Carnegie Mellon University

Software Engineering Institute

Software Product Lines

Course Introduction

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Software Product Lines

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM19-0013

Introductions

Instructor Introduction

Participant Introductions

- name
- company/position
- background
- course expectations

Course Objectives

The Software Product Lines course is a two-day course designed to

- provide an introduction to software product lines
- introduce participants to the technical and management practices needed to succeed with software product lines
- provide guidelines and patterns helpful in the practical application of product line techniques
- illustrate software product line concepts with case studies
- introduce participants to a software product line diagnostic method and an adoption roadmap
- relate software product lines to other technology and business trends

Course Entry Criteria

This course is designed for software engineers and technical managers interested in effective reuse strategies.

Participants should have

- experience in designing and developing software-intensive systems
- some familiarity with modern software engineering concepts and management practices
- an understanding of basic software architecture concepts

Course Outcomes

Participants will have a basic understanding of

- the essential activities involved in fielding a software product line
- the costs and benefits of adopting a product line approach
- the software engineering, technical management, and organizational management practices necessary for successful software product lines
- product line practice patterns that aid in product line adoption
- a product line diagnostic method and an adoption roadmap
- how a product line approach can be combined with other technology and business trends

Course Structure

Course Introduction

Part 1: Software Product Line Fundamentals

Part 2: Software Product Line Practice Areas

Part 3: Putting the Practice Areas into Action

Wrap-Up

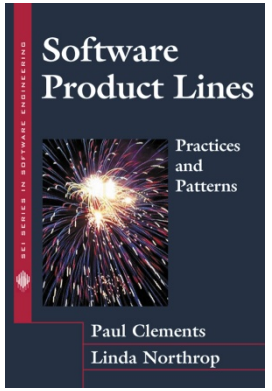
Course Strategy

Lectures and discussion will be used to introduce concepts.

Case studies will be used to illustrate concepts.

Short exercises will be used to reinforce concepts.

The course content is based on
Software Product Lines: Practices and Patterns.



Course Materials

Course notebook

- course description
- course outline
- printed slides
- exercises
- bibliography
- glossary

Software Product Lines: Practices and Patterns book

Supplemental materials

- Salion Case Study

Course Agenda: Day One

Course Introduction

Part 1: Software Product Line Fundamentals

- Basic Ideas and Terms
- Benefits of Software Product Lines
- The Three Essential Activities

Part 2: Software Product Line Practice Areas

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

Course Agenda: Day Two

Part 3: Putting the Practice Areas into Action

- Case Studies
  - Cummins, Inc.
  - Control Channel Toolkit
  - Salion, Inc.
- Software Product Line Practice Patterns
- Product Line Technical Probe
- Product Line Adoption Roadmap

Wrap-Up

Rules of Engagement

We will be very busy over the next two days.

To complete everything and get the most from the course, we will need to follow some rules of engagement:

- Your participation is essential.
- Feel free to ask questions at any time.
- Discussion is good, but we might need to cut some discussions short in the interest of time.
- Please try to limit side discussions during the lectures.
- Please turn off mobile phones, laptops, and PDAs.
- Let's try to start on time.

Course Completion Criteria

To receive a certificate for completion of this course participants must

- actively participate in classroom discussions during both days
- not miss any classroom time

Questions So Far?

Carnegie Mellon University

Software Engineering Institute

Part 1: Software Product Line Fundamentals
Session 1: Basic Ideas and Terms

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Course Structure

Course Introduction

Part 1: Software Product Line Fundamentals

Part 2: Software Product Line Practice Areas

Part 3: Putting the Practice Areas into Action

Wrap-Up

Session 1 Objectives

This session will introduce participants to

- the concept of software product lines
- the basic terms associated with software product lines

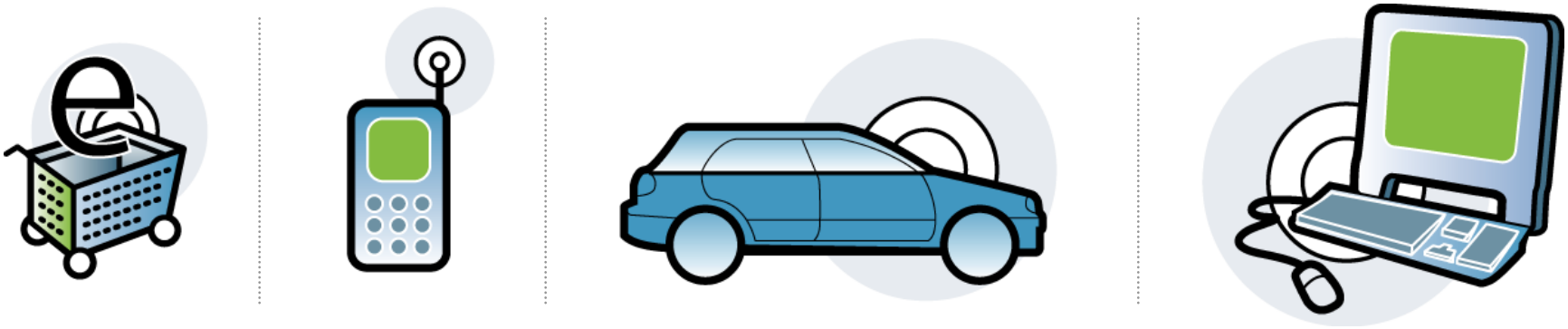
Session 1 Contents

What a Software Product Line Is

What Software Product Lines Are Not

Terminology

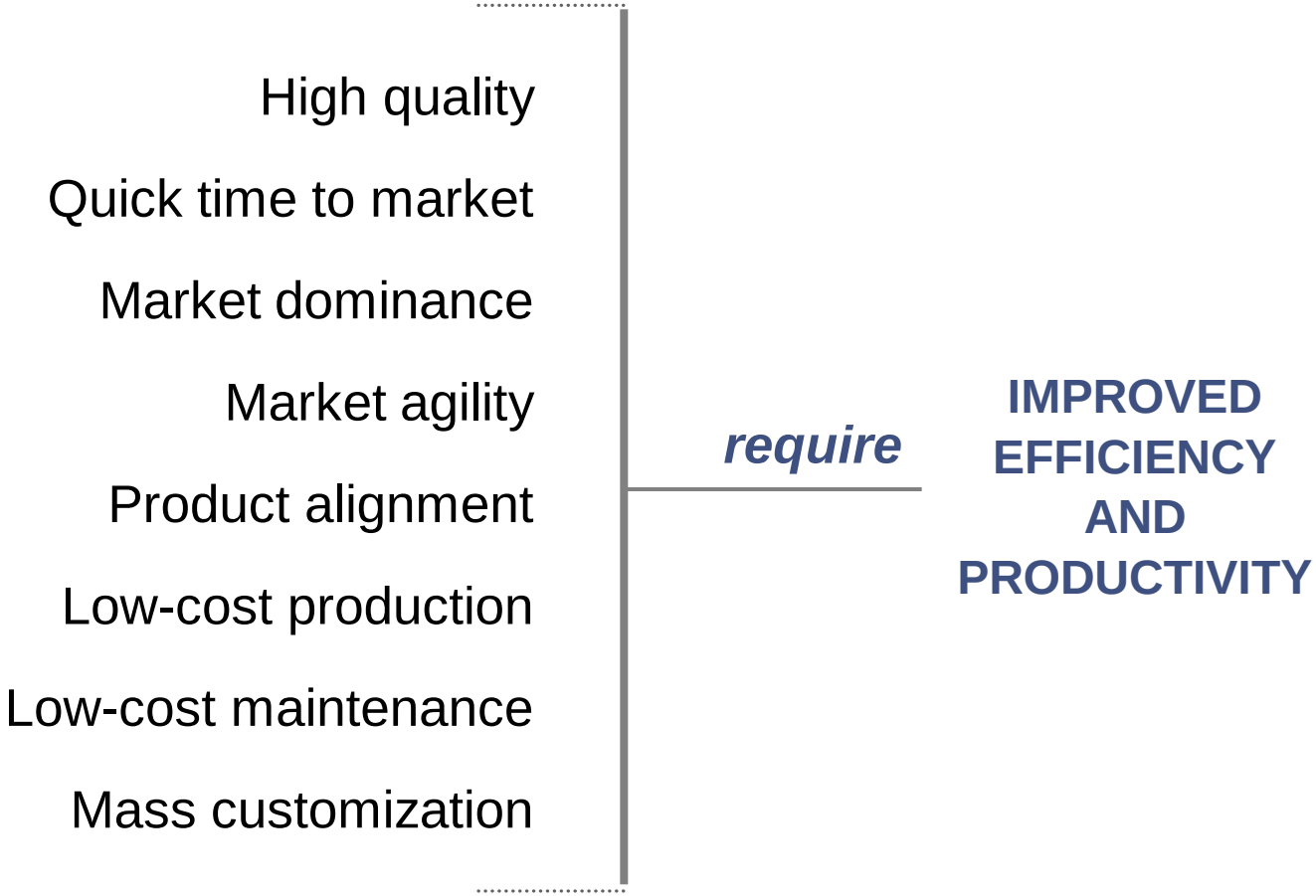
Business Success Requires Software Prowess



Software pervades every sector.

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

Universal Business Goals

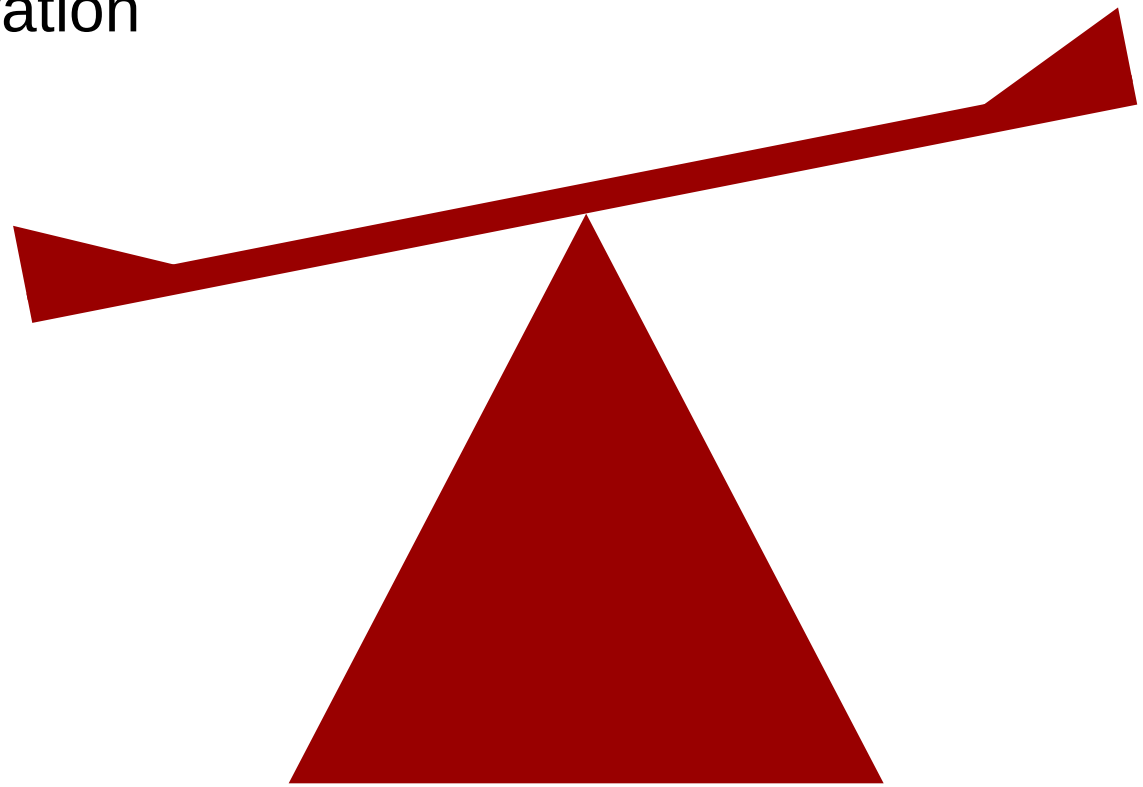


Software (System) Strategies

Process improvement

Technology innovation

Reuse



Few Systems Are Unique

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

A reuse strategy makes sense.

Reuse: Management Challenge

Over the next n years you have m similar systems under development and mildly (wildly) different development and maintenance approaches.

At the same time, you have less money to spend, fewer people to work with, and less time to get the job done.

And, oh by the way, the systems are more complex.

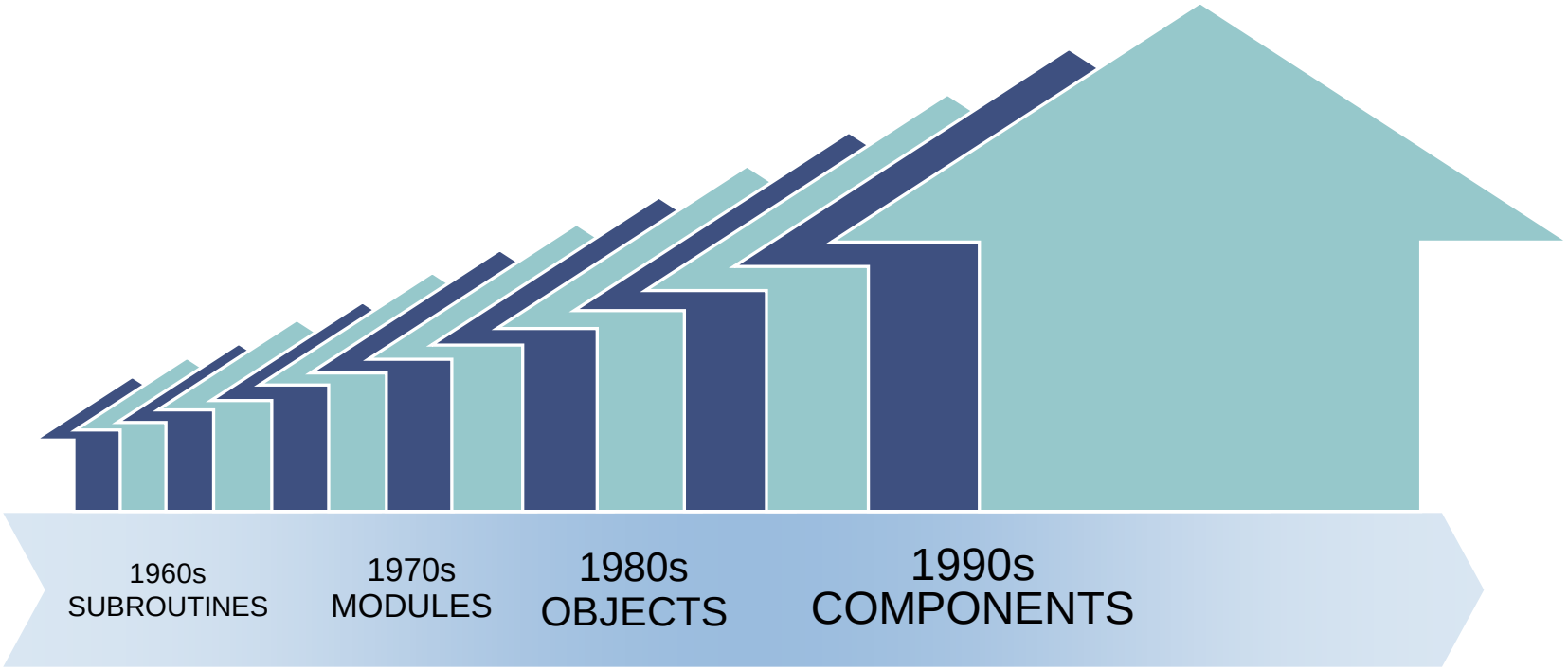
Reuse: Developer Challenge

Over the next n years you develop m similar systems using mildly (wildly) different development and maintenance approaches.

You reuse code, applications, directories, data, and so forth.

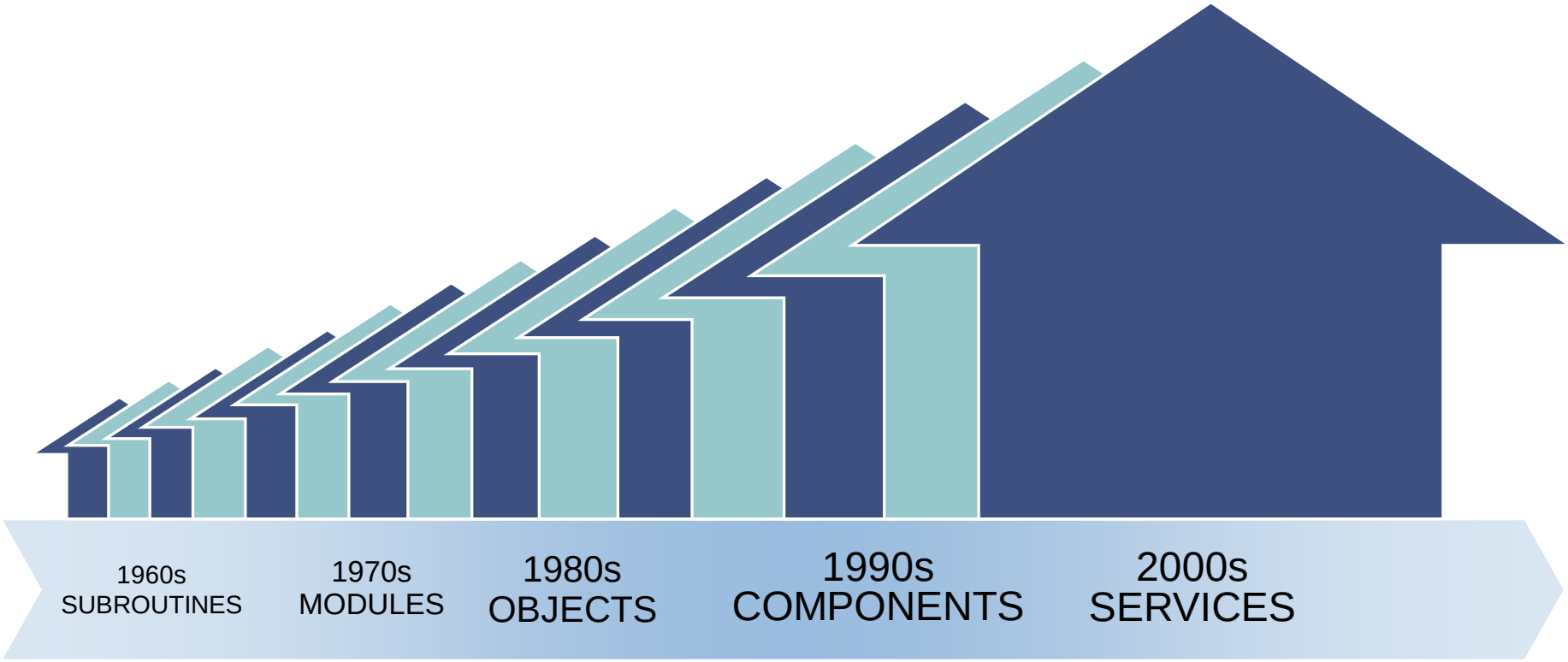
But you still have integration problems, testing nightmares, problems responding to functional volatility, problems handling multiple yet similar configurations, and continuous pressure from management to do more in less time.

