KSC, JPL
NASA: Goddard
NRO: CCT
JNIC
DMSO
US Army: FBCB2, CECOM, ATSC, FCS, AMTS
US Army: ASA(ALT), Aviation, TAPO, BC
US Navy: Navsea, DDX, OAET, CLIP
US Air Force: F-22, JMPS, ESC

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
Business Success Requires Software Prowess

Software pervades every sector.
Software has become the bottom line for many organizations, even those who never envisioned themselves in the software business.

Cell Phone Today
~2 million lines of code

Cell Phone in 2010
~ 10 million lines of code

This year’s cars
~35 million lines of code

Cars in 2010
~ 100 million lines of code
Universal Needs

Deploy new products (services) at a rapid pace
Accommodate a growing demand for new product features across a wide spectrum of feature categories
Connect products in increasingly unprecedented ways
Exploit a rapidly changing technology base
Gain a competitive edge
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
Basic Goal

SUBSTANTIAL
QUICK
SUSTAINABLE
PROFIT
Software (System) Strategies

Process improvement
Technology innovation
Reuse
Few Systems Are Unique

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

A reuse strategy makes sense.
Reusing: An Early Topic of Discussion

“My thesis is that the software industry is weakly founded, in part because of the absence of a software components sub-industry.” [McIlroy, 1969]

“Most industry observers agree that improved software development productivity and product quality will bring an end to the software crisis. In such a world, reusable software would abound.” [Pressman, 1982]

“What is needed is the ability to create templates of program units that can be written just once and then tailored to particular needs at translation time. As we shall see, Ada provides a general and very powerful tool to do just this.” [Booch, 1986]

“If one accepts that reusability is essential to better software quality, the object-oriented approach provides a promising set of solutions.” [Meyer, 1987]

“Reusable components would be schematized and placed in a large library that would act as a clearing house for reusable software, and royalties would be paid for use of reusable components.” [Lubars, 1988]
Reuse History

Focus was small-grained, opportunistic, and technology-driven. Results did not meet business goals.
Focus was small-grained, opportunistic, and technology-driven. Results did not meet business goals.
The Real Truth About Reuse

Reuse means taking something developed for one system and using it in another.

“The XYZ System is built with 80% reuse.”

A statement like this is vacuous.

• It is not clear what is being reused.
• It is not clear that the “reuse” has any benefit.

Reusing code or components without an architecture focus and without pre-planning results in

• short-term perceived win
• long-term costs and problems
• failure to meet business goals
Software Reuse Fact And Fiction

The Fiction:

And then we’ll be able to construct software systems by picking out parts and plugging them together, just like Legos…

The Fact:

It’s more like having a bathtub full of Tinkertoys, Legos, Erector Set parts, Lincoln Logs, Block City, and six other incompatible kits -- picking out parts that fit specific functions and expecting them to fit together.
Imagine Strategic Reuse

BUSINESS STRATEGY

TECHNICAL STRATEGY

STRATEGIC REUSE
CelsiusTech: Ship System 2000

A family of 55 ship systems

- Need for developers dropped from 210 to roughly 30.
- Time to field decreased from about 9 years to about 3 years.
- Integration test of 1-1.5 million SLOC requires 1-2 people.
- Rehosting to a new platform/OS takes 3 months.
- Cost and schedule targets are predictably met.
Cummins Inc.: Diesel Control Systems

Over 20 product groups with over 1,000 separate engine applications

- Product cycle time was slashed from 250 person-months to a few person-months.
- Build and integration time was reduced from one year to one week.
- Quality goals are exceeded.
- Customer satisfaction is high.
- Product schedules are met.
National Reconnaissance Office/ Raytheon: Control Channel Toolkit

Ground-based spacecraft command and control systems

- First system had 10 times fewer defects than usual.
- The incremental build time was reduced from months to weeks.
- The system development time and costs decreased by 50%.
- There was decreased product risk.
Market Maker GMBH: Merger

Internet-based stock market software

- Each product is “uniquely” configured.
- Putting up a customized system takes three days.
Nokia Mobile Phones

Product lines with 25-30 new products per year versus 5 per year originally.

Across products there are

- varying number of keys
- varying display sizes
- varying sets of features
- 58 languages supported
- 130 countries served
- multiple protocols
- needs for backwards compatibility
- configurable features
- needs for product behavior
- change after release
How Did They Do It?
Reuse History: From Ad Hoc To Systematic
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

BUSINESS GOALS/ APPLICATION DOMAIN
is satisfied by
ARCHITECTURE
used to structure
COMPONENTS and SERVICES

pertain to

share an

are built from

PRODUCTS

CORE ASSETS
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
The Key Concepts

Use of a core asset base in production of a related set of products
The Key Concepts

Use of a core asset base in production of a related set of products

- Architecture
- Production Plan
- Scope Definition

Business Case
Software Product Lines Are Not

Fortuitous small-grained reuse
- reuse libraries containing algorithms, modules, objects, or components

Single-system development with reuse
- modifying code as necessary for the single system only

Just component-based or service-based development
- selecting components or services from an in-house library, the marketplace, or the Web with no architecture focus

Just versions of a single product
- rather, simultaneous release and support of multiple products

Just a configurable architecture
- a good start, but only part of the reuse potential

Just a set of technical standards
- constraining choices without an architecture-based reuse strategy
Software Product Lines Are

Software product lines involve strategic, planned reuse that yields predictable results.
Widespread Use of Software Product Lines

Successful software product lines have been built for families of among other things

- mobile phones
- command and control ship systems
- ground-based spacecraft systems
- avionics systems
- command and control/situation awareness systems
- pagers
- engine control systems
- mass storage devices
- billing systems
- web-based retail systems
- printers
- consumer electronic products
- acquisition management enterprise systems
- financial and tax systems
- medical devices
- farm manager software
Specific Examples - 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 Communications
  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
Specific Examples - 2

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

**Rockwell Collins**
Commercial flight control system avionics, Common Army Avionics System (CAAS), U.S. Army helicopters

**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**
Warfare Centers

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

**testo**
Climate and flue gas measurement devices

**alltel**
Support software

**FIDELITY NATIONAL FINANCIAL**
Pagers product line

Software Product Lines: Reuse That Makes Business Sense
Linda Northrop
© 2007 Carnegie Mellon University
Real World Motivation

Organizations use product line practices to:

- achieve large scale productivity gains
- improve time to market
- maintain market presence
- sustain unprecedented growth
- achieve greater market agility
- compensate for an inability to hire
- enable mass customization
- get control of diverse product configurations
- improve product quality
- increase customer satisfaction
- increase predictability of cost, schedule, and quality
Example Organizational Benefits

Improved productivity
- by as much as 10x

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
## Costs Of A Software Product Line

<table>
<thead>
<tr>
<th>Core Assets</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Must support variation inherent in the product line</td>
</tr>
<tr>
<td>Software Components</td>
<td>Must be designed to be general without a loss of performance; must build in support for variation points</td>
</tr>
<tr>
<td>Test Plans, Test Cases, Test Data</td>
<td>Must consider variation points and multiple instances of the product line</td>
</tr>
<tr>
<td>Business Case and Market Analysis</td>
<td>Must address a family of software products, not just one product</td>
</tr>
<tr>
<td>Project Plans</td>
<td>Must be generic or be made extensible to accommodate product variations</td>
</tr>
<tr>
<td>Tools and Processes</td>
<td>Must be more robust</td>
</tr>
<tr>
<td>People, Skills, Training</td>
<td>Must involve training and expertise centered around the assets and procedures associated with the product line</td>
</tr>
</tbody>
</table>
Economics Of Product Lines

*Software Product-Line Engineering: A Family-Based Software Development Process*
Economics Of Product Lines

Numbers of Products

Cumulative Costs

Current Practice

With Product Line Approach

PAYOFF POINT


Software Product-Line Engineering: A Family-Based Software Development Process

Necessary Changes

The product line architecture is central to success.

- Business Approach
- Development Approach
- Organizational Structure and Personnel
- Management
**Why Is Software Architecture Important?**

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

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

Contexts for product lines vary widely, based on

- nature of products
- nature of market or mission
- business goals
- organizational infrastructure
- workforce distribution
- process discipline
- artifact maturity

But there are universal essential activities and practices.
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 – available in early 2007
SEI Information Sources

Case studies, experience reports, and surveys

Workshops and conferences

Applied research

Collaborations with customers on actual product lines
The Three Essential Activities

- Core Asset Development
- Product Development
- Management
Core Asset Development

Product Constraints
Production Constraints
Production Strategy
Preexisting Assets