Reuse History

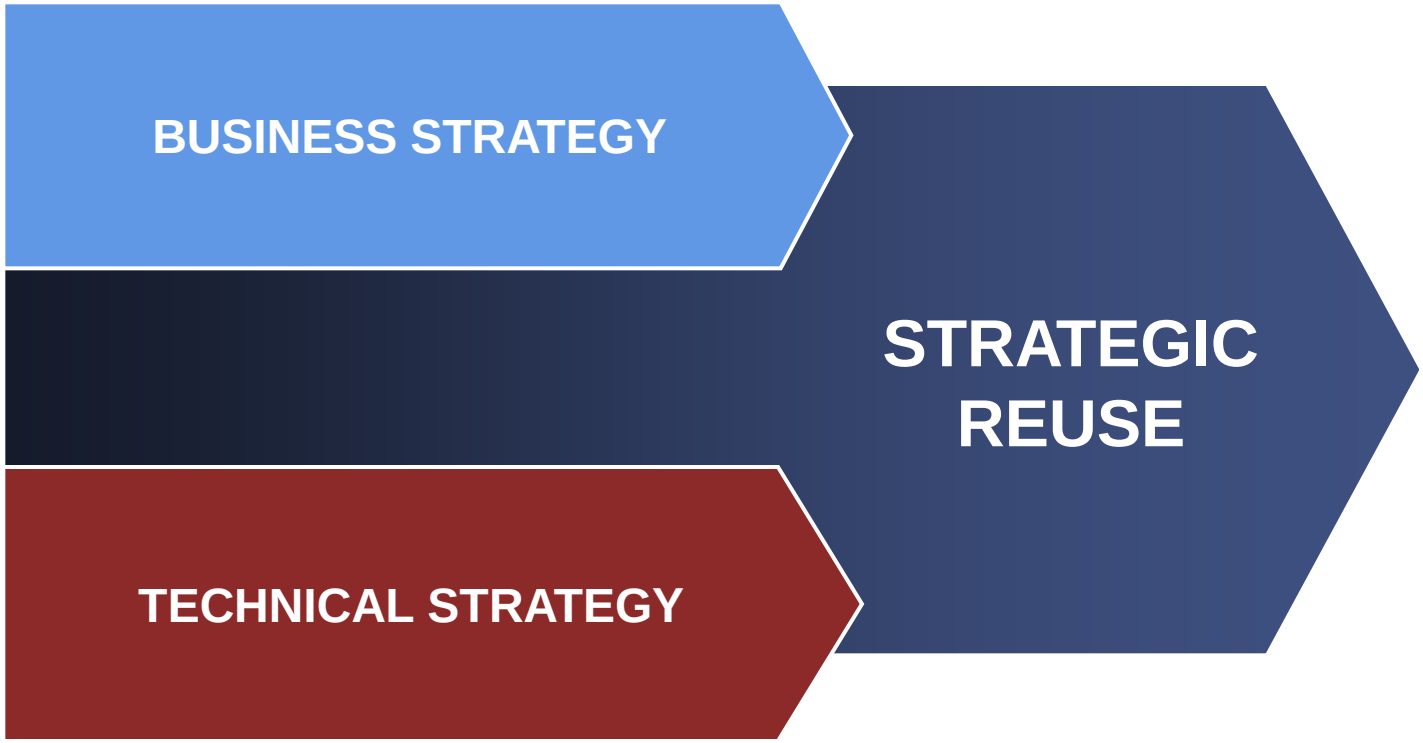


Focus was small-grained, opportunistic, and technology-driven. Results did not meet business goals.

Reuse History



Imagine 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.
- Customer satisfaction is high.

Cummins, Inc.: Diesel Engine Control Systems

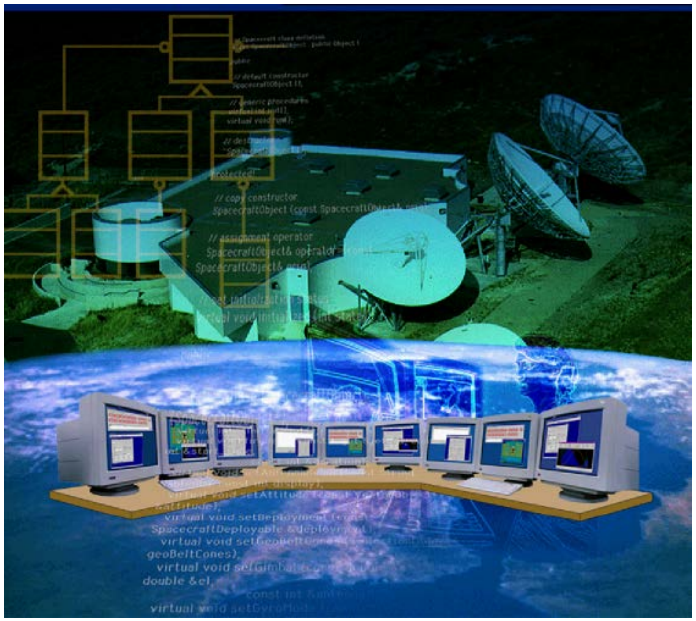
Over 20 product groups with over 1,000 separate engine applications

- 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.
- Product line approach enabled Cummins to quickly enter and then dominate the *industrial* diesel engine market.

National Reconnaissance Office/Raytheon: Control Channel Toolkit

Ground-based spacecraft command and control systems

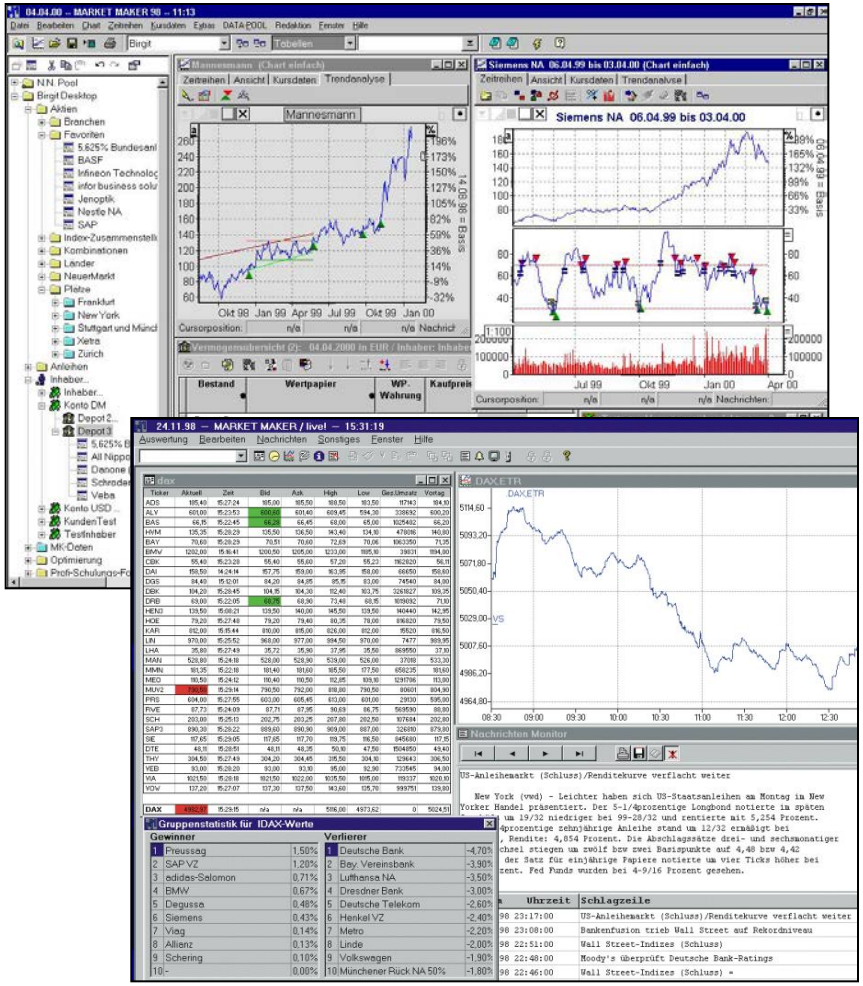
- First system had 10 times fewer defects than usual.
- The incremental build time was reduced from months to weeks.
- The system development time and costs decreased by 50%.
- There was decreased product risk.



Market Maker GMBH: Merger

Internet-based stock market software

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



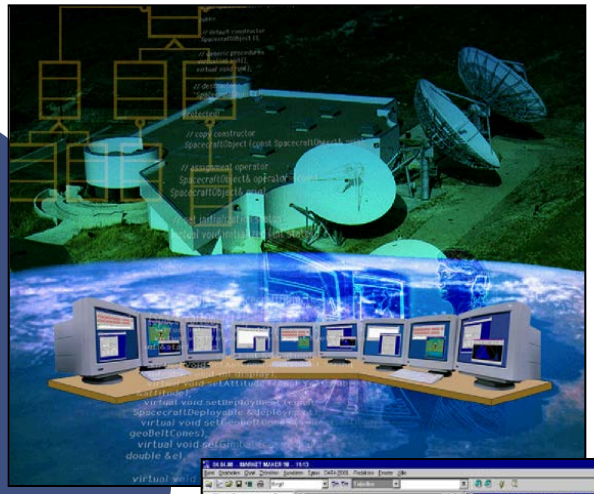
Nokia Mobile Phones

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

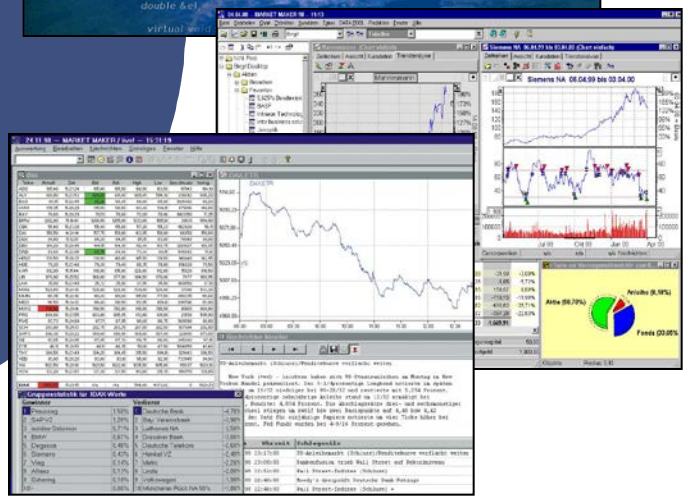
Across products there are

- varying numbers 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
- changes after product release

How Did They Do It?



SOFTWARE PRODUCT LINES



Product Lines Are a Proven Concept

Product lines—building a family of products from interchangeable parts—have existed for centuries.

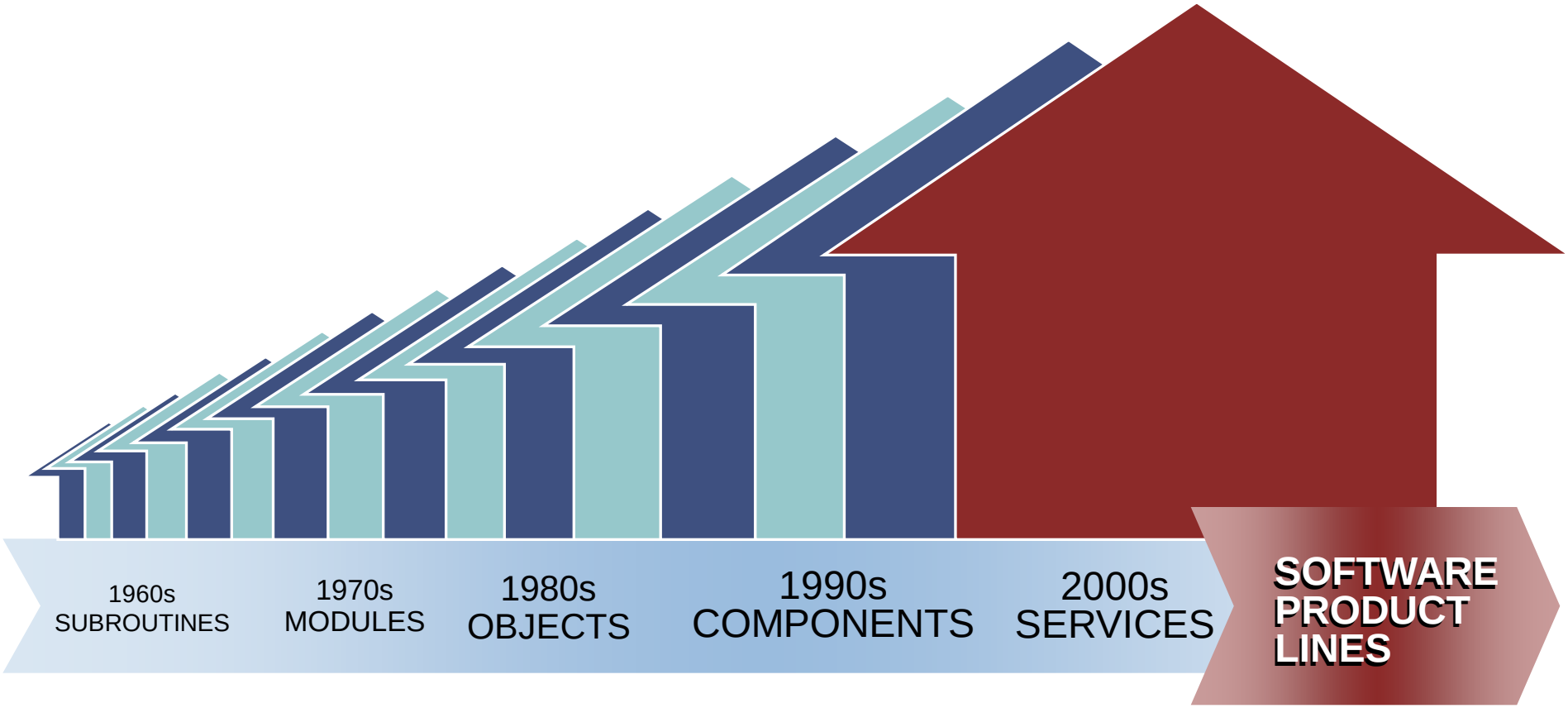
- Li Chieh, the state architect of the Chinese emperor Hui-tsung, published a set of building codes for official buildings. This book defined a set of reusable designs: a “product line” of buildings.
- IBM’s System/360 family of computers, introduced in 1964, was a product line of computers.
- The Airbus A318/319/320/321 family of commercial aircraft is a product line, as is the Boeing 757/767 family.
- McDonalds and Burger King each have a product line of restaurant menu items.





Source: www.burgerking.com

Reuse History: From Ad Hoc to Systematic

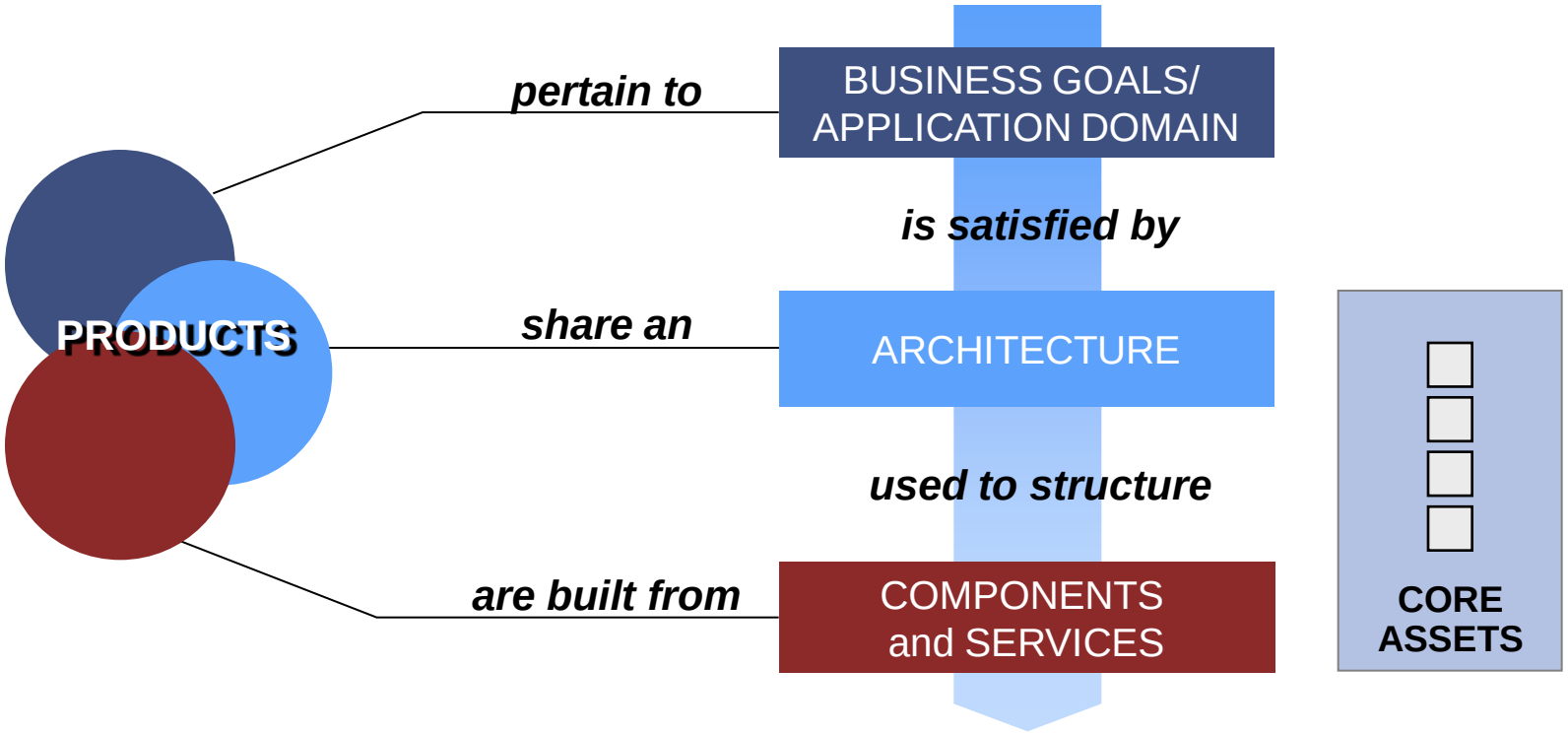


What Is a Software Product Line?

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.

- a new application of a proven concept
- an innovative, growing concept in software engineering

Software Product Lines



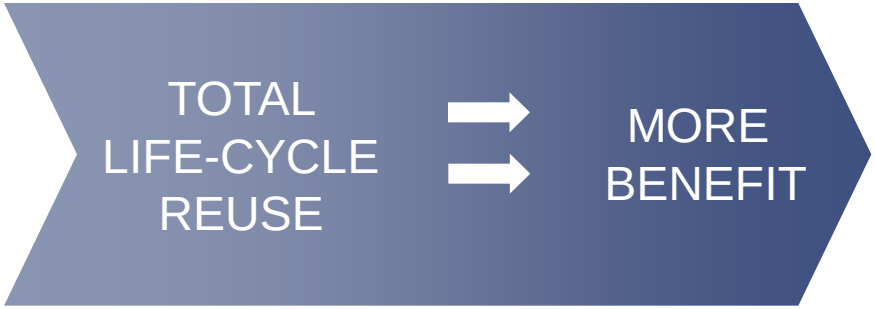
Product lines

- take economic advantage of commonality
- bound variation

How Do Product Lines Help?