Core Asset Development

Management

Product Line Scope
Core Asset Base
Production Plan
Attached Processes
Product Line Production Plan

Production Plan

Product Constraints
Production Constraints
Production Strategy

Production Method

+ + + +

Production Process

+ +

Project Details
Product Development

Product Description

Product Line Scope
Core Asset Base
Production Plan

Product Development

Management

Products
Feedback
New Assets
Product Constraints
Management
Management

Management at multiple levels plays a critical role in the successful product line practice by:

- achieving the right organizational structure
- allocating resources
- coordinating and supervising
- providing training
- rewarding employees appropriately
- developing and communicating an acquisition strategy
- managing external interfaces
- creating and implementing a product line adoption plan
- launching and institutionalizing the approach in a manner appropriate to the organization
Managing A Software Product Line Requires Leadership

A key role for software product line management is that of champion. A champion must

- set and maintain the vision
- ensure that the appropriate goals and measures are in place
- “sell” the product line up and down the chain
- sustain morale
- deflect potential derailments
- solicit feedback and continuously improve the approach
Each of these is essential, as is the blending of all three.
Different Approaches - 1

**Proactive:** Develop the core assets first.

- Develop the scope first and use it as a “mission” statement.
- Products come to market quickly with minimum code writing.
- Requires upfront investment and predictive knowledge

**Reactive:** Start with one or more products.

- From them, generate the product line core assets and then future products; the scope evolves more dramatically.
- Much lower cost of entry
- The architecture and other core assets must be robust, extensible, and appropriate to future product line needs.
Different Approaches - 2

**Incremental:** In either a reactive or proactive approach, it is possible to develop the core asset base in stages, while planning from the beginning to develop a product line.

- Develop part of the core asset base, including the architecture and some of the components.
- Develop one or more products.
- Develop part of the rest of the core asset base.
- Develop more products.
- Evolve more of the core asset base.
- ...
## Alternate Terminology

<table>
<thead>
<tr>
<th>Our Terminology</th>
<th>Alternate Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Line</td>
<td>Product Family</td>
</tr>
<tr>
<td>Software Core Assets</td>
<td>Platform</td>
</tr>
<tr>
<td>Business Unit</td>
<td>Product Line</td>
</tr>
<tr>
<td>Product</td>
<td>Customization</td>
</tr>
<tr>
<td>Core Asset Development</td>
<td>Domain Engineering</td>
</tr>
<tr>
<td>Product Development</td>
<td>Application Engineering</td>
</tr>
</tbody>
</table>
Driving The Essential Activities

Beneath the level of the essential activities are essential practices that fall into practice areas.

A **practice area** is a body of work or a collection of activities that an organization must master to successfully carry out the essential work of a product line.
Three Categories Of Practice Areas

- Organizational Management Practice Areas
- Technical Management Practice Areas
- Software Engineering Practice Areas

Enable and orchestrate  Manage and support
## Framework

### Essential Activities

<table>
<thead>
<tr>
<th>Practice Areas</th>
<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>
<td></td>
</tr>
<tr>
<td>Architecture Evaluation</td>
<td>Data Collection, Metrics, and Tracking</td>
<td>Customer Interface Management</td>
<td></td>
</tr>
<tr>
<td>Component Development</td>
<td>Make/Buy/Mine/Commission Analysis</td>
<td>Developing an Acquisition Strategy</td>
<td></td>
</tr>
<tr>
<td>COTS Utilization</td>
<td>Process Definition</td>
<td>Funding</td>
<td></td>
</tr>
<tr>
<td>Mining Existing Assets</td>
<td>Scoping</td>
<td>Launching and Institutionalizing</td>
<td></td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>Technical Planning</td>
<td>Market Analysis</td>
<td></td>
</tr>
<tr>
<td>Software System Integration</td>
<td>Technical Risk Management</td>
<td>Operations</td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>Tool Support</td>
<td>Organizational Planning</td>
<td></td>
</tr>
<tr>
<td>Understanding Relevant Domains</td>
<td></td>
<td>Organizational Risk Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Structuring the Organization</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technology Forecasting</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Training</td>
<td></td>
</tr>
</tbody>
</table>
## Framework Version 5.0

### ESSENTIAL ACTIVITIES

[Diagram of Core Asset Development, Product Development, and Management cycles]