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

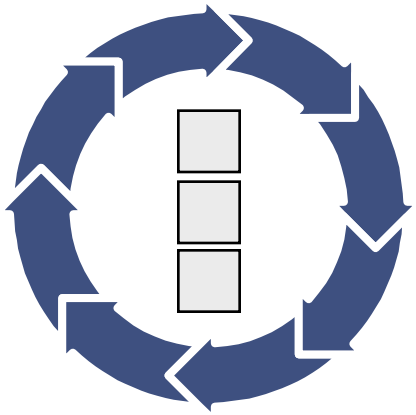
- requirements and requirements analysis
- domain model
- software architecture and design
- performance engineering
- documentation
- test plans, test cases, and test data
- people: their knowledge and skills
- processes, methods, and tools
- defect elimination
- budgets, schedules, and work plans
-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



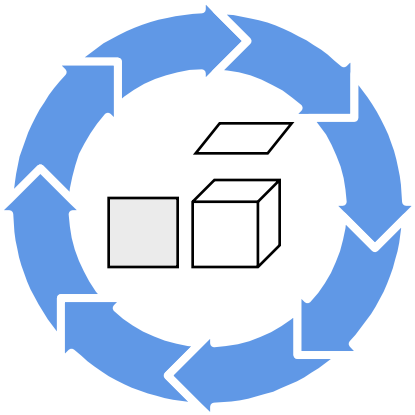
PRODUCT LINES = STRATEGIC REUSE

The Key Concepts

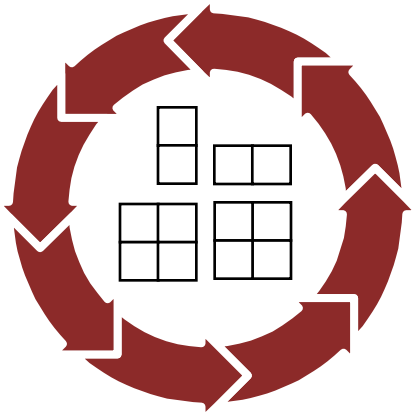
Use of a core asset base



in the production

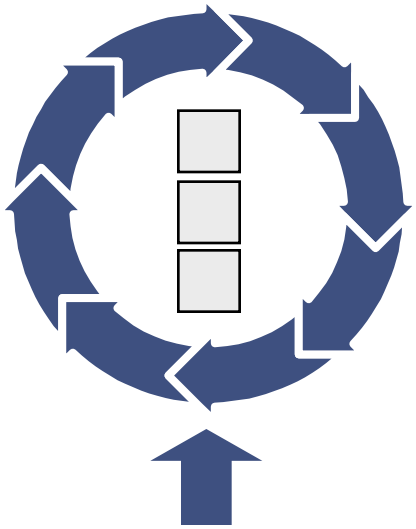


of a related set of products



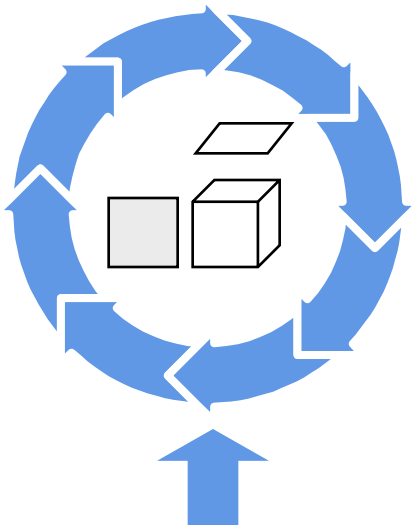
The Key Concepts

Use of a core asset base



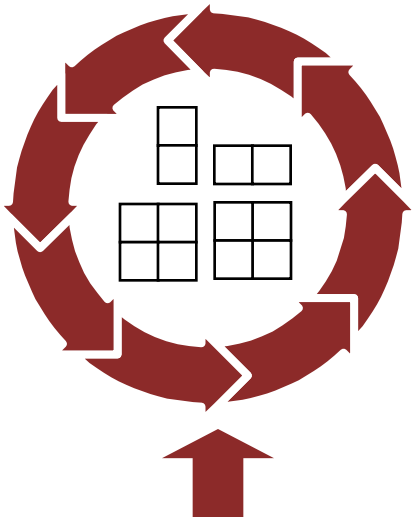
Architecture

in the production



Production Plan

of a related set of products



Scope Definition
Business Case

Session 1 Contents

Software Product Lines Are Not - 1

Fortuitous small-grained reuse

- reuse libraries containing algorithms, modules, objects, or components
- Benefits depend on
  - a software engineer's predisposition to use what is in the library
  - suitability of library contents for particular needs
  - successful adaptation and integration of library units into the rest of the system
- Reuse is not planned, enabled, or enforced, and results are not predictable.

Single-system development with reuse

- borrowing opportunistically from previous efforts—"clone and own"
- modifying code as necessary for the single system only
- core asset base never cultivated

Software Product Lines Are Not - 2

Just component-based or service-based development

- selecting components or services from an in-house library, the Web, or the marketplace
- missing a product line architecture, a production plan, and a management infrastructure

Just a configurable architecture

- involves the use of a reference architecture or application framework
- does not involve the planned reuse of other assets

Releases and versions of single products

- involves the sequential release of products over time
- no simultaneous release/support of multiple products

Software Product Lines Are Not - 3

Just a set of technical standards

- constraints to promote interoperability and to decrease the cost associated with maintaining and supporting commercial components
- does not provide assets and production capability

Session 1 Contents

What a Software Product Line Is

What Software Product Lines Are Not

Terminology

Important Terms - 1

Core asset: A reusable artifact or resource that is used in the production of more than one product in a software product line. A core asset may be an architecture, a software component, a requirements statement or specification, a document, a plan, a test case, a process description, or any other useful element of a software production process.

Core asset base: The complete set of core assets associated with a given software product line.

Development: A generic term to describe how software comes to be. It may involve building, contracting, open-market purchasing, reworking existing assets, or any combination of these.

Domain: An area of knowledge or activity characterized by a set of concepts and terminology understood by practitioners in that area.

Important Terms - 2

Product: Deployed software or software-intensive system.

Reuse: Using an item more than once.

Software product line practice: The systematic use of core assets to assemble, instantiate, or generate the multiple products that constitute a software product line.

Strategic reuse: Planned, systematic reuse that implements tightly connected business and technical strategies.

Alternate Terminology

Our Terminology	Alternate Terminology
Product Line	Product Family
Software Core Assets	Platform
Business Unit	Product Line
Product	Customization
Core Asset Development	Domain Engineering
Product Development	Application Engineering

Session 1 Summary

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 involve strategic, planned reuse that differs from earlier software reuse paradigms.

Exercise 1

Examples of non-software and software product lines.

Carnegie Mellon University

Software Engineering Institute

Part 1: Software Product Line Fundamentals
Session 2: Benefits of Software Product Lines

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 2 Objectives

This session will acquaint participants with

- the organizational benefits associated with software product lines
- the individual benefits associated with software product lines
- an examination of the benefits versus the costs of software product lines

Session 2 Contents

Organizational Benefits

Individual Benefits

Costs

Organizational Benefits

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

More Examples of Product Line Benefits - 1

Hewlett Packard (HP) - printer systems

- 2-7x cycle-time improvement (some 10x)
- sample project
  - shipped 5x number of products
  - that were 4x as complex
  - and had 3x the number of features
  - with 4x products shipped/person
- The products from HP's Owen Firmware Cooperative were produced using 1/4 of the staff, in 1/3 of the time, and with 1/25 the number of bugs

Motorola - FLEXworks Project (family of one-way pagers)

- 4x cycle-time improvement
- 80% reuse

More Examples of Product Line Benefits - 2

Bosch – gasoline systems engine control software

- successfully entered new emerging markets
- regained market share in the standard segments
- reduced resource consumption by 20…30%, minimized calibration effort, and supported increased reuse

LSI Logic – RAID controller firmware product line

- in two years, produced nearly 90 different controller firmware products, supporting multiple controller hardware platforms, and multiple customer customizations from one code base

For others, see the Product Line Hall of Fame

http://www.sei.cmu.edu/productlines/plp_hof.html

and the Product Line Catalog

http://www.sei.cmu.edu/productlines/plp_catalog.html

Summary: 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

Session 2 Contents

Organizational Benefits

Individual Benefits

Costs

Individuals Who Benefit

CEO

Architect

COO

Core Asset Developer

Technical Manager

Marketer

End User

Customer

Individuals Benefits - 1

CEO

- option to quickly develop new products
- large productivity gains
- greatly improved time to market
- sustained growth and market presence
- ability to economically capture a market niche

COO

- efficient use of workforce
- ability to explore new markets, new technology, and/or new products
- fluid personnel pool

Technical Manager

- increased predictability
- well-established roles and responsibilities
- efficient production

Individual Benefits - 2

Architect or Core Asset Developer

- greater challenge
- work has more impact
- prestige within the organization
- as marketable as the product line

Software Product Developer

- higher morale
- greater job satisfaction
  - can focus on truly unique aspects of products
  - easier software integration
  - fewer schedule delays
  - greater mobility within organization
  - more marketable outside
  - has time to learn new technology
  - is part of a team, building products with an established quality record and reputation

Individual Benefits - 3

Marketer

- predictable high-quality products
- predictable delivery
- can sell products with a pedigree

Customer

- higher quality products
- predictable delivery date
- predictable cost
- known costs for unique requirements
- well-tested training materials and documentation
- shared maintenance costs
- users' group

Individual Benefits - 4

End User

- product alignment – common “look and feel”
- fewer defects
- better training materials and documentation
- a network of other users

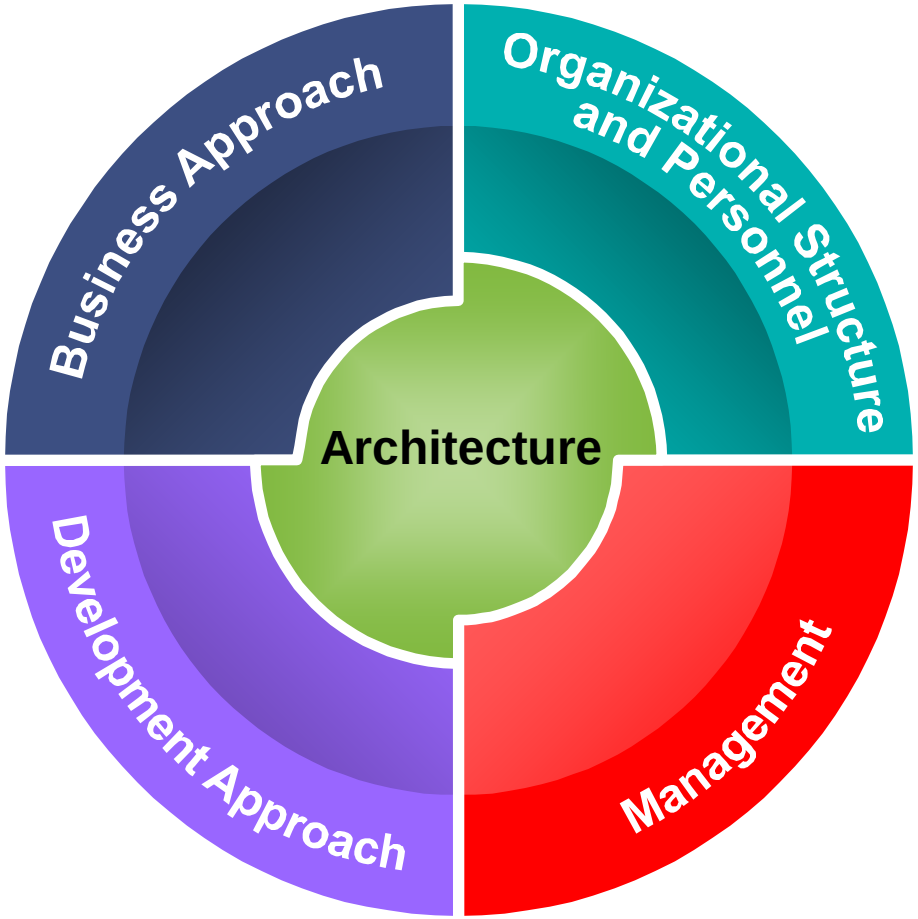
Session 2 Contents

Organizational Benefits

Individual Benefits

Costs

Necessary Changes

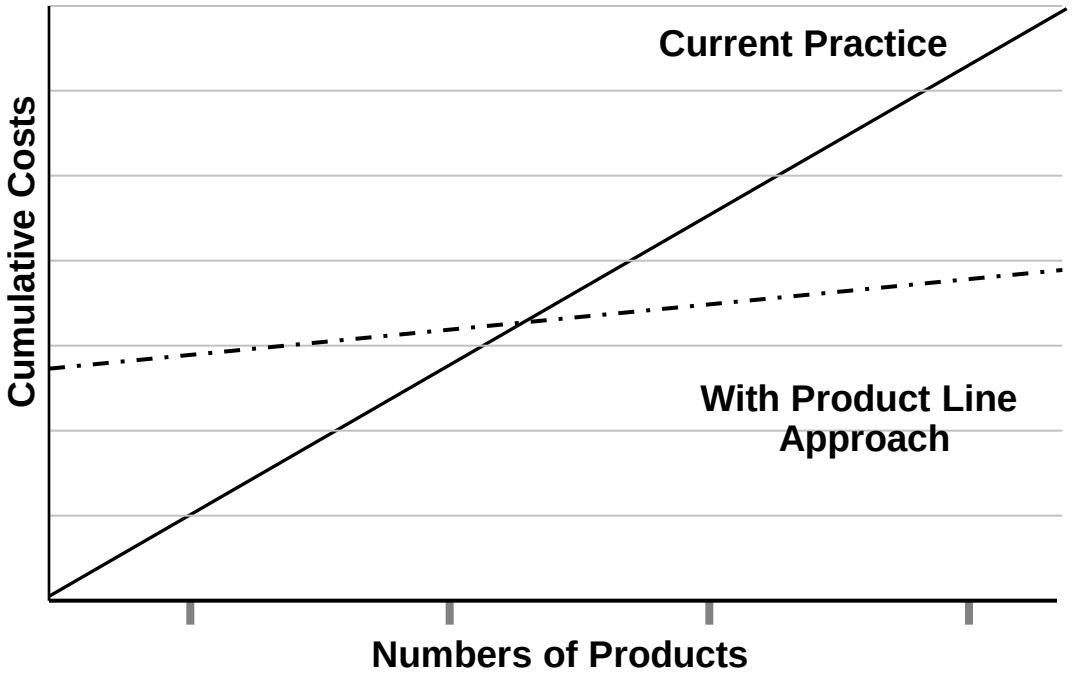


The product line architecture is central to success.

Costs of a Software Product Line

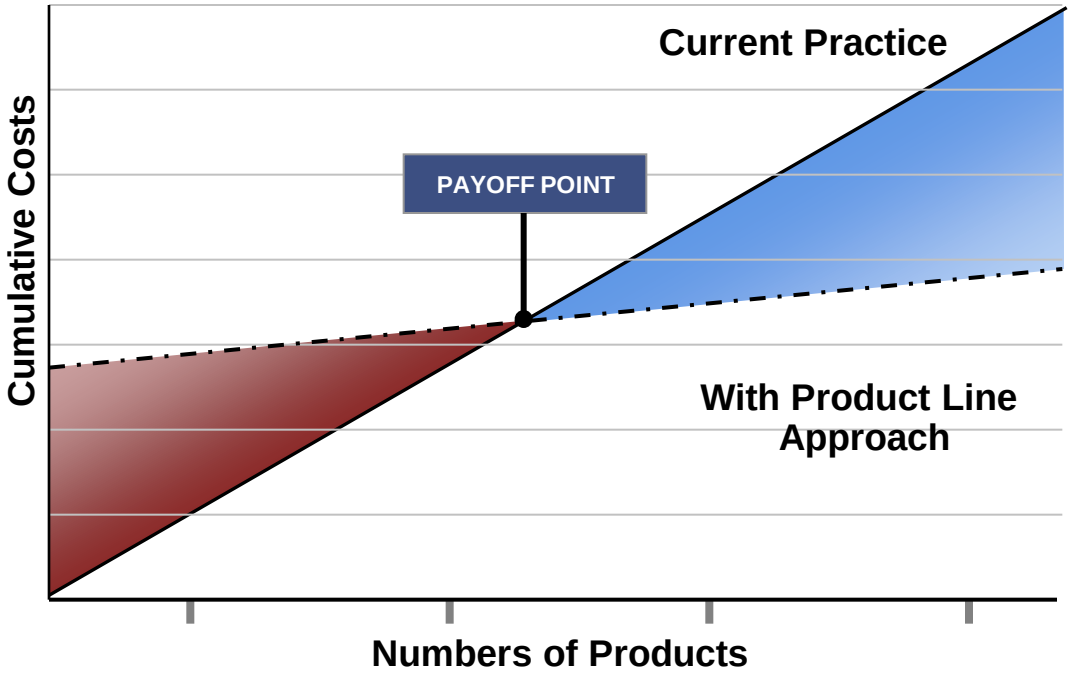
Core Assets	Costs
Architecture	Must support variation inherent in the product line
Software Components	Must be designed to be general without a loss of performance; must build in support for variation points
Test Plans, Test Cases, Test Data	Must consider variation points and multiple instances of the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

Economics of Product Lines



Weiss, D. M. & and Lai, C. T. R.
Software Product-Line Engineering: A Family-Based Software Development Process.
Reading, MA: Addison-Wesley, 1999.

Economics of Product Lines



Weiss, D. M. & and Lai, C. T. R.
Software Product-Line Engineering: A Family-Based Software Development Process.
Reading, MA: Addison-Wesley, 1999.

Session 2 Summary

Software product lines have been proven to yield significant organizational benefits in terms of increased productivity and quality, and decreased cost and time to market.

Software product lines have been proven to benefit individuals in specific and desirable ways.

There are costs associated with a product line approach for software that cannot be ignored.

Discussion

Identify other potential product line core assets and list the benefits and costs associated with each one.

How would you decide whether the potential benefits of a software product line approach outweigh costs and risks in your organization?

Carnegie Mellon University

Software Engineering Institute

Part 1: Software Product Line Fundamentals
Session 3: The Three Essential Activities

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 3 Objective

This session will introduce participants to the three essential activities for software product lines.

Session 3 Contents

What Are the Essential Activities?

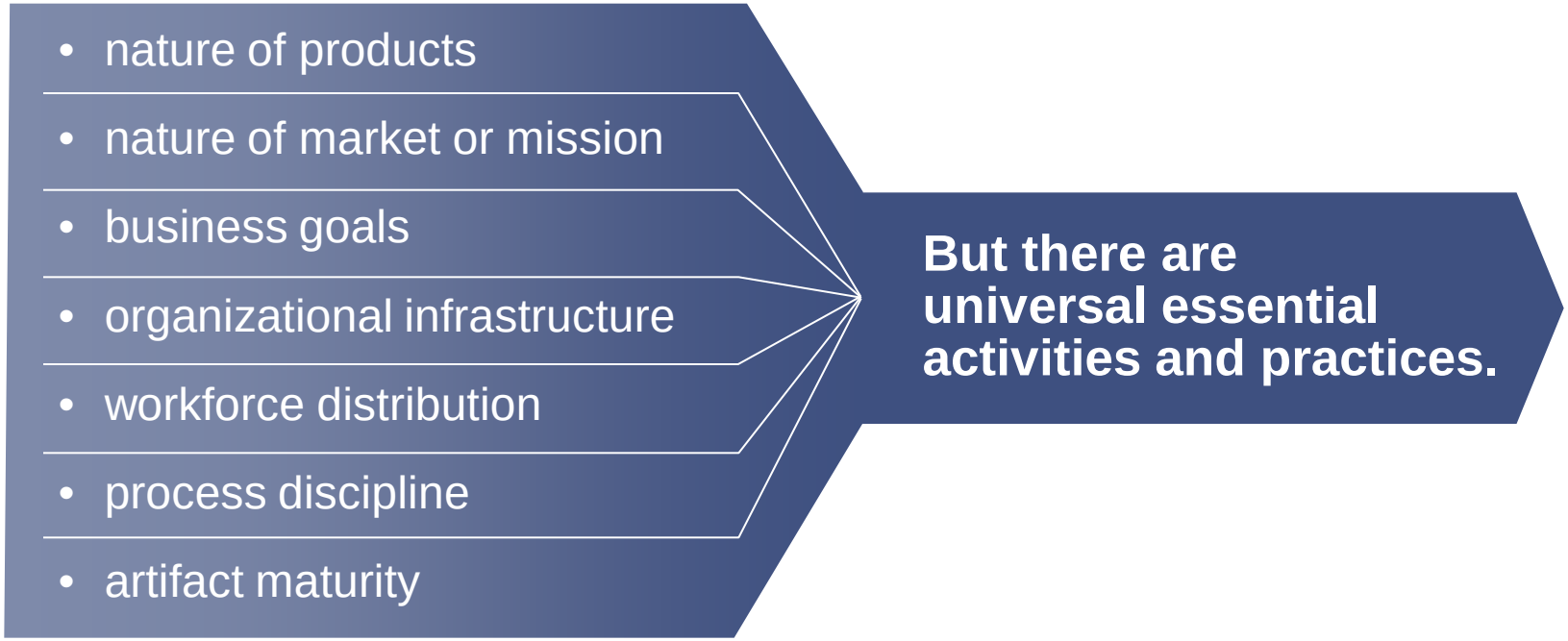
Core Asset Development

Product Development

Management

Product Line Practice

Contexts for product lines vary widely, based on



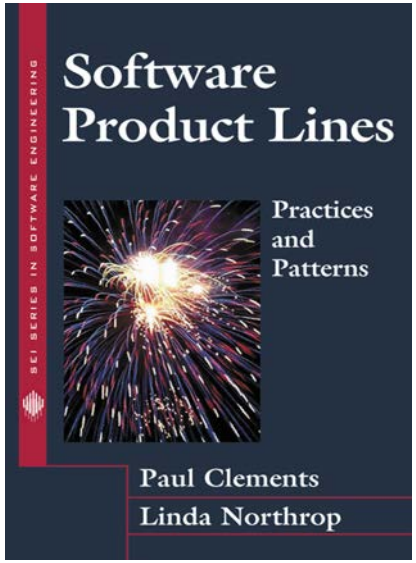
The SEI Framework for Software Product Line Practice[SM]

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 –
http://www.sei.cmu.edu/productlines/framework.html



[SM] Framework for Software Product Line Practice is a service mark of Carnegie Mellon University.