### 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>Measurement and Tracking</td>
<td>Customer Interface Management</td>
</tr>
<tr>
<td>Component Development</td>
<td>Make/Buy/Mine/Commission Analysis</td>
<td>Developing an Acquisition Strategy</td>
</tr>
<tr>
<td><strong>Using Externally Available Software</strong></td>
<td><strong>Process Discipline</strong></td>
<td>Funding</td>
</tr>
<tr>
<td>Mining Existing Assets</td>
<td>Scoping</td>
<td>Launching and Institutionalizing</td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>Technical Planning</td>
<td>Market Analysis</td>
</tr>
<tr>
<td>Software System Integration</td>
<td>Technical Risk Management</td>
<td>Operations</td>
</tr>
<tr>
<td>Testing</td>
<td>Tool Support</td>
<td>Organizational Planning</td>
</tr>
<tr>
<td>Understanding Relevant Domains</td>
<td></td>
<td>Organizational Risk Management</td>
</tr>
<tr>
<td><strong>Key:</strong></td>
<td><strong>New Name and Substantial Change</strong></td>
<td>Structuring the Organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technology Forecasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Training</td>
</tr>
</tbody>
</table>
## 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></td>
<td></td>
</tr>
<tr>
<td><strong>Key:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Name and Substantial Change</td>
<td></td>
<td>Organizational Risk Management</td>
</tr>
<tr>
<td>Substantial Change</td>
<td></td>
<td>Structuring the Organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Technology Forecasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Training</td>
</tr>
</tbody>
</table>
Dilemma: How Do You Apply The 29 Practice Areas?

Organizations still have to figure out how to put the practice areas into play.

Twenty-nine is a big number.
Help To Make It Happen

PRACTICE AREAS

<table>
<thead>
<tr>
<th>Software Engineering</th>
<th>Technical Management</th>
<th>Organizational Management</th>
</tr>
</thead>
</table>

GUIDANCE

- Case Studies
- Patterns
- Probe
- Curriculum
Case Studies

**CelsiusTech** – CMU/SEI-96-TR-016
http://www.sei.cmu.edu/publications/documents/01.reports/96.tr.016.html

**Cummins, Inc.** *Software Product Lines: Practices and Patterns*

**Market Maker** *Software Product Lines: Practices and Patterns*

**NRO/Raytheon** – CMU/SEI-2001-TR-030
http://www.sei.cmu.edu/publications/documents/01.reports/02tr030.html

**NUWC** – CMU/SEI-2002-TN-018
http://www.sei.cmu.edu/publications/documents/02.reports/02tn018.html

**Salion, Inc.** – CMU/SEI-2002-TR-038
http://www.sei.cmu.edu/publications/documents/02.reports/02tr038.html

**U.S. Army** – CMU/SEI-2005-TR-019
http://www.sei.cmu.edu/publications/documents/05.reports/05tr019.html
Help To Make It Happen

ESSENTIAL ACTIVITIES

PRACTICE AREAS

- Software Engineering
- Technical Management
- Organizational Management

GUIDANCE

- Case Studies
- Patterns
- Probe
- Curriculum
Patterns Can Help

Patterns are a way of expressing common context and problem-solution pairs.

Patterns have been found to be useful in building architecture, economics, software architecture, software design, software implementation, process improvement, and others.

Patterns assist in effecting a divide and conquer approach.
Software Product Line Practice Patterns

- **Context**
  - Organizational Situation

- **Problem**
  - What part of a product line effort needs to be accomplished

- **Solution**
  - Grouping of practice areas
  - Relations among these practice areas (and/or groups if there is more than one)
What To Build Pattern - 1

Name:
The *What to Build* pattern helps an organization determine what products ought to be in its software product line – what products to build.

Context:
An organization has decided to field a software product line and knows the general product area for the set of products.

Problem:
To determine what products should be included in the product line

Solution:
Determining what to build requires information related to the product area, technology, and market; the business justification; and the process for describing the set of products to be included in the product line.
What To Build Pattern - 2

Understanding Relevant Domains

Market Analysis

Technology Forecasting

Scoping

Building a Business Case

Product Set

Domain Models

Market Climate

Technology Predictions

Product Line Scope

Product Set

Justification

Business Case

Dynamic Structure
Factory Pattern - 1

Name:
The *Factory* pattern is a composite pattern that describes the entire product line organization.

Context:
An organization is considering (or fielding) a product line.