The Goals of the Framework

The goals of the Framework are to

- Identify the foundational concepts underlying software product lines and the essential activities to consider before developing a product line.
- Identify practice areas that an organization developing software product lines must master.
- Define practices in each practice area, where current knowledge is sufficient to do so.
- Provide guidance to an organization about how to move to a product line approach for software.

The Framework is not a maturity model or a process guide.

SEI Information Sources

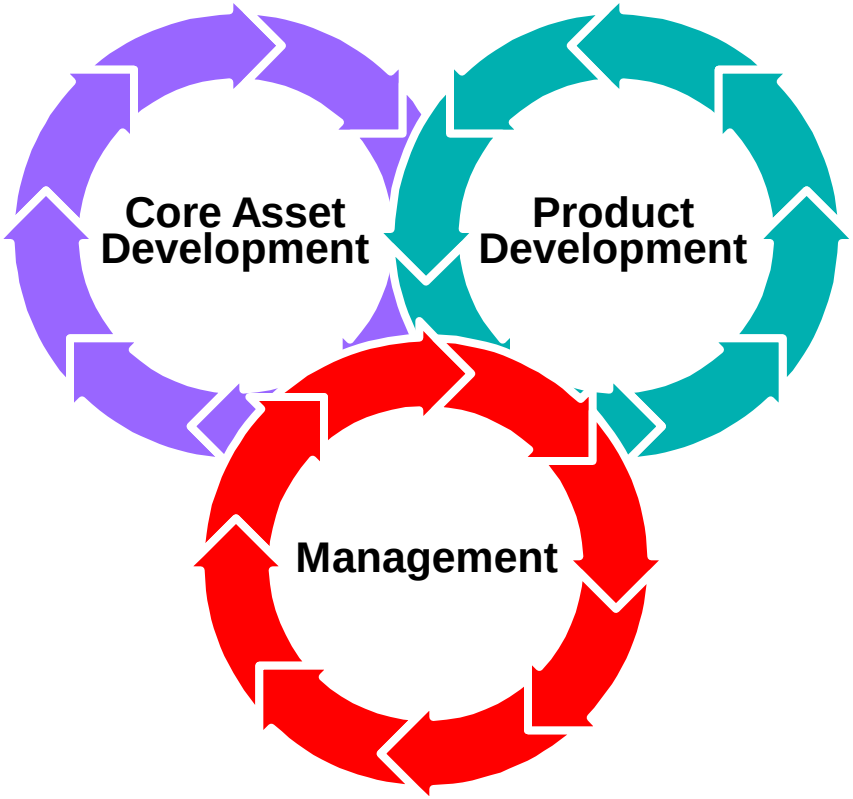
Case studies, experience reports, and surveys

Workshops and conferences

Applied research

Collaborations with customers on actual product lines

The Three Essential Activities



The Nature of the Essential Activities

All three activities are interrelated and highly iterative.

There is no “first” activity.

- In some contexts, existing products are mined for core assets.
- In others, core assets may be developed or procured for future use.

There is a strong feedback loop between the core assets and the products.

Strong management at multiple levels is needed throughout.

Session 3 Contents

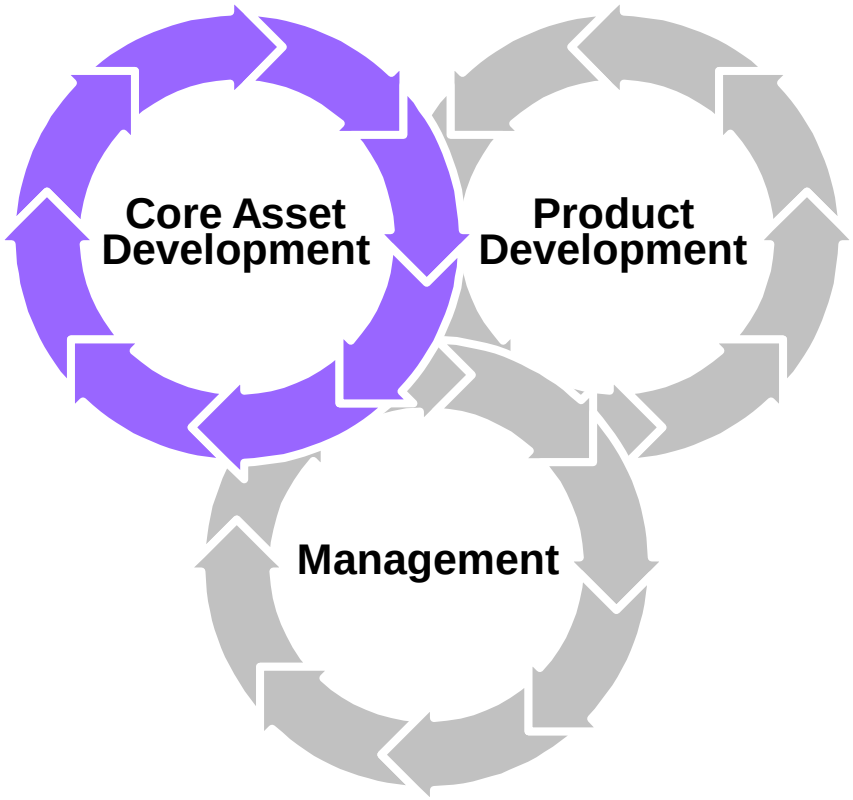
What Are the Essential Activities?

Core Asset Development

Product Development

Management

Core Asset Development



Core Asset Development Goal

The goal of core asset development is to establish a production capability for products.

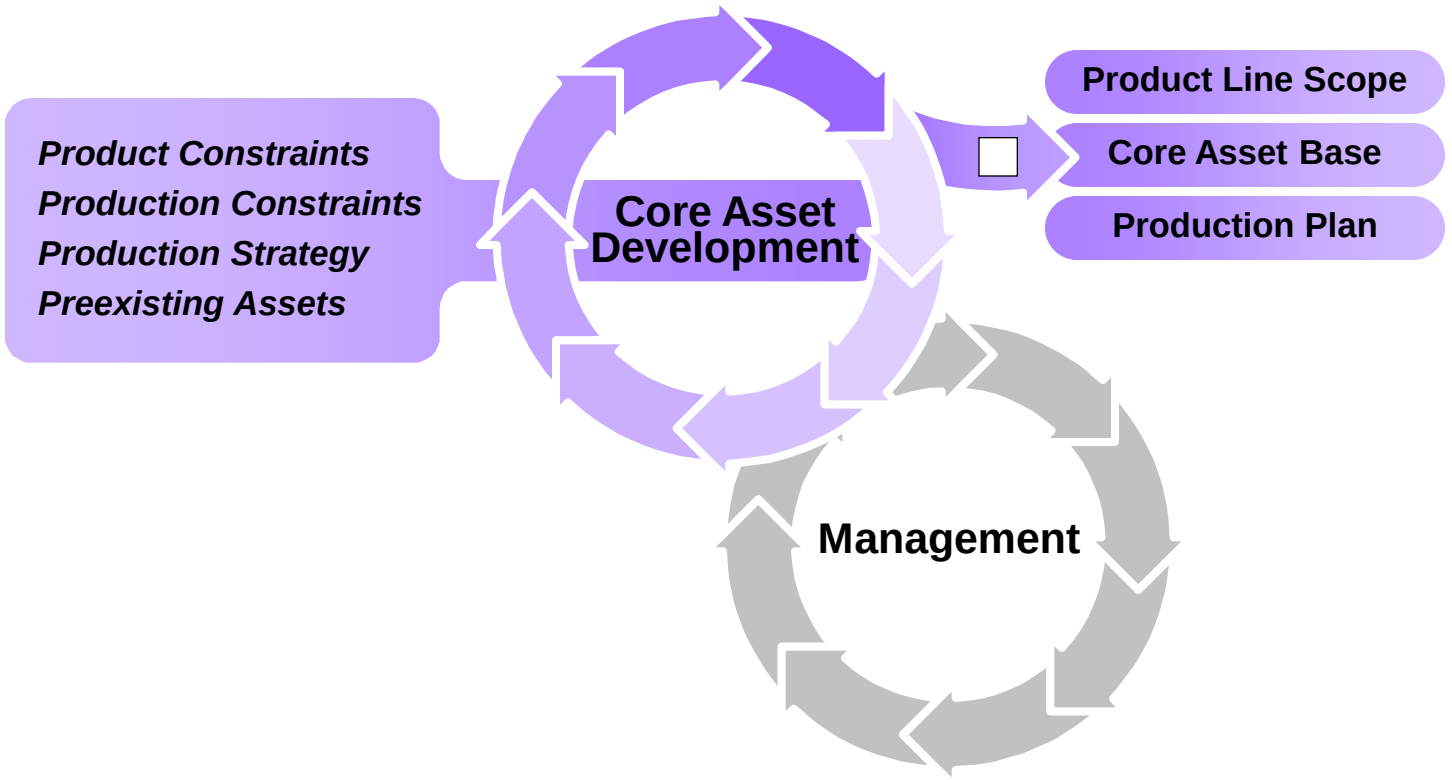
Core asset development is iterative.

- New core assets are added as more commonality among products is planned or discovered.
- Existing core assets are modified to keep pace with product needs.

Core asset development happens in a situational context of existing constraints and resources. This context influences

- how core asset development is carried out
- the nature of the outputs it produces

Core Asset Development



Contextual Factors

1. **Product constraints:** the commonalities and variations among the products in the product line, behavioral features, quality requirements, organizational or domain standards, required infrastructure, preexisting products that will form the basis for the product line, and so forth
2. **Production constraints:** imposed time and resource limits to produce the products in the product line
3. **Production strategy:** the overall approach for realizing both the core assets and products
   a. informed by the product and production constraints
   b. informs the architecture and its components and their growth path
   c. drives the process by which products will be developed from core assets
4. **Preexisting assets:** the intellectual property (legacy assets, off-the-shelf assets, open source assets, Web assets, libraries, frameworks, algorithms, tools) that can be incorporated into the core asset base

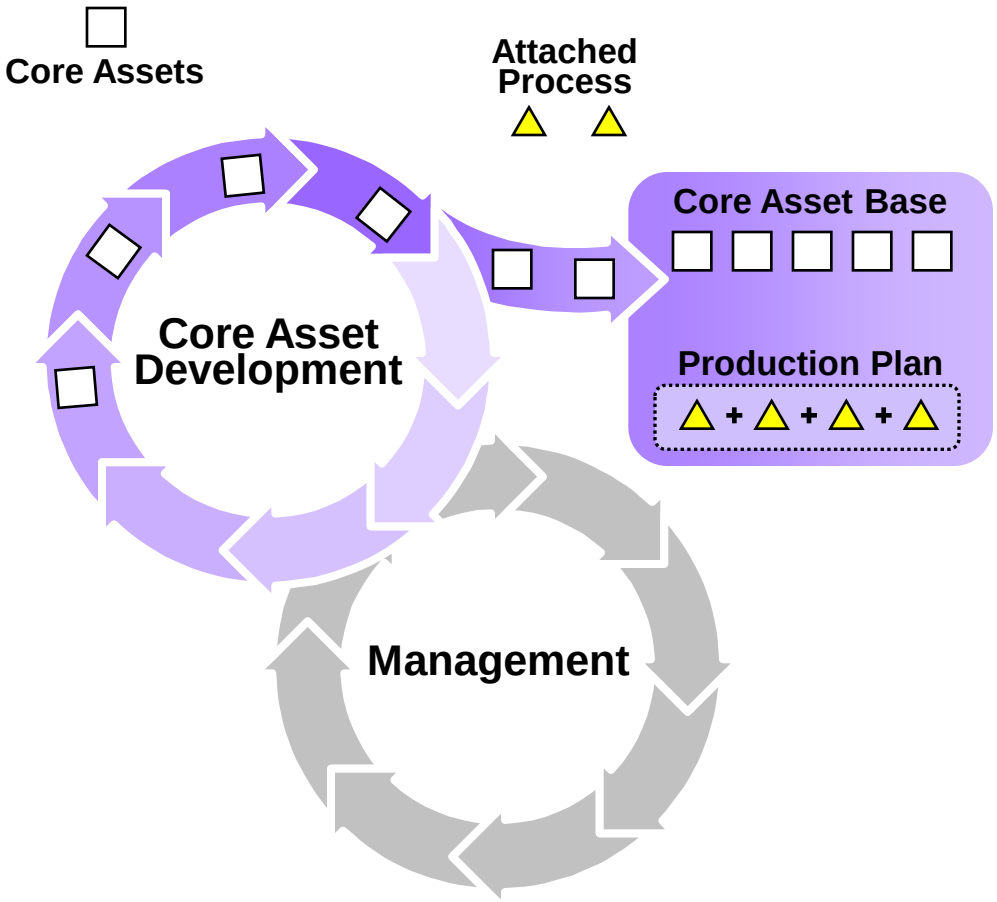
Core Asset Development Outputs - 1

1. **Product line scope:** The product line scope is a description of the products that will constitute the product line or that the product line is capable of including.
   - In its simplest form, the scope may consist of an enumerated list of product names, but more typically is cast in terms of the things the products have in common and the ways in which the products vary.
   - A product line's scope reflects the organization's strategic market goals**.**
   - For a product line to be successful, its scope must be defined carefully.
   - The scope of a product line evolves.
   - Defining the product line scope is often referred to as *scoping.*

Core Asset Development Outputs - 2

2. **Core asset base:** The core asset base includes all the core assets, which are the basis for the production of products in the product line.
   - Not every core asset will be used in every product in the product line, but all will be used in enough products to make their coordinated development, maintenance, and evolution pay off.
   - Among the core assets is the product line architecture—a software architecture that will satisfy the needs of the product line in general and the individual products in particular by explicitly admitting a set of variation points.
   - Other technical core assets may include components, models, test plans, test cases, requirements specifications, tools, processes, and production infrastructure.
   - Nontechnical core assets may include schedules, budgets, plans, training materials, marketing collateral, and a concept of operations (CONOPS) that describes how the organization operates as a product line organization.
   - Each core asset should have associated with it an *attached process* that specifies how it will be used in the development of actual products.

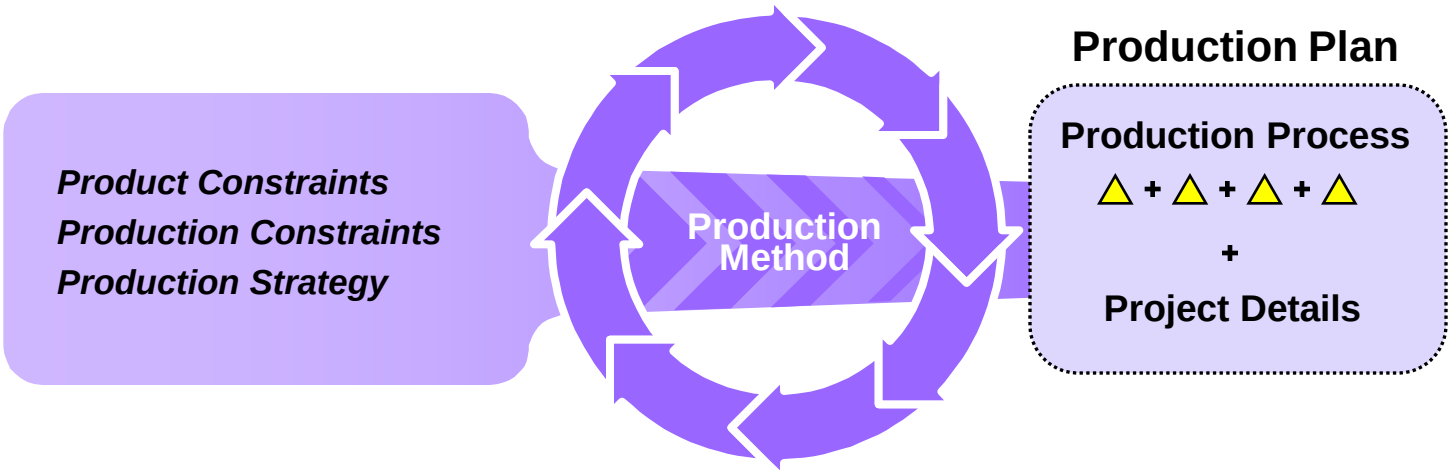
Attached Processes



Core Asset Development Outputs - 3

3. **Production plan:** A production plan prescribes how the products are produced from the core assets. The production plan fulfills two roles:
   a. It includes the process to be used for building products—the ***production process*** that is essentially a set of the attached processes with the necessary process "glue."
      - must indicate how variation points will be exercised
      - must appropriately address the product builders for whom it is intended
      - should describe how specific tools are to be used
      - satisfies the production strategy and production constraints
      - reflects the ***production method***, which is the overall implementation approach that specifies the models, processes, and tools to be used in the attached processes
   b. It lays out the project details to enable execution and management of the production process.
      - includes schedule, bill of materials, and metrics

Product Line Production Plan



Session 3 Contents

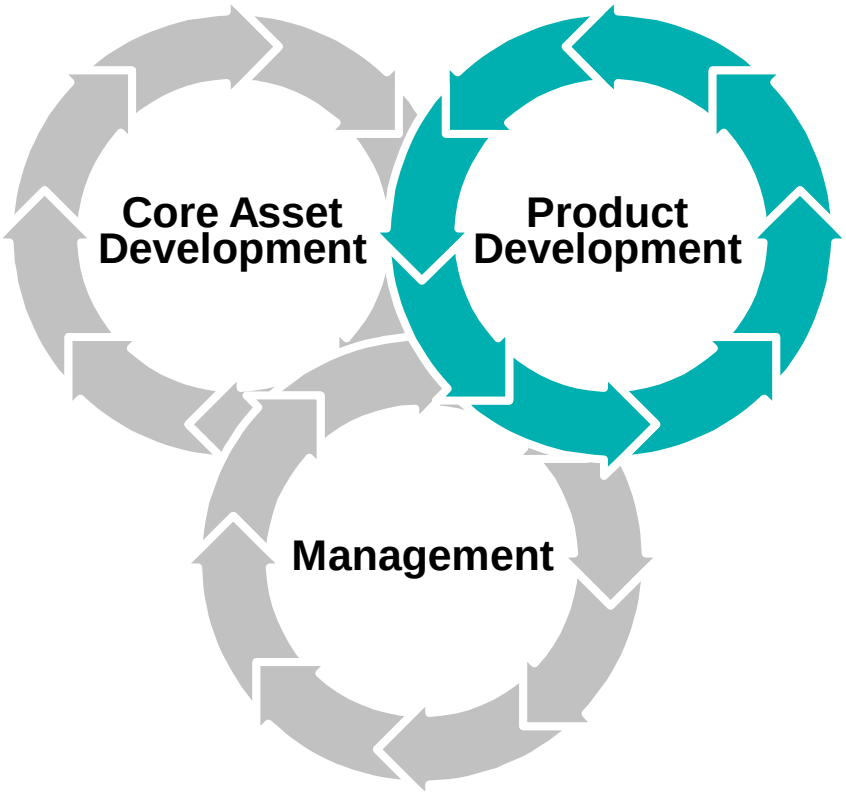
What Are the Essential Activities?

Core Asset Development

Product Development

Management

Product Development



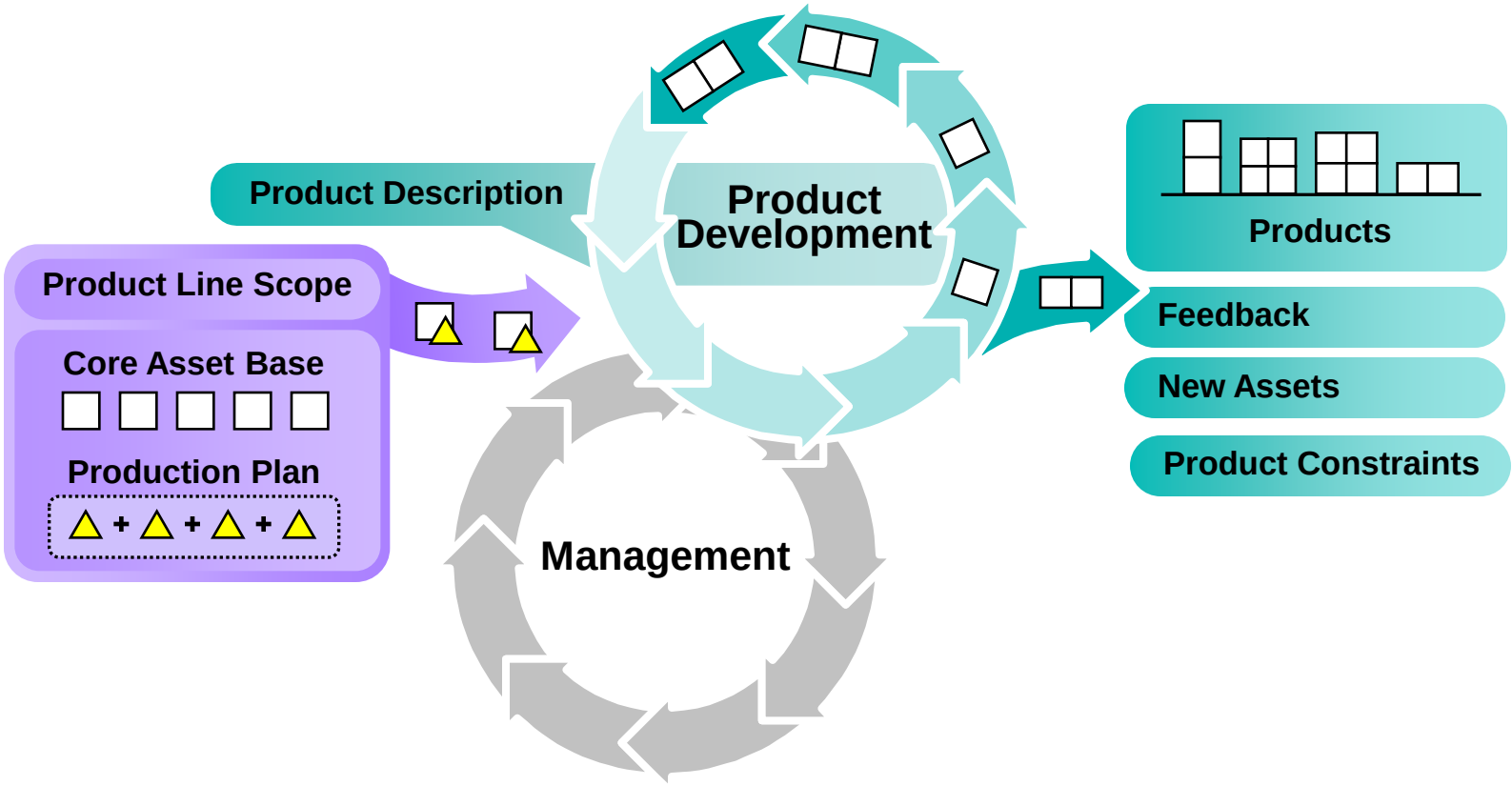
Product Development Goal

The activity of efficiently turning out products is the ultimate product line goal.

Mature product line organizations, however, prioritize the health of the overall product line, especially the core asset base, over the production of individual products.

Those organizations recognize that nurturing their production capability enables them to rapidly produce many products.

Product Development



Product Development Inputs and Outputs

Product development depends on

- three inputs that are the outputs of core asset development (the product line scope, core assets, and production plan)
- plus the product description for each individual product

The creation of products also has a strong feedback effect on the product line providing

- feedback to improve existing core assets
- new assets
- new product constraints

As a special case, the products developed could actually be components.

Session 3 Contents

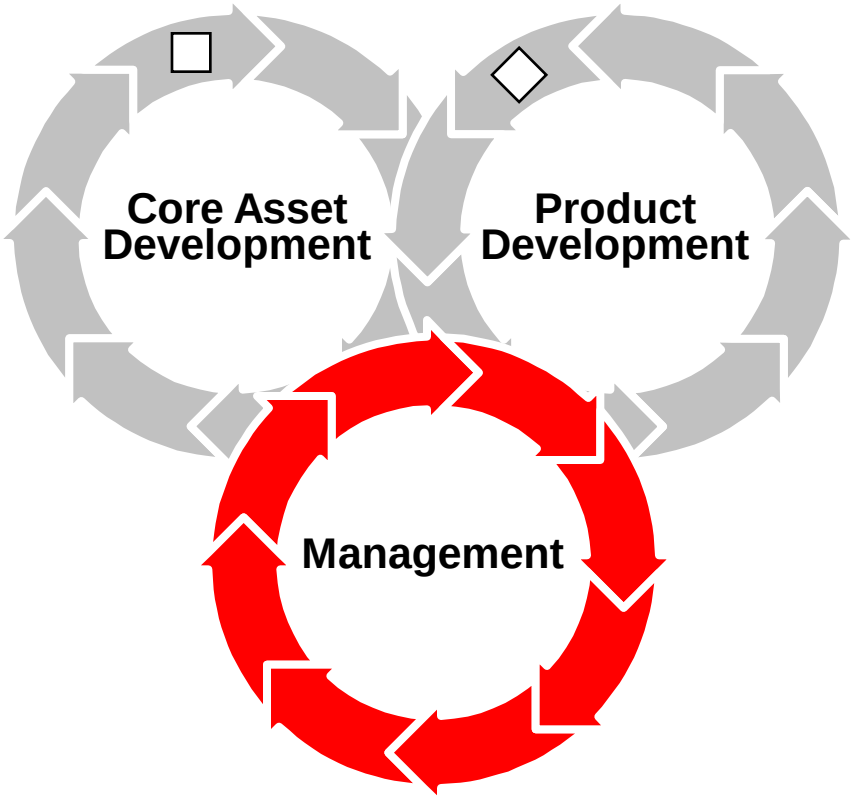
What Are the Essential Activities?

Core Asset Development

Product Development

Management

Management

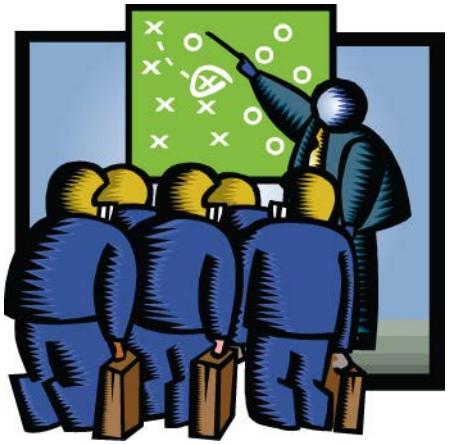


Management - 1

Management at multiple levels plays a critical role in successful product line practice.

Technical management

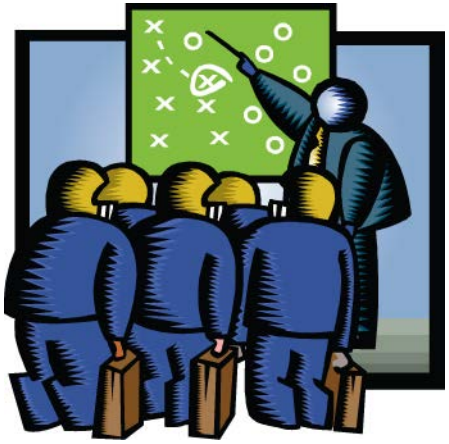
- oversees core asset development and product development
  - makes sure those involved are engaged in the required activities, follow processes defined for the product line, and collect data sufficient to track progress
- provides the project management elements of the production plan
- decides on the production method



Management - 2

Organizational management

- identifies production constraints and ultimately determines the production strategy
- determines a funding model
- achieves the right organizational structure
- allocates resources
- orchestrates the technical activities
- provides training
- rewards employees appropriately
- develops and communicates an acquisition strategy
- manages external interfaces
- creates and implements a product line adoption plan
- launches and institutionalizes the approach in a manner appropriate to the organization



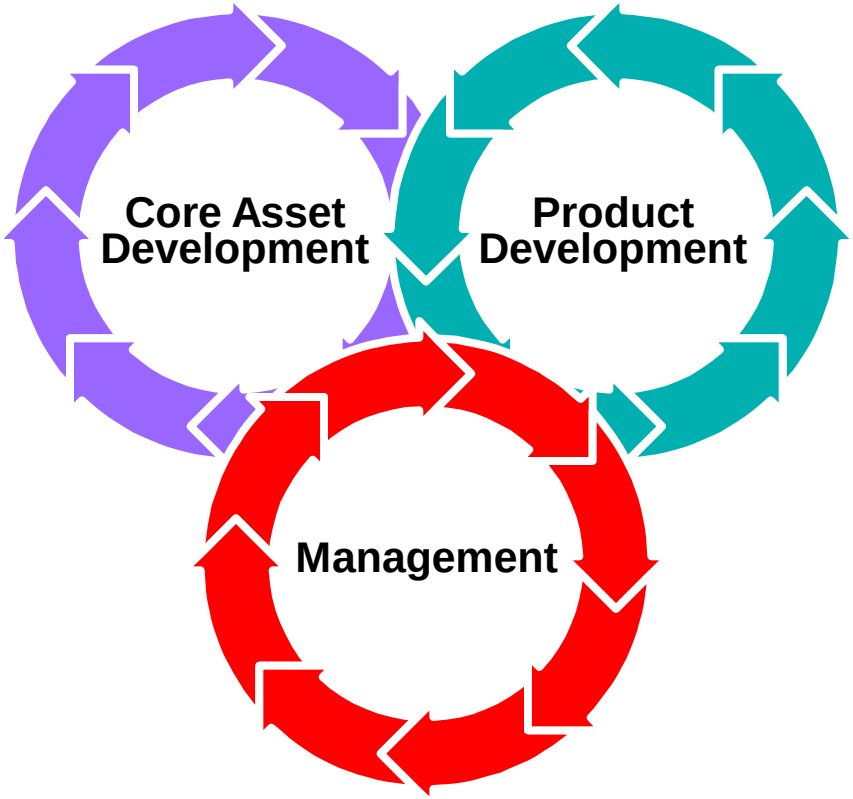
Managing a Software Product Line Requires Leadership

Some individual or group should be designated to fill the product line management role and act as a product line champion (or find and empower one).

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

Essential Product Line Activities



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 up-front 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.
- …

Session 3 Summary

There are three essential activities involved in a software product line approach: core asset development, product development, and management.

Core asset development establishes the production capability for products.

Product development yields the products in the product line.

Management, at both organizational and technical levels, orchestrates the product line effort and ensures that the right practices are in operation.

Software product lines require a blend of technology and business practices.

Discussion

Describe what might be an attached process for each of the following: a product line architecture, a component, a test case, and a product development estimate.

Can you imagine a situation in which an organization might have to choose between the health of its product line and the production of a specific product?

What are telltale signs that it is doing one or the other?

If you wanted to move to a product line approach in your organization, which approach (proactive or reactive) would you take? Why?

Would you use an incremental approach? Why?

Carnegie Mellon University

Software Engineering Institute

Part 2: Software Product Line Practice Areas
Session 1: Practice Areas

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Course Structure

Course Introduction

Part 1: Software Product Line Fundamentals

Part 2: Software Product Line Practice Areas

Part 3: Putting the Practice Areas into Action

Wrap-Up

Driving the Essential Activities

Supporting 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.

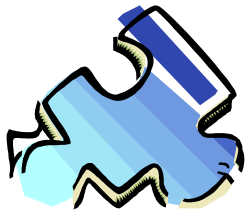
The practice areas represent common activities in software development that are adapted to the needs of a product line approach.

Practice areas help to make the essential activities more achievable by defining activities that are smaller and more tractable than a broad imperative such as “develop core assets.”

Session 1 Objective

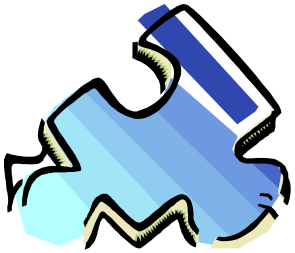
This session will introduce participants to the concept of product line practice areas.

Practice Area Descriptions

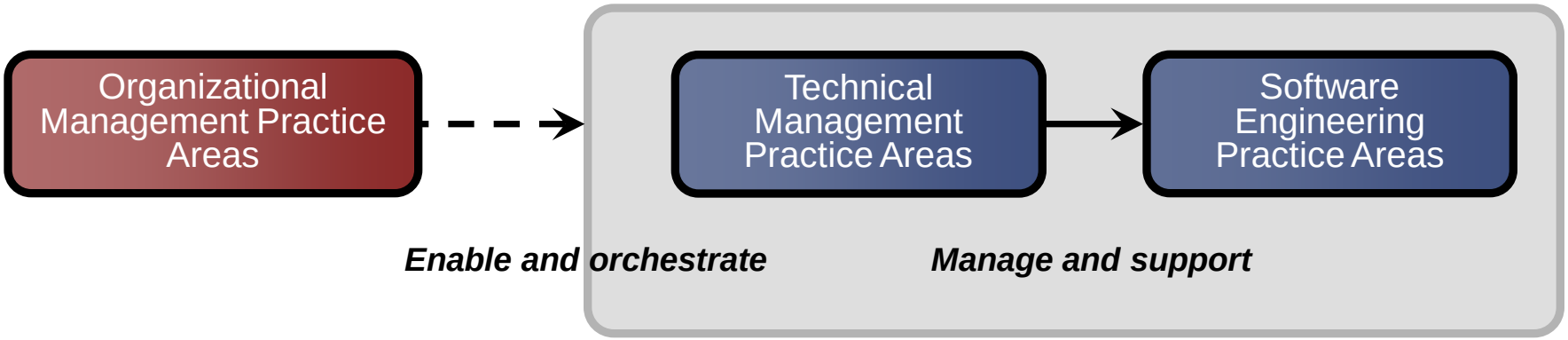


Each practice area is described with

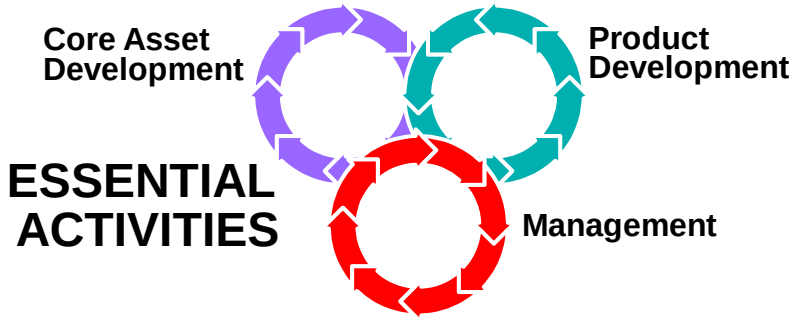
- an introductory description
- aspects that are peculiar to product lines
- its application to core asset development
- its application to product development
- example practices
- associated risks
- further reading



Three Categories of Practice Areas



Framework Version 5.0



PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management
Architecture Definition	Configuration Management	Building a Business Case
Architecture Evaluation	Make/Buy/Mine/Commission Analysis	Customer Interface Management
Component Development	Measurement and Tracking	Developing an Acquisition Strategy
Mining Existing Assets	Process Discipline	Funding
Requirements Engineering	Scoping	Launching and Institutionalizing
Software System Integration	Technical Planning	Market Analysis
Testing	Technical Risk Management	Operations
Understanding Relevant Domains	Tool Support	Organizational Planning
Using Externally Available Software		Organizational Risk Management
		Structuring the Organization
		Technology Forecasting
		Training

Session 1 Summary

Software product line practice areas make the essential activities more achievable by defining activities that are smaller and more tractable than the three essential activities.

Practice areas are divided into three categories:

1. software engineering
2. technical management
3. organizational management

Carnegie Mellon University

Software Engineering Institute

Part 2: Software Product Line Practice Areas
Session 2: Software Engineering Practice Areas

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 2 Objectives

This session will

- introduce participants to the software engineering practice areas, in general
- acquaint participants with the following specific software engineering practice areas:
  - Understanding Relevant Domains
  - Requirements Engineering
  - Architecture Definition
  - Component Development

Session 2 Contents

Software Engineering Practice Areas

Understanding Relevant Domains

Requirements Engineering

Architecture Definition

Component Development

Software Engineering Practice Areas - 1

Software engineering practice areas are those necessary for applying the appropriate technology to create and evolve both the core assets and products.

Software Engineering Practice Areas - 2

Architecture Definition
Architecture Evaluation
Component Development
Mining Existing Assets
Requirements Engineering
Software System Integration
Testing
Understanding Relevant Domains
Using Externally Available Software

Session 2 Contents

Software Engineering Practice Areas

Understanding Relevant Domains

Requirements Engineering

Architecture Definition

Component Development

Understanding Relevant Domains - 1

Involves systematically capturing and using knowledge of systems similar to those that will be in the product line.

Deep and broad domain understanding is critical.

Domain knowledge is characterized by

- a set of concepts and terminology understood by practitioners in that area of expertise
- an understanding of recurring problems
- known solutions within the domain

Understanding Relevant Domains - 2

The practice of understanding the relevant domains involves the following responsibilities:

- identifying the appropriate areas of expertise (domains)
- identifying the recurring problems and known solutions within these domains
- capturing and representing this information in ways that allow it to be communicated to the stakeholders, and used and reused across the entire effort

You need to have enough information to make sound business decisions without necessarily undertaking an extensive analysis of all applicable domain knowledge.

Understanding Relevant Domains - 3

How is the understanding achieved?

- from a reservoir of expertise from prior experience
- from hired outside experts
- through domain analysis methods that gather, organize, and communicate domain information in the form of a domain model

The extent to which a formal analysis is performed depends on

- the depth of the organization's domain experience
- the amount of resources that can be devoted to analysis

Domain understanding should include whatever key domain information can be used as the vehicle for analysis and reasoning.

Understanding Relevant Domains: Aspects Peculiar to Product Lines

Understanding the relevant domains is the first step to defining the *commonality and variation* expected across the product line.

Domain information for a product line helps determine

- which capabilities tend to be common across systems in the domain(s) and which variations are present
- which subsets of capabilities might be packaged together as assets for the product line
- what constraints apply to systems in the domain
- what the key terms are and how they are defined
- which assets typically constitute members in the domain
- whether to continue with a product line development effort

Domain understanding informs the product line's scope and its associated business case, which in turn drive the architecture and the other core assets.

Understanding Relevant Domains: Example Practices

Have domain experts available

Scope, commonality, and variability (SCV) analysis

Domain analysis and design process (DADP)

Feature-oriented domain analysis (FODA)

Synthesis process of the reuse-driven software processes (RSP) approach

Domain analysis of organizational domain modeling (ODM)

Product Line Software Engineering Customizable Domain Analysis (PuLSE-CDA)

Domain-Specific Modeling (DSM)

Understanding Relevant Domains: Practice Risks

Inadequate domain understanding in the organization will jeopardize the product line effort.

Inadequate domain understanding can result from

- analysis paralysis
- lack of access to the necessary domain expertise
- inadequate documentation and sharing of relevant domain information
- lack of understanding of an analysis process
- lack of appropriate tool support for the process and products of domain analysis
- lack of management commitment

Understanding Relevant Domains: For Further Reading

[Arango 1994a]

[Ardis 2000a]

[Bayer 2000a]

[Geppert 2003a]

[Kang 1990a]

[Kang 1998a]

[Lee 2000a]

[SPC 1993b]

[STARS 1996a]

[Tolvanen 2005a]

[Weiss 1999a]

Session 2 Contents

Software Engineering Practice Areas

Understanding Relevant Domains

Requirements Engineering

Architecture Definition

Component Development

Requirements Engineering - 1

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements… No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later."