Problem:
To map the entire product line effort

Solution:
Fielding a product line involves

- deciding what to build
- building and running the production capability
- preparing the organization
- designing and providing the product parts
- running the assembly line
- monitoring the process
Factory Pattern - 2

Dynamic Structure

- What to Build → Product Parts → Product Builder
- Process Discipline → Assembly Line

Cold Start → In Motion → Monitor

Informs and information flow

Supports
# Current Set Of Patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly Line</td>
<td></td>
</tr>
<tr>
<td>Cold Start</td>
<td>Warm Start</td>
</tr>
<tr>
<td>Curriculum</td>
<td></td>
</tr>
<tr>
<td>Each Asset</td>
<td>Each Asset Apprentice</td>
</tr>
<tr>
<td></td>
<td>Evolve Each Asset</td>
</tr>
<tr>
<td>Essentials Coverage</td>
<td></td>
</tr>
<tr>
<td>Factory</td>
<td>Adoption Factory</td>
</tr>
<tr>
<td>In Motion</td>
<td></td>
</tr>
<tr>
<td>Monitor</td>
<td></td>
</tr>
<tr>
<td>Process</td>
<td>Process Improvement</td>
</tr>
<tr>
<td>Product Parts</td>
<td>Green Field</td>
</tr>
<tr>
<td></td>
<td>Barren Field</td>
</tr>
<tr>
<td></td>
<td>Plowed Field</td>
</tr>
<tr>
<td>What to Build</td>
<td>Analysis</td>
</tr>
<tr>
<td></td>
<td>Forced March</td>
</tr>
</tbody>
</table>
Help To Make It Happen

ESSENTIAL ACTIVITIES

Core Asset Development

Product Development

Management

PRACTICE AREAS

Software Engineering | Technical Management | Organizational Management

GUIDANCE

Case Studies

Patterns

Probe

Curriculum
What Is An SEI Product Line Technical Probe (PLTP)?

The SEI PLTP is a method for examining an organization’s readiness to adopt or ability to succeed with a software product line approach.

- It is a diagnostic tool based on the SEI Framework for Software Product Line Practice.
- The 29 practice areas are the basis of data collection and analysis.
Help To Make It Happen

PRACTICE AREAS

<table>
<thead>
<tr>
<th>Software Engineering</th>
<th>Technical Management</th>
<th>Organizational Management</th>
</tr>
</thead>
</table>

GUIDANCE

- Case Studies
- Patterns
- Probe

Curriculum
# The SEI Software Product Line Curriculum

## Three Certificate Programs

<table>
<thead>
<tr>
<th>Five Courses</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>

✓: course required to receive certificate
The Product Line Adoption Endgame

To have an *operational software product line*.

To do that, an organization must

- have
  - a core asset base
  - supportive processes and organizational structures
- develop products from that asset base in a way that achieves business goals
- improve and extend the software product line effort as long as it makes sense
Barriers To Product Line Adoption

Cost, cost, and cost...
You have to invest to eventually SAVE
Barriers To Product Line Adoption
More Barriers

- Lack of knowledge
- Need for organizational change
- Cultural resistance
- Lack of sufficient management support
- Lack of necessary talent
- Incompatible development processes
- Globalization of workforce
- Stove-piped mentality
- No clear path to follow

Change management models are useful.

A product line adoption roadmap is helpful.
The SEI Adoption Factory Pattern

Phases

<table>
<thead>
<tr>
<th>Establish Context</th>
<th>Establish Production Capability</th>
<th>Operate Product Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each Asset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What to Build</td>
<td>Product Parts</td>
<td>Product Builder</td>
</tr>
<tr>
<td>Process Discipline</td>
<td>Assembly Line</td>
<td></td>
</tr>
</tbody>
</table>