– Fred Brooks, 1987

Requirements are statements of

- what the system must do
- how it must behave
- the properties it must exhibit
- the qualities it must possess
- the constraints that the system and its development must satisfy

Requirements Engineering - 2

Requirements engineering consists of

- *requirements elicitation*: discovering, reviewing, documenting, and understanding the user's needs and constraints
- *requirements analysis*: refining the user's needs and constraints
- *requirements specification:* documenting the user's needs and constraints clearly and precisely
- *requirements verification:* ensuring that the system requirements are complete, correct, consistent, and clear
- *requirements management:* scheduling, coordinating, and documenting the requirements engineering activities

[Dorfman 1997a]

Requirements Engineering - 3

Conceptually, there are three roles at work:

1. requestor – originator of the requirements who may come from many different organizations
2. developer – designer/implementer of the system
3. author – writer of the requirements

The requirements are pervasive and long lived, affecting all stages of the system's (potentially very long) life.

The longer the system's lifetime, the more it is exposed to changes in the requirements.

Maintaining traceability is a key task during requirements engineering:

- across time
- across products

Requirements Engineering: Aspects Peculiar to Product Lines - 1

Product line requirements are **common** requirements (those that all members of the product line share), and they must be managed:

- The common requirements are maintained as a separate core asset.
- The common requirements are written with variation points.
- Requirements for a particular product exercise or fill in those variation points.
- Product-specific requirements are stored as a set of deltas relative to the product line requirements.
- The deltas can be an addendum, a set of changes, or a set of parameter instantiations.

Requirements Engineering: Aspects Peculiar to Product Lines - 2

Requirements **elicitation** must explicitly capture anticipated variations over the foreseeable lifetime of the product line. The scope is input.

Requirements **analysis** involves finding commonalities and identifying variations.

Requirements **specification** includes preparing a product-line-wide set of requirements and product-specific requirements.

Requirements **verification** includes a broader reviewer pool and occurs in stages.

Requirements **management** must allow for the dual/staged (common, specific) nature of the requirements process.

Requirements Engineering: Aspects Peculiar to Product Lines - 3

Product line requirements

- use the product line scope as input
- validate that scope and its associated business case
- drive the architecture and software core assets
- drive the product-line-wide testing
- form the basis for product schedules and budgets

Requirements Engineering: Example Practices - 1

Domain analysis techniques can be used to expand the scope of the requirements elicitation, to identify and plan for anticipated changes, to determine fundamental commonalities and variations in the products of the product line, and to support the creation of robust architectures.

Stakeholder-view modeling can support the prioritized modeling of the significant stakeholder requirements for the product line.

Feature modeling can be used to complement object and use case modeling and to organize the results of the commonality and variation analysis in preparation for reuse.

Requirements Engineering: Example Practices - 2

Use case modeling with variation points can be used to capture and describe commonality and variation within product line requirements. A *variation point* is a location within a use case where a variation (i.e., a variability) occurs.

Change-case modeling can be used to explicitly identify and capture anticipated changes. Change cases allow designers to plan for change.

Traceability of requirements to their associated work products can be used to ensure that the design and implementation of a product satisfies the requirements for that product.

Requirements Engineering: Practice Risks

The major risk is inappropriate requirements. Inappropriate requirements result from

- failing to distinguish between product-line-wide and product-specific requirements
- insufficient generality, which results in a design that is too brittle
- excessive generality, which leads to excessive effort in core asset production
- having the wrong variation points, which leads to an inability to be flexible to customers' needs
- failing to account for qualities other than functions, which results in products that don't meet required quality attribute goals

Requirements Engineering: For Further Reading

[Birk 2003a]

[Chastek 2001a]

[Davis 1990a]

[Dorfman 1997a]

[Faulk 1997a]

[Schmid 2006a]

[Sommerville 1997a]

Session 2 Contents

Software Engineering Practice Areas

Understanding Relevant Domains

Requirements Engineering

Architecture Definition

Component Development

Architecture Definition - 1

The software architecture of a software system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.[1]

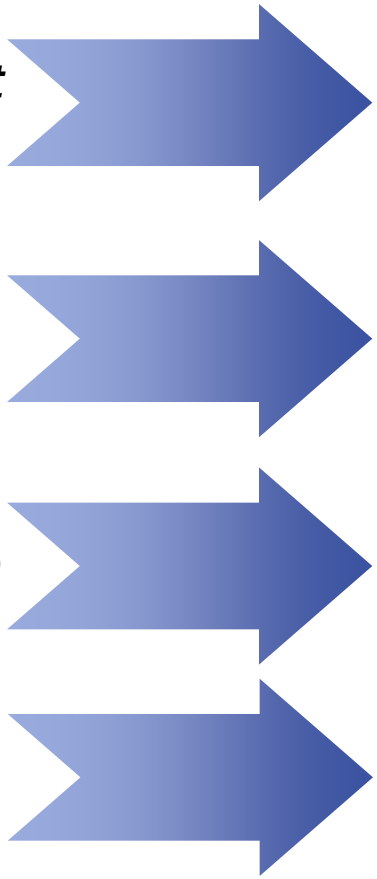
Architecture is

- the blueprint for a system and the project building it
- the carrier of most system quality attributes
- a forum for resource tradeoffs
- a contract that allows multi-party development
- an essential part of complex systems

[1] Bass, L.; Clements, P. & Kazman, R. *Software Architecture in Practice, 2nd Edition.* Reading, MA: Addison-Wesley, 2003.

Architecture Definition - 2

Architecture



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

Key to system ***evolution***

- manage future uncertainty
- assure cost-effective agility

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

Architecture Definition - 3

Defining an architecture involves

- knowing the architecturally significant requirements and constraints
- designing the important architectural structures
- defining the elements and relations the structures comprise
- specifying element interfaces and providing infrastructure services

Defining an architecture carries the additional obligations of

- documenting the architecture in appropriate views
- communicating it
- evaluating it for fitness of purpose
- assuring conformance to it

Architecture Definition: Aspects Peculiar to Product Lines - 1

Of all a product line's core assets, the architecture may well be the most important one for ensuring technical success.

A product line architecture must

- apply to all members of the product line (even if their functions and quality attributes differ)
- embody the commonalities and accommodate the variations needed by the products
- include specific mechanisms for variation

The variation mechanisms chosen must support

- the variations reflected in the products
- the production strategy and production constraints
- efficient integration

Architecture Definition: Aspects Peculiar to Product Lines - 2

The product line architecture relates to product line requirements:

- Common requirements tend to map to the non-variable parts of the architecture.
- Product-specific requirements tend to map to the variations provided for by the architecture.

Both types of requirements require planning for the **managed set of features** that define the product line.

Each product may have its own architecture, which is an instance of the product line architecture achieved by exercising the variation mechanisms.

Architecture Definition: Example Practices - 1

Understanding the requirements for the architecture

- SEI Quality Attribute Workshop
- use of the production strategy
- planning for architectural variation

Architecture Definition: Example Practices - 2

Designing the architecture

- architecture definition and architecture-based development approaches (e.g., RUP)
- architectural patterns: client-server, n-tier, cooperating process, layered, and so forth
  - Such patterns have known quality attributes and areas of applicability.
  - Patterns impose a vocabulary of design.
- the SEI Attribute-Driven Design (ADD) method
  - Given desired quality attributes, it helps the architect choose architectural strategies/tactics.
- service-oriented architectures: work by “stringing together” services (that have specific, well-defined functionality) into chains that do sophisticated, useful work
- aspect-oriented software development
  - an approach for modularizing system properties that otherwise would be distributed across modules

Architecture Definition: Example Practices - 3

Architecture variation mechanisms

- component omission or replication
- component substitution where components may be
  - aspects
  - code components
  - plug-ins
  - services
- parameterization (including macros, templates)
- compile-time selection of different implementations (e.g., *#ifdef*)
- inheritance, specialization, and delegation
- configuration and module interconnection languages
- generation and generators
- application or service component frameworks

Architecture Definition: Example Practices - 4

Documenting the architecture

- Unified Modeling Language (UML)
- the SEI “Views and Beyond” approach, which first documents relevant views and then information that applies across views
- specifying component interfaces using a contractual approach

Architecture Definition: Practice Risks - 1

Having an unsuitable product line architecture will result in

- components that do not fit together or interact properly
- products that do not meet their behavioral, performance, or other quality attribute goals
- products that should be in the scope, but cannot be produced from the core assets at hand
- a tedious and ad hoc product-building process

Unsuitable architectures are characterized by

- inappropriate parameterization
- inadequate specifications
- flawed design of the important architectural structures
- wrong level of specificity
- excessive inter-element dependencies

Architecture Definition: Practice Risks - 2

Unsuitable architectures can result from

- the lack of a skilled architect
- the lack of sound input
- poor communication
- lack of supportive management and culture
- architecture in a vacuum
- poor tools
- poor timing

Architecture Definition: For Further Reading

General software architecture

[Bass 2003a]

[Buschmann 1996a]

[Hofmeister 2000a]

[Schmidt 2000a]

[SEI 2007b]

[Shaw 1996a]

Product line architecture

[Bosch 2000a]

Problem solving

[Jackson 2000a]

Architecture documentation

[Clements 2000a]

[OMG 2007a]

From an object-oriented perspective

[Booch 1994a]

[Buschmann 1996a]

[Jacobson 1997a]

[Krutchen 2004a]

[Smith 2001a]

Discussion

1. Why are requirements engineering, scoping, and understanding relevant domains closely related in a product line context? Given their close relationship, would you use the same or different people to carry out each practice area?

2. Which design or architecture patterns might be especially applicable to a product line architecture? Why?

3. Which qualities should a product line architect possess?

Session 2 Contents

Software Engineering Practice Areas

Understanding Relevant Domains

Requirements Engineering

Architecture Definition

Component Development

Component Development - 1

A software architecture defines a list of components that populate it.

This list gives the development, mining, and acquisition teams their marching orders for supplying the parts that the software system will comprise.

Component Development - 2

Components are the units of software that go together to form whole systems (products), as dictated by the software architecture for those products. More precisely

> *"A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties."*[1]

Component development refers to the production of components that implement specific functionality within the context of a software architecture.

1 Szyperski, C. *Component Software: Beyond Object-Oriented Programming, 2nd Ed.* Reading, MA: Addison-Wesley, 2002.

Component Development: Aspects Peculiar to Product Lines

In a product line, developed components are either included in the core asset base—and hence used in multiple products in the product line—or are product-specific components.

Components that are included in the core asset base must support the flexibility needed to satisfy the variation points specified in the product line architecture and/or the product line requirements.

The singular aspect of component development that is peculiar to product lines is providing required variation in the developed components.

Component Development: Example Practices - 1

Variation mechanisms

- **aggregation/delegation:** an object-oriented technique in which, the functionality of an object is extended by delegating work it cannot normally perform to an object that can
- **inheritance:** assigns base functionality to a superclass, and extended or specialized functionality to a subclass
- **overloading:** reusing a named functionality to operate on different data types
- **object attributes in Delphi:** Variation is achieved by modifying the attribute values or the actual set of attributes.
- **dynamic class loading in Java:** classes are loaded into memory when needed. A product can query its context and that of its user to decide at runtime which classes to load.
- **static libraries:** contain external functions that are linked to after compilation time. By changing the libraries, one can change the implementations of functions whose names and signatures are known.

Component Development: Example Practices - 2

Variation mechanisms

- **dynamic link libraries:** give the flexibility of static libraries but defer the decision until runtime based on the context and execution conditions
- **conditional compilation:** puts multiple implementations of a module in the same file, with one chosen at compile time by providing the appropriate pre-processor directives
- **reflection:** the ability of a program to manipulate data that represents information about itself, or its execution environment or state. Reflective programs can adjust their behavior based on their context.
- **aspect-oriented programming**: (Already described in the “Architecture Definition” practice area.)
- **design patterns:** extensible, OO solution templates catalogued in various handbooks

Component Development: Practice Risks

The overriding risk in component development is building unsuitable components for the software product line applications.

Doing so will result in

- poor product quality
- inability to field products quickly
- low customer satisfaction
- low organizational morale

Unsuitable components can result from

- an ill-conceived component development effort
- insufficient variability
- too much variability
- choice of the wrong variation mechanism(s)
- poor-quality components

Component Development: For Further Reading

[Anastasopoulos 2000a]

[Jacobson 1997a]

[Szyperski 2002a]

Session 2 Summary

Software engineering practice areas are those necessary for applying the appropriate technology to create and evolve both the core assets and products.

There are nine software engineering practice areas.

It is essential to have a solid understanding of the domains involved in the product line before proceeding with a product line effort.

The product line requirements, architecture, and components must address not only the common features and qualities across the products in the product line but also the variation among them. Explicit variation mechanisms are necessary.

Discussion

1. The “Architecture Evaluation” practice area is one we didn’t examine in detail. What are some important quality attributes to consider when evaluating a product line architecture?

2. Should every component be designed with the same variation mechanism, or is there a case to be made for different components using different mechanisms to achieve their variability?

Carnegie Mellon University

Software Engineering Institute

Part 2: Software Product Line Practice Areas
Session 3: Technical Management Practice Areas

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 3 Objectives

This session will

- introduce participants to the technical management practice areas, in general
- acquaint participants with the following specific technical management practice areas:
  - Scoping
  - Configuration Management
  - Measurement and Tracking

Session 3 Contents

Technical Management Practice Areas

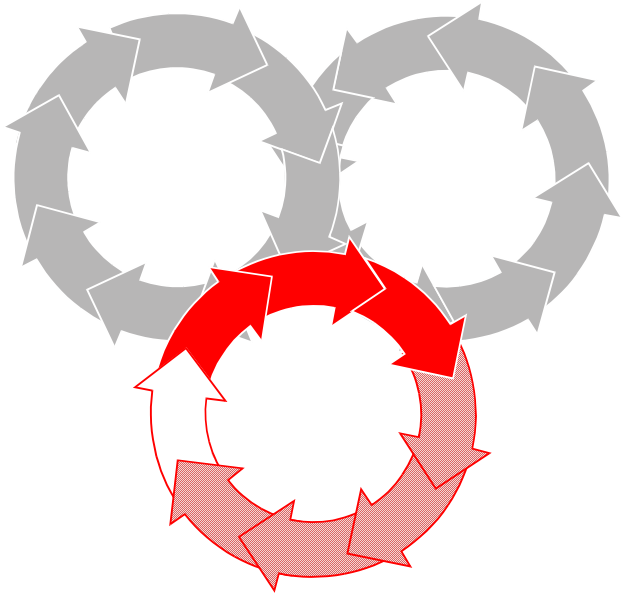
Scoping

Configuration Management

Measurement and Tracking

Technical Management Practice Areas - 1

Technical management practice areas are those necessary for managing the creation and evolution of the core assets and the products.



Technical Management Practice Areas - 2

Configuration Management

Make/Buy/Mine/Commission Analysis

Measurement and Tracking

Process Discipline

Scoping

Technical Planning

Technical Risk Management

Tool Support

Session 3 Contents

Technical Management Practice Areas

Scoping

Configuration Management

Measurement and Tracking

Scoping

Scoping bounds a system or set of systems by defining those behaviors or aspects that are in and those that are out.

All system development involves scoping; there is no system for which everything is in.

In conventional development, scoping is usually done informally (if at all), as a prelude to the requirements engineering activity.

Scoping: Aspects Peculiar to Product Lines - 1

The overall goal is to define which products are in the product line and which are out.

The scope represents the organization's best prediction about what products it will be asked to build in the foreseeable future.

The goal is to draw the boundary, so the product line is profitable.

The following things drive the scope definition:

- prevailing or predicted market drivers
- the nature of competing efforts
- the business goals that led to the product line approach
- technology forecasts that identify expected future technologies

Scoping: Aspects Peculiar to Product Lines - 2

Scope definition lets you determine if a proposed new product can be reasonably developed as part of the existing (or planned) product line.

The scope starts out broad and very general.

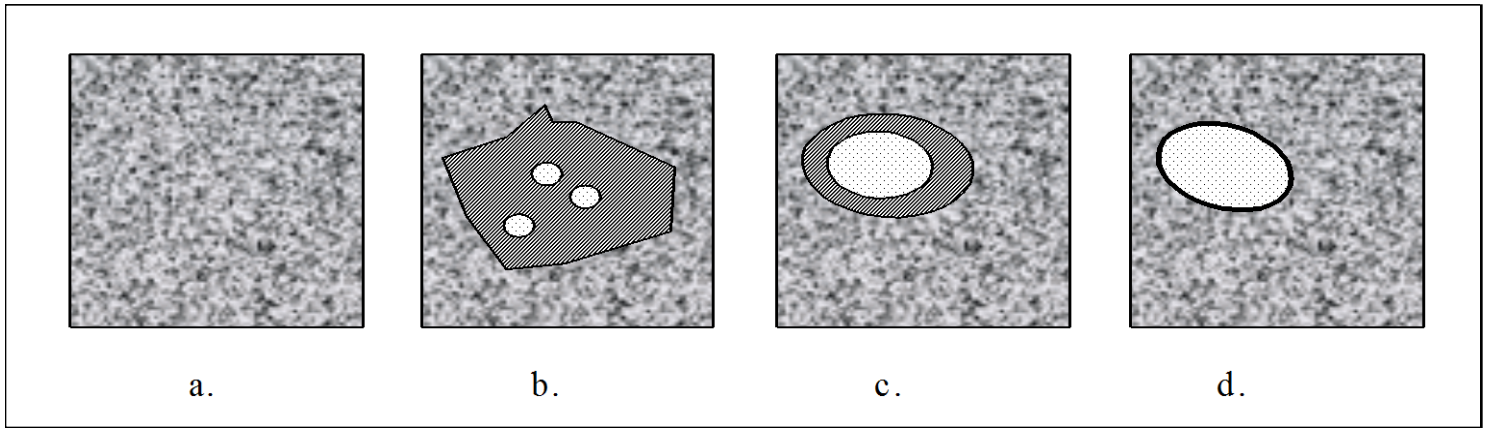
In a product line of Web software

- Browsers are definitely in.
- Aircraft flight simulators are definitely out.
- Email handlers are… well, we aren't sure yet.

The scope grows more detailed as our knowledge increases and the product line matures.

Initially, many possible systems will be "on the cusp," meaning their "in/out" decision must made on a case-by-case basis. That's healthy.

Scoping: Aspects Peculiar to Product Lines - 3



a: space of all possible products
b: early, coarse-grained “in/out” decisions
c: product line scope with a healthy area of indecision
d: product line scope = product line requirements

If many products appear on the cusp over time, you need to reactively adjust the scope.

Proactively Adjusting the Scope

Companies highly skilled at product line engineering routinely adjust their scope to take advantage of opportunities that are in the market.

CelsiusTech

- saw that air defense systems were just a short distance away in scope from ship systems
- was able to enter the air defense market quickly and effectively. Forty percent of its air defense system was complete on day one.

Cummins, Inc.