Focus Areas

Product

Process

Organization

Focus Areas Informs and information flow

Supports

Cold Start

In Motion

Monitor
### Associated Practice Areas

<table>
<thead>
<tr>
<th>Establish Context</th>
<th>Establish Production Capability</th>
<th>Operate Product Line</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product</strong></td>
<td><strong>Process</strong></td>
<td><strong>Organization</strong></td>
</tr>
<tr>
<td>• Marketing Analysis</td>
<td>• Requirements Engineering</td>
<td>• Requirements Engineering</td>
</tr>
<tr>
<td>• Understanding Relevant Domains</td>
<td>• Architecture Definition</td>
<td>• Architecture Definition</td>
</tr>
<tr>
<td>• Technology Forecasting</td>
<td>• Architecture Evaluation</td>
<td>• Architecture Evaluation</td>
</tr>
<tr>
<td>• Building a Business Case</td>
<td>• Mining Existing Assets</td>
<td>• Mining Existing Assets</td>
</tr>
<tr>
<td>• Scoping</td>
<td>• Component Development</td>
<td>• Component Development</td>
</tr>
<tr>
<td></td>
<td>• Using Externally Available Software</td>
<td>• Using Externally Available Software</td>
</tr>
<tr>
<td></td>
<td>• Software System Integration</td>
<td>• Software System Integration</td>
</tr>
<tr>
<td></td>
<td>• Testing</td>
<td>• Testing</td>
</tr>
<tr>
<td>• Process Discipline</td>
<td>• Make/Buy/Mine/Commission</td>
<td>• Make/Buy/Mine/Commission</td>
</tr>
<tr>
<td></td>
<td>• Configuration Management</td>
<td>• Configuration Management</td>
</tr>
<tr>
<td></td>
<td>• Tool Support</td>
<td>• Tool Support</td>
</tr>
<tr>
<td></td>
<td>• Measurement and Tracking</td>
<td>• Measurement and Tracking</td>
</tr>
<tr>
<td></td>
<td>• Technical Planning</td>
<td>• Technical Planning</td>
</tr>
<tr>
<td></td>
<td>• Technical Risk Management</td>
<td>• Technical Risk Management</td>
</tr>
<tr>
<td>• Launching and Institutionalizing</td>
<td>• Launching and Institutionalizing</td>
<td>• Measurement and Tracking</td>
</tr>
<tr>
<td>• Funding</td>
<td>• Funding</td>
<td>• Technical Risk Management</td>
</tr>
<tr>
<td>• Structuring the Organization</td>
<td>• Structuring the Organization</td>
<td>• Organizational Risk Management</td>
</tr>
<tr>
<td>• Operations</td>
<td>• Operations</td>
<td>• Customer Interface Management</td>
</tr>
<tr>
<td>• Organizational Planning</td>
<td>• Organizational Planning</td>
<td>• Organizational Planning</td>
</tr>
<tr>
<td>• Customer Interface Management</td>
<td>• Customer Interface Management</td>
<td>• Training</td>
</tr>
<tr>
<td>• Organizational Risk Management</td>
<td>• Organizational Risk Management</td>
<td>• Developing an Acquisition Strategy</td>
</tr>
<tr>
<td>• Developing an Acquisition Strategy</td>
<td>• Developing an Acquisition Strategy</td>
<td>• Technical Planning</td>
</tr>
<tr>
<td>• Training</td>
<td>• Training</td>
<td>• Organizational Planning</td>
</tr>
</tbody>
</table>
In A Nutshell

Software product lines epitomize the concept of strategic, planned reuse. The product line concept is about more than a new technology. It is a new way of doing one’s software business.

There are essential product line activities and practices areas as well as product line patterns to make the move to product lines more manageable.

**PRACTICE AREAS**

<table>
<thead>
<tr>
<th>Software Engineering</th>
<th>Technical Management</th>
<th>Organizational Management</th>
</tr>
</thead>
</table>

**ESSENTIAL ACTIVITIES**

Core Asset Development  
Product Development  
Management
What’s Different About Reuse With Software Product Lines?

- Business dimension
- Iteration
- Architecture focus
- Preplanning
- Process and product connection
At The Heart Of Successful Product Lines

- A pressing need that addresses the heart of the business
- Long and deep domain experience
- A legacy base from which to build
- Architectural excellence
- Process discipline
- Management commitment
- Loyalty to the product line as a single entity
Summary of SEI Contributions

Models and Guidance
- A Framework for Software Product Line Practice℠
- 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 Product Line Economics (SIMPLE)
- Product Line Technical Probe℠ (PLTP℠)
- 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

Technical Reports, publications, and Web site
Final Word

If properly managed, the benefits of a product line approach far exceed the costs.

Strategic software reuse through a well-managed product line approach achieves business goals for:

- efficiency
- time to market
- productivity
- quality
- agility
Questions – Now Or Later

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

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

World Wide Web:
http://www.sei.cmu.edu/productlines
SEI Fax: 412-268-5758