- developed a software product line for automotive diesel engines
- saw a lucrative underutilized market nearby in industrial diesel engines
- was able to quickly enter and dominate the industrial diesel engine market

Motorola

- developed software product line for one-way pagers
- saw nearby market for two-way pagers and was able to use the same product line architecture for both

Scoping: Example Practices - 1

Apply the *What to Build* pattern (described in Part 3, Session 4 of this course)

Examine existing products yours—and others:

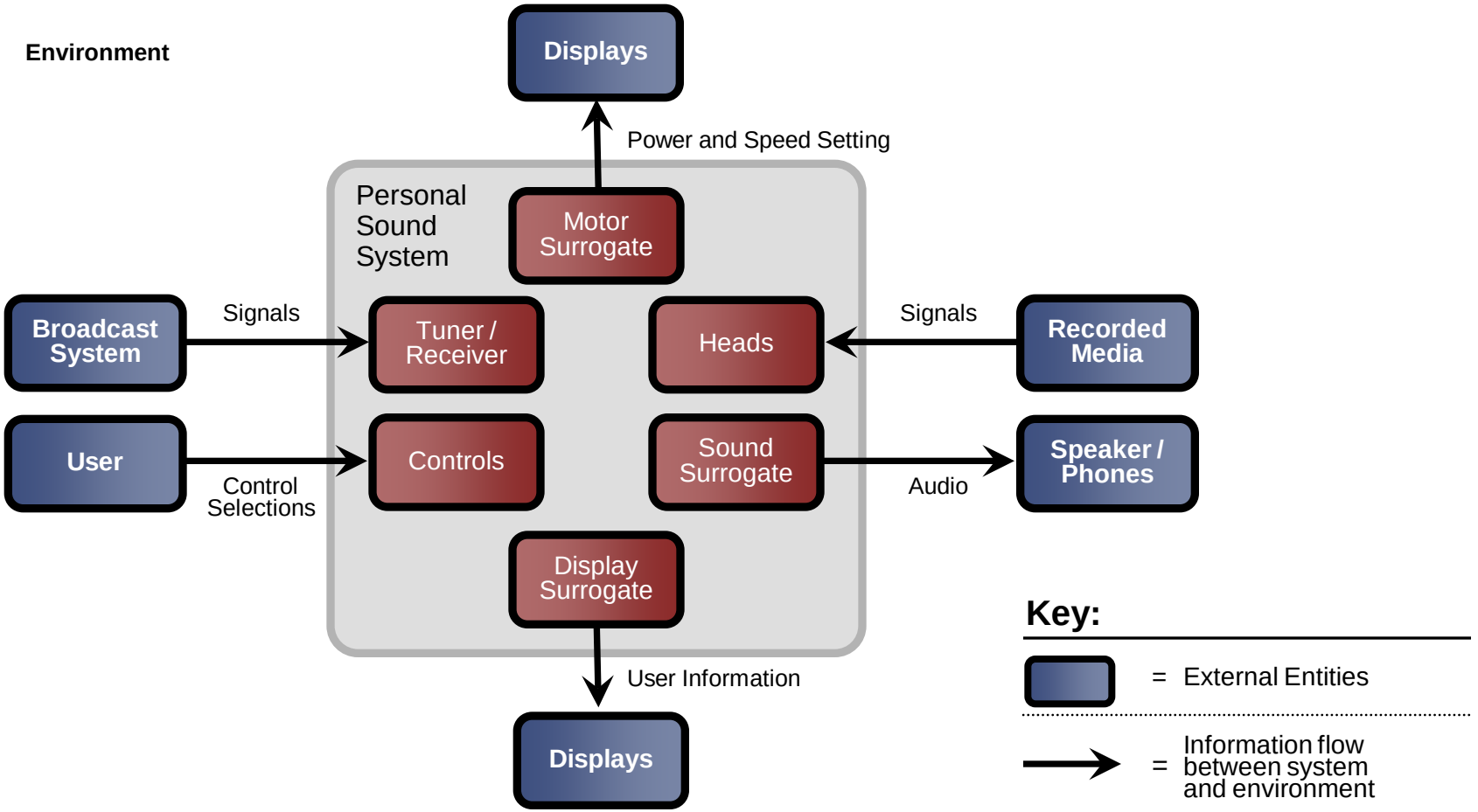
- Identify similar products.
- Gather documentation.
- Conduct surveys.
- Identify relevant capabilities and other factors.
- Determine which elements to include in the product line.

Draw a context diagram:

- depicts the important entities that affect or are affected by the product line

Scoping: Example Practices - 2

Context diagram of a personal sound system



Scoping: Example Practices - 3

Conduct stakeholder workshops:

- Identify business goals to be met by product line.
- Map the product line business goals to the organization's business goals and to users' needs
- Describe current and potential future products that will constitute the product line.
- Identify product and production constraints that may include platforms, standards, protocols, and processes.

Develop an attribute/product matrix:

- sorts, in order of priority, the important attributes by which products in the product line differ

Develop product line scenarios:

- describe user or system interactions with products in the product line

Scoping: Example Practices - 4

Attribute/product matrix for a personal audio system

	Low-Cost Model	Mid-Priced Model	High-End Model
Radio Tuner	analog	digital presets	digital presets
Displays	none	frequency	frequency, graphical equalizer
Audio Control	volume	bass added	full-spectrum equalizer

Scoping: Example Practices - 5

PuLSE-Eco (DeBaud and Schmid) is a method for determining the scope of a product line.

- Product candidates are identified, based on input about the system domain and stakeholders. Candidates include existing, planned, and potential systems. The result is a list of potential characteristics for products in the product line.
- Products and characteristics are combined into a product map, a kind of product/attribute matrix.
- Evaluation functions are created from stakeholder and business goals. These functions will enable the prediction of the costs and benefits of imbuing a particular product with a particular characteristic (such as a feature).
- Potential products are characterized using product maps and the evaluation functions.
- Benefit analysis gathers the characteristic and evaluation information and determines the scope of the product line.

Scoping: Practice Risks

The major risk associated with product line scoping is that the wrong set of products will be targeted.

- The scope is too big or too small.
- The scope includes the wrong products.
- Essential stakeholders don't participate in defining the scope.

These risks will lead to a chronic inability to respond to the needs of the market and the premature breakdown of the product line.

Scoping: For Further Reading

[Cohen 1998a]

[DeBaud 1999a]

[Fritsch 2004a]

[Jandourek 1996a]

[Robertson 1998a]

[Schmid 2001a]

[Withey 1996a]

Discussion

In what ways does a scope definition resemble or differ from a product line requirements specification?

Session 3 Contents

Technical Management Practice Areas

Scoping

Configuration Management

Measurement and Tracking

Configuration Management - 1

Configuration management (CM) refers to a discipline for evaluating, coordinating, approving or disapproving, and implementing changes in artifacts that are used to construct and maintain software systems.

An artifact may be an item of hardware, software, or documentation.

CM enables the management of artifacts from the initial concept through design, implementation, testing, baselining, building, release, and maintenance.

CM is intended to eliminate the confusion and error brought about by the existence of different versions of artifacts.

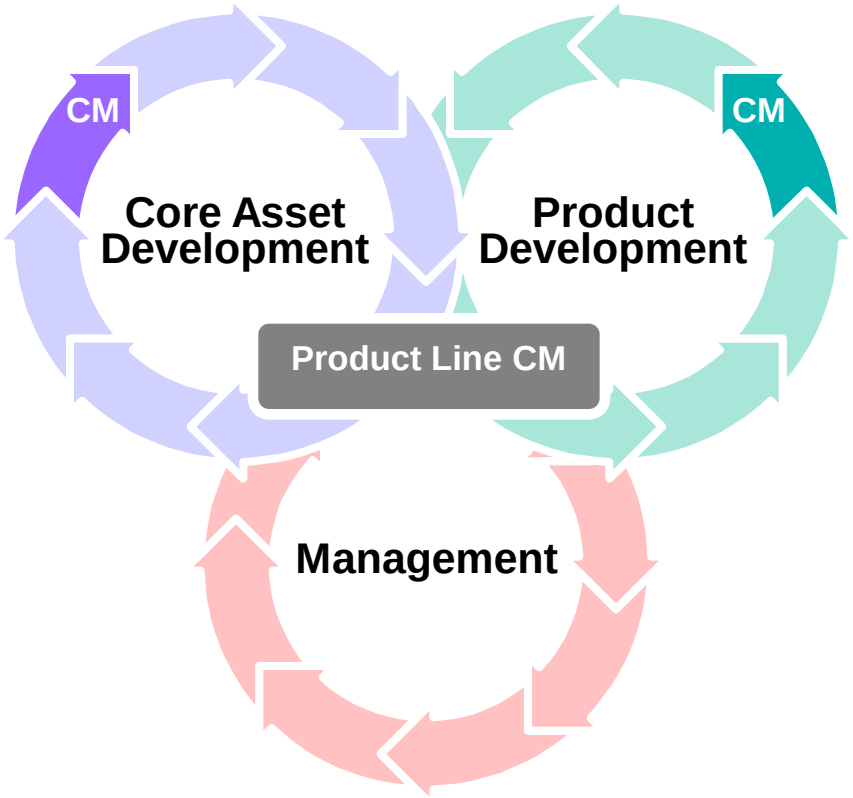
Configuration Management - 2

Successful CM requires a well-defined and institutionalized set of policies and standards that clearly define

- the set of artifacts (configuration items) under the jurisdiction of CM
- how artifacts are named
- how artifacts enter and leave the controlled set
- how an artifact under CM is allowed to change
- how under CM different versions of an artifact are made available and under what conditions each version can be used
- how CM tools are used to enable and enforce CM

These policies and standards are documented in a CM process or plan that informs everyone in the organization just how CM is carried out.

Configuration Management: Aspects Peculiar to Product Lines - 1



CM for product lines is a multidimensional version of CM for single systems.

Configuration Management: Aspects Peculiar to Product Lines - 2

The mission of product line CM may be stated as allowing the rapid reconstruction of any version of any product that may have been built using various versions of the core assets and development/operating environment *plus* various versions of product-specific artifacts.

Whereas “ordinary” CM may be different for each product, product line CM keeps all products under control together.

Product line CM must also support the process of merging results either because new versions of core assets are included in a product or because product-specific results are introduced into the core asset base.

Since introducing changes may affect multiple versions of multiple products, an impact analysis is an essential part of product line CM.

Configuration Management: Aspects Peculiar to Product Lines - 3

Tools, processes, and environments for product line CM must support the following capabilities:

- parallel development: supporting different groups working on the same items for different purposes
- distributed engineering: Organizations that develop product lines might be at more than one site.
- build and release management: Build management enables developers to create a version of a product for testing and/or integration. Release management builds the final customer solution that also includes instantiation of the developing and testing environment. In a product line context, release management includes the release of core assets to product developers.
- change management: Resolutions of change proposals must be communicated, and any resultant changes must be planned, assigned, tracked, and broadcast.
- configuration and workspace management: Configuration and workspace management specifies what a configuration is.

Configuration Management: Example Practices - 1

IEEE/ANSI standard for CM plans

- detailed IEEE/ANSI CM standard that contains a comprehensive outline for a CM plan and several fully worked out examples of CM plans for different kinds of systems and organizations. One sample plan is the “Software Configuration Management Plan for a Product Line System.”

Best practices from practitioners, for example

- SCM patterns
- applying concept of configuration models from knowledge engineering to the problem of product derivation in product lines

Configuration Management: Example Practices - 2

CMMI steps for CM

- The SEI Capability Maturity Model® Integration for System and Software Engineering lists the following practices as instrumental for a CM capability in an organization:
  - Identify the configuration items, components, and related work products that will be placed under CM.
  - Establish and maintain a CM and change management system for controlling work products.
  - Create or release baselines for internal use and for delivery to the customer.
  - Track change requests for the configuration items.
  - Control changes in the content of configuration items.
  - Establish and maintain records describing configuration items.
  - Perform configuration audits to maintain the integrity of the configuration baselines.

Configuration Management: Practice Risks

Without an adequate adhered-to CM process in place, developers will not be able to build and deploy products correctly, let alone recreate versions of products produced in the past.

Inadequate CM control can result from the following:

- an insufficiently robust process
- CM occurring too late
- multiple core asset evolution paths
- unenforced CM practices
- insufficiently robust tool support
- CM manager not understanding the tool capabilities or special needs of product line CM

Configuration Management: For Further Reading

[Burrows 2005a]

[Crossroads 2005a]

[IEEE 1987a]

[Krueger 2002a]

[Leon 2005a]

Discussion

What is the relationship between the CM tool and the CM process?

Session 3 Contents

Technical Management Practice Areas

Scoping

Configuration Management

Measurement and Tracking

Measurement and Tracking

The purpose of measurement is to

- support project tracking
- guide decision making
- determine whether goals are being met over time
- provide management with a justification for effort

Measurement-based tracking is based on

- defining and refining project goals that measurement helps track
- identifying success criteria and indicators for each goal
- defining appropriate measures
- developing a plan to operationalize and verify the measures

The measurement activity comprises two phases:

1. Initiation Phase
2. Performance Phase

Measurement and Tracking: Aspects Peculiar to Product Lines - 1

Practices largely mirror those for single system efforts.

But unlike single-system projects, data collection must provide information from three perspectives:

- core asset development
- product development
- management of the overall product line

Measurement and Tracking: Aspects Peculiar to Product Lines - 2

Stakeholder	Tracking Concern
Product line manager or product line management oversight group	Efficiency, effectiveness, progress against goals, and satisfaction of production constraints of the overall product line effort
Managers of core asset development	Quality and usefulness of the core assets Productivity of those producing core assets
Individual product managers	Quality of the products Efficiency of product developers

Measurement and Tracking: Aspects Peculiar to Product Lines - 3

Measures required to track the progress of the overall product line effort are mostly aggregated from the measures required to track progress of its constituent core asset and product efforts.

For example:

Product line goal: Increased Profitability of Product Development

Tracked by

- Measures reported while component core assets are assembled according to the production plan
- Cost measures associated with the development and evolution of the core assets

Measurement and Tracking: Aspects Peculiar to Product Lines - 4

Measures for effectiveness indicate	Which helps to determine
which core assets are used by product efforts and how often	whether the available core assets are useful
how many bugs are found in core assets by the product developers	the quality of the core assets
how much product efforts expend in finding, tailoring, and integrating assets	needed improvements in supporting infrastructure
where product efforts spend time otherwise	opportunities for future core asset or infrastructure work

Measurement and Tracking: Example Practices - 1

Goal-driven software measurement (GDSM)

1. Define the goals.
2. Refine the goals using clarifying questions.
3. Define subgoals for the relevant stakeholder.
4. Operationalize the goal statements.
5. Determine the success criteria for the stakeholder.
6. Determine the success, progress, and analysis indicators associated with those criteria.
7. Define the organizational strategies and activities for achieving their goals.
8. Determine the measures and data elements needed by all identified indicators.
9. Assess the current infrastructure and identify actions needed to implement the measures.
10. Develop a measurement plan that addresses actions to be taken and how measurements will be verified.

Measurement and Tracking: Example Practices - 2

Choose product line measures based on indicators that are of interest to a product line manager, a core asset development manager, and a product development manager.

Collect data using data collection techniques appropriate to the associated types of measures. Collection techniques (which can be manual or automated) include

- direct measurement of observable attributes of a process or product
- indirect measurement of objective attributes
- surveys for measuring subjective attributes
- derivation of implicit attribute measures as computations from other measures

Use measure-based reuse models.

Measurement and Tracking: Practice Risks

A poor measurement and tracking program will

- be a waste of time and resources
- have an opportunity cost
- cause resentment and mistrust of future measurement efforts
- fail to inform management accurately
- potentially result in bad management decisions

Potential causes of poor measurement and tracking include

- measure mismatch
- goals without measures
- measurement not being aligned
- costly measures
- measures without planned actions
- management by numbers

Measurement and Tracking: For Further Reading

[Geppert 2003a]

[Park 1996a]

[Poulin 1997a]

[Zubrow 2003a]

Discussion

1. Reduced time to market is a key product line goal. How would you measure time to market?

2. Some metrics can backfire. Think of the Dilbert cartoon character who, upon hearing that his company was paying employees who found bugs, went off to his cubicle to “write myself a new car.” Which metrics might encourage the wrong behavior? What would you measure in this case (instead of the “number of bugs found”) to fix the situation?

Session 3 Summary

Technical management practice areas are those necessary for managing the creation and evolution of the core assets and the products.

There are eight technical management practice areas.

The product line scope defines what products are targeted by the product line.

CM is more complex for product lines than for single-system development.

The measurement and tracking activity for product lines differs from that for single-system development in that it must provide information about the three essential product line activities.

Carnegie Mellon University

Software Engineering Institute

Part 2: Software Product Line Practice Areas
Session 4: Organizational Management Practice Areas

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 4 Objectives

This session will

- introduce participants to the organizational management practice areas, in general
- acquaint participants with the following specific organizational management practice areas:
  - Launching and Institutionalizing
  - Structuring the Organization
  - Building a Business Case
  - Funding

Session 2 Contents

Organizational Management Practice Areas

Launching and Institutionalizing

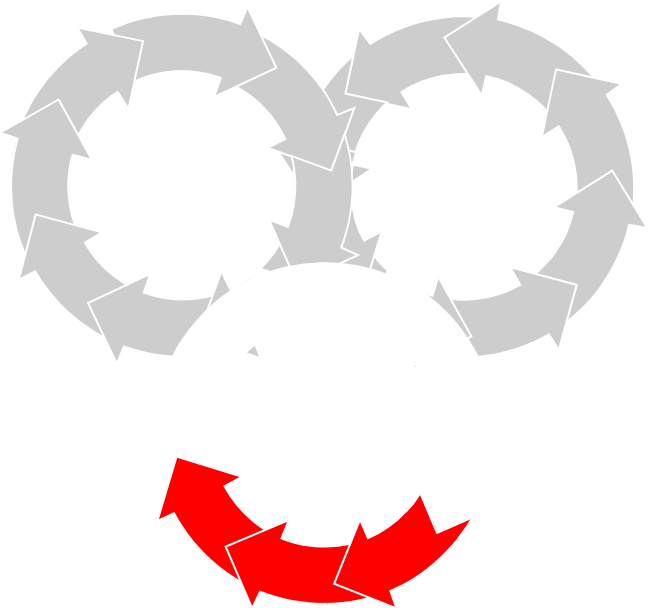
Structuring the Organization

Building a Business Case

Funding

Organizational Management Practice Areas - 1

Organizational management practice areas are those necessary for orchestrating the entire software product line effort.



Organizational Management Practice Areas - 2

Building a Business Case
Customer Interface Management
Developing an Acquisition Strategy
Funding
Launching and Institutionalizing
Market Analysis
Operations
Organizational Planning
Organizational Risk Management
Structuring the Organization
Technology Forecasting
Training

Session 2 Contents

Organizational Management Practice Areas

Launching and Institutionalizing

Structuring the Organization

Building a Business Case

Funding

Launching and Institutionalizing

Launching and institutionalizing is about organizational change.

Change projects

- help organizations adopt a new technology or new way of doing business
- are highly context dependent; there is no invariant recipe.
- take into consideration the human aspects of change

Organizational change involves

- assessing the current state
- identifying the desired state
- bridging the gulf

Launching and Institutionalizing: Aspects Peculiar to Product Lines - 1

The change being launched and institutionalized is of course the software product line approach.

Launching and institutionalizing a product line involves technology and business change.

Launching a product line involves the judicious and timely adoption of product line practices.

- The goal is to have an operational product line.
- An adoption plan shows how all parts of the organization adopt product line capabilities, perhaps in a highly staged manner.

Launching and Institutionalizing: Aspects Peculiar to Product Lines - 2

Institutionalizing a product line requires that the organization consistently use product line practices to achieve its business goals.

Product lines become community practice.

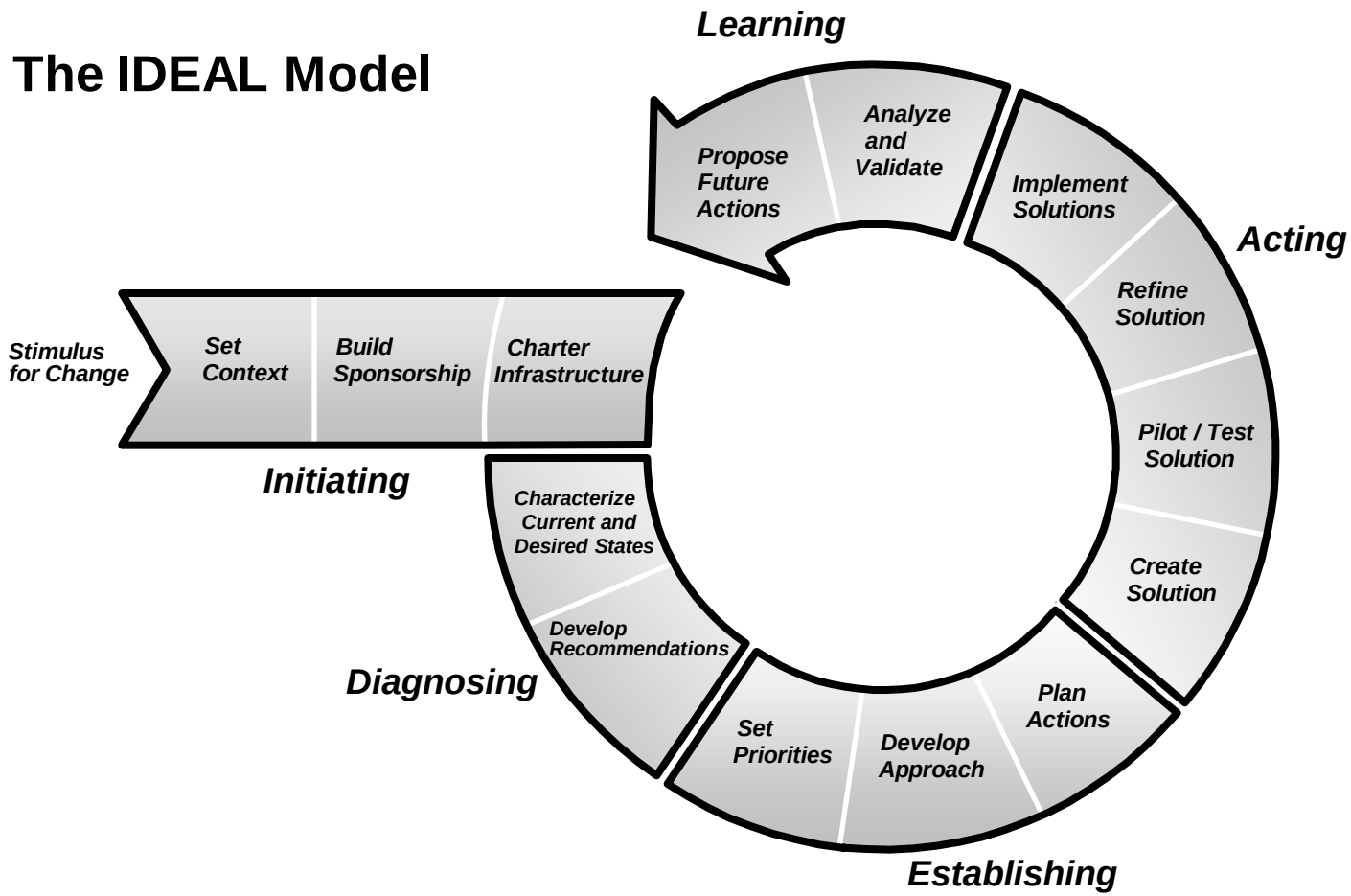
In short, launching and institutionalizing involves putting all of the practice areas into place in an appropriate manner.

Launching and Institutionalizing: Example Practices - 1

The SEI Adoption Factory pattern (described in Part 3, Session 6 of this course) can serve as a generic roadmap for product line adoption.

Use the SEI IDEAL model or the IDEAL model and the Adoption Factory pattern together.

Launching and Institutionalizing: Example Practices - 2



[McFeeley 96]

Launching and Institutionalizing: Example Practices - 3

Pilot projects can be an important way to reduce risk, learn more, and build advocacy. The criteria for choosing a pilot include

- scope: The pilot should be done in a relatively short time frame with reasonable resources.
- importance and visibility: The organization should care whether the pilot succeeds. But the pilot should not be so important that its failure would be disastrous.
- probability of success: The effort should have a reasonable chance to succeed.
- choice of participants: Participants in the pilot should be advocates (or at least be open-minded).

Use proactive, reactive, and incremental approaches.

Use lightweight approaches first.

Launching and Institutionalizing: Example Practices - 4

Use product line diagnostics:

- SEI Product Line Technical Probe[SM] (PLTP[SM])
- SEI Product Line Quick Look
- Bosch Product Line Potential Analysis [Fritsch 2004a]
- Business, Architecture, Process, Organization (BAPO) evaluation [van der Linden 2004a]

Develop product line goals, objectives, and strategies.

Use process improvement as a basis for launching and institutionalizing.

[SM] Product Line Technical Probe and PLTP are service marks of Carnegie Mellon University.

Launching and Institutionalizing: Practice Risks - 1

An inappropriate launching and institutionalizing strategy can

- result in failure of the product line to meet its business goals
- cause an organization to reject the product line approach

Launching and Institutionalizing: Practice Risks - 2

An inappropriate strategy can be caused by

- the lack of an identifiable champion or having a champion in the wrong position
- approach mismatch
- inadequate management commitment
- insufficient staff commitment
- insufficient bounding or trying to solve everything at once
- inappropriate application
- premature standardization
- missed or delayed standardization opportunities
- insufficient tailoring of standard practices or trying to force-fit practices on the organization
- failure to evolve the approach
- ineffective dissemination of information, such as through inadequate training
- lack of an emerging community

Launching and Institutionalizing: For Further Reading

[Ardis 2000a]

[Boeckle 2002a]

[Bosch 2002a]

[Brassard 2001a]

[Clements 2002c, Ch. 7]

[Clements 2002c, Ch. 8]

[Fritsch 2004a]

[Northrop 2004a]

[Wappler 2000a]

[van der Linden 2004a]

Discussion

1. Successful launching requires cross-organizational buy-in. Which groups in the organization would you target?

2. Where in an organization (yours, perhaps) would you expect to encounter resistance to product line practice adoption? Why? What would you do to counter it?

3. Sometimes successfully launched product lines fail to become institutionalized. Why would that happen?

Session 2 Contents

Organizational Management Practice Areas

Launching and Institutionalizing

Structuring the Organization

Building a Business Case

Funding

Structuring the Organization

An organization’s structure determines which groups are responsible for which activities in a development effort.

Often in traditional structures, a single team has total responsibility for all aspects of a product.

Structuring the Organization: Aspects Peculiar to Product Lines - 1

In a product line organization, roles should be identified that

- determine the production strategy
- determine the product line scope and associated business case and refresh each in a routine and ongoing way
- produce and maintain the product line architecture
- determine the product line and product requirements
- design, produce, and maintain core assets and the associated production plan
- assess core assets for their utility and guide their evolution
- produce products
- determine the processes to be followed, and measure or assure compliance
- maintain the production environment
- forecast new trends and technologies

Structuring the Organization: Aspects Peculiar to Product Lines - 2

For product line organizations, the key question is: Who builds the core assets?

- a separate group: To avoid over-specialization of core assets to the product whose team built them
- a product group(s): To avoid over-generalization and inefficiency; to avoid “beauty but not profit”

Factors to consider:

- size of the effort and the number of products
- new development or mostly legacy-based development
- the funding model
- the high or low cost of tailoring core assets
- the volatility of core assets
- parallel or sequential product development

Structuring the Organization: Aspects Peculiar to Product Lines - 3

Responsibilities of the unit(s) assigned to product development include

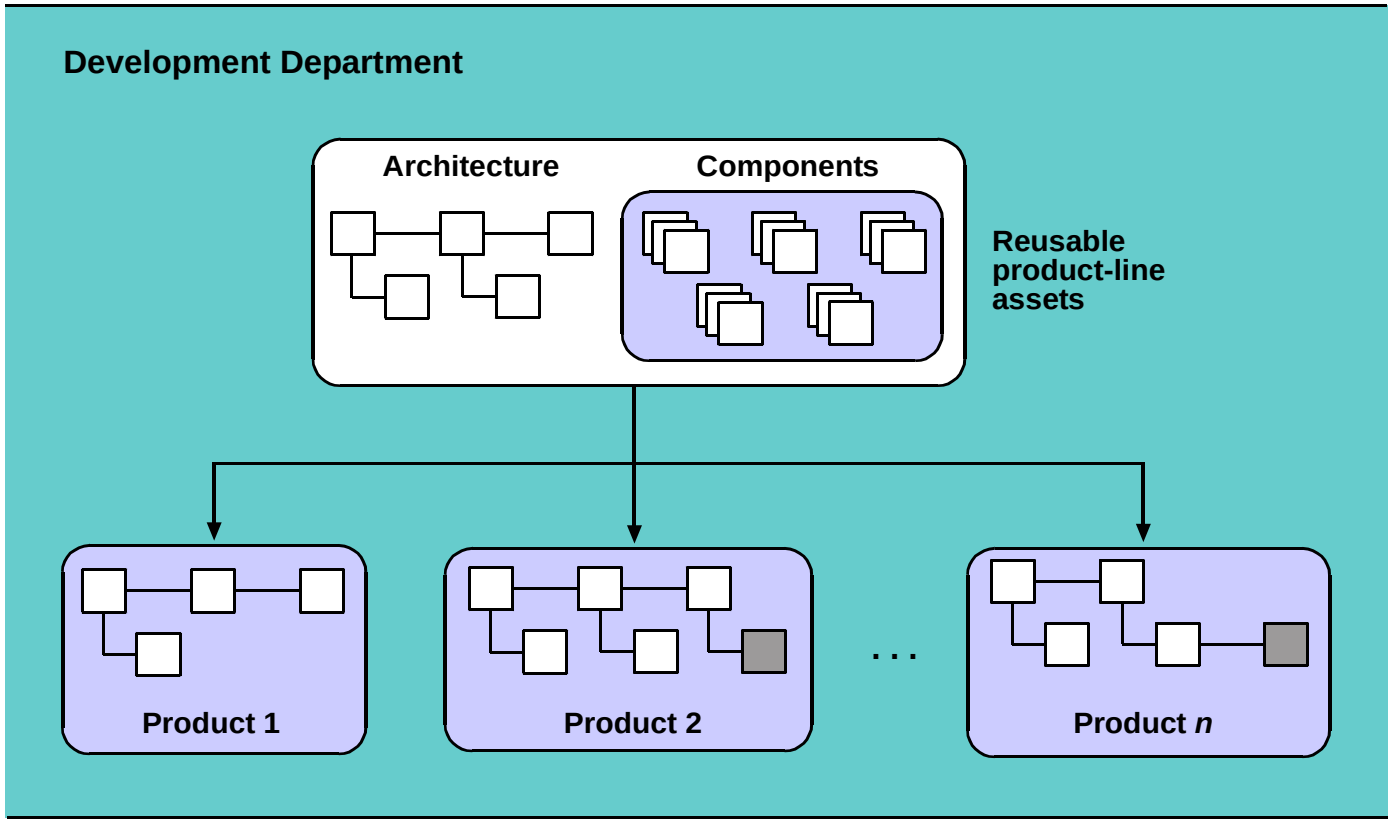
- making sure each new product uses the core asset base according to the production plan
- working with the core asset owners to evolve new capabilities if necessary
- providing feedback to the core asset developers concerning the suitability and quality of the core assets

If product development does not occur in a separate unit or an “open” philosophy is embraced, the product developer role includes improving and evolving core assets.

The organizational structure and the concept of operations need to be closely coordinated and mutually supportive. Both will evolve over time.

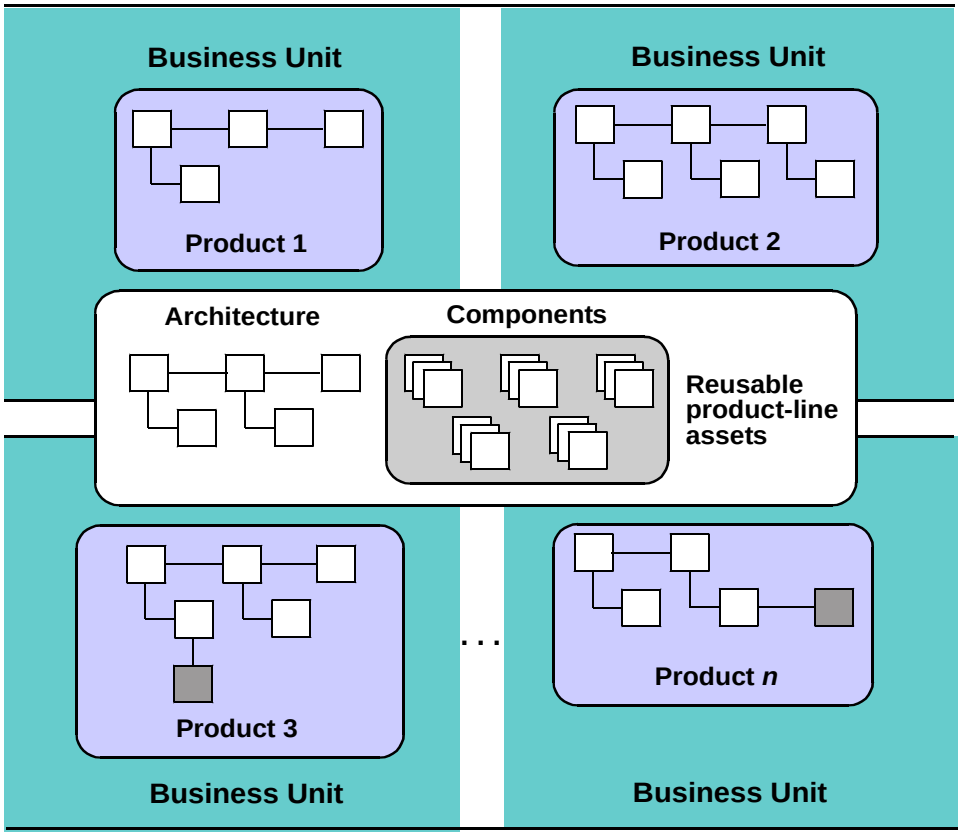
Structuring the Organization: Example Practices - 1

Organizational models from a product line survey [Bosch 2000b]



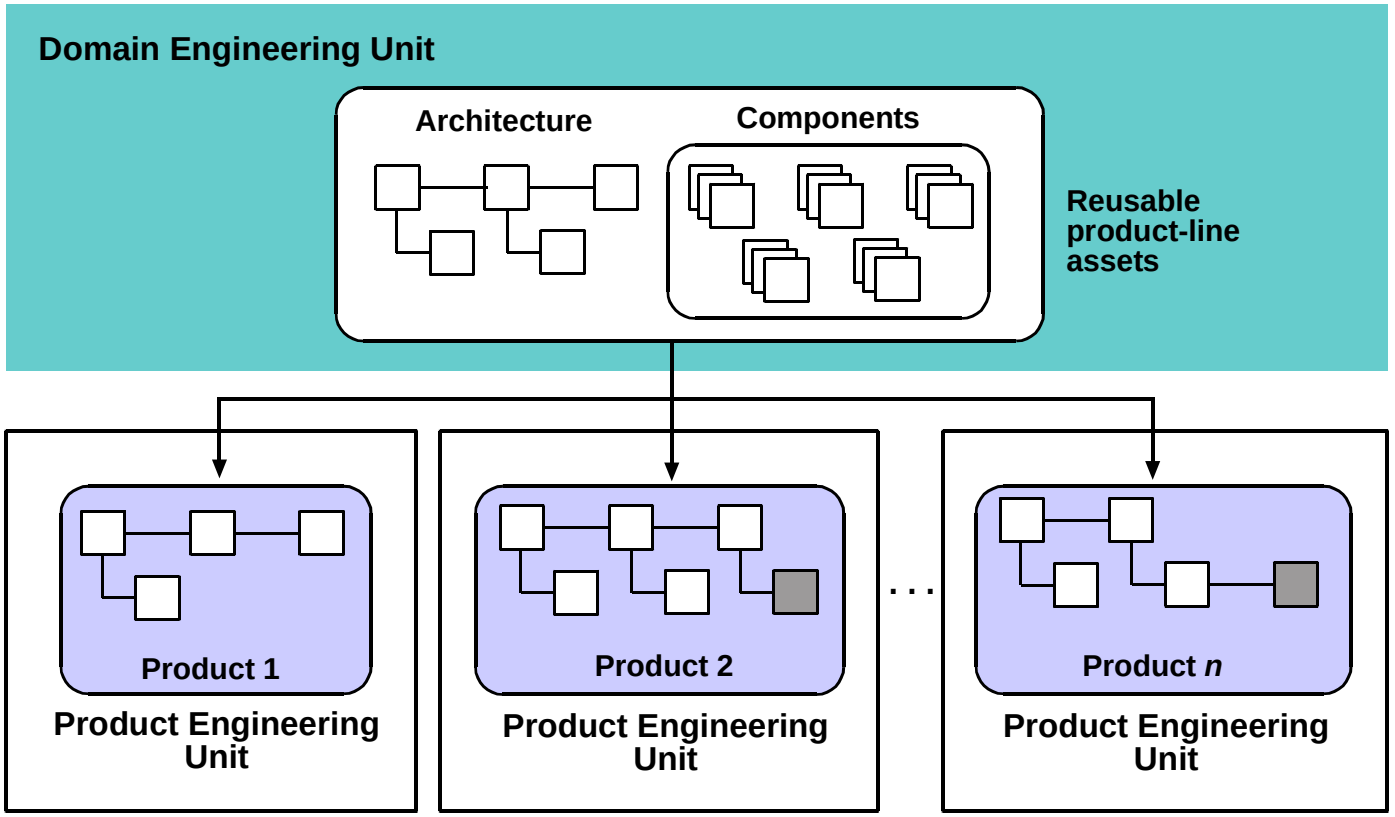
Structuring the Organization: Example Practices - 2

Organizational models [Bosch 2000b] (cont.)



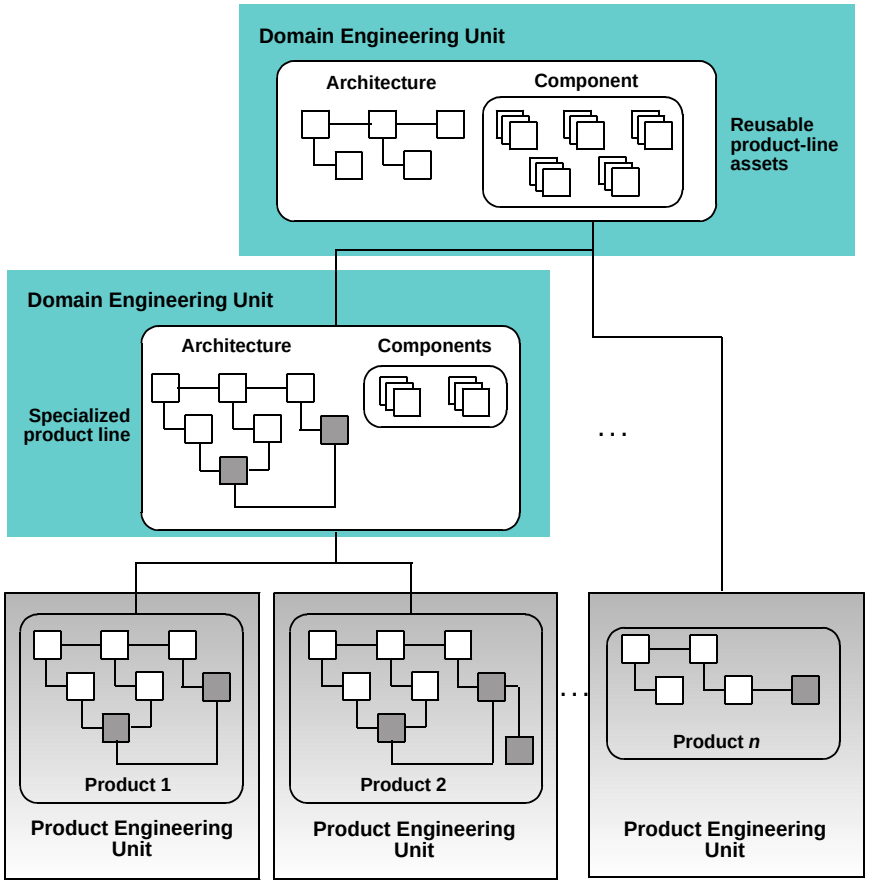
Structuring the Organization: Example Practices - 3

Organizational models [Bosch 2000b] (cont.)



Structuring the Organization: Example Practices - 4

Organizational models [Bosch 2000b] (cont.)



Structuring the Organization: Example Practices - 5

In *Software Reuse* (Chapter 9), Jacobson, Griss, and Jonsson prescribe the following “competence units” that contain “workers with similar competencies and entity object types that these workers are responsible for:”

- requirements capture unit
- design unit
- testing unit
- component engineering unit
- architecture unit
- component support unit

Hybrid approaches: small distributed core group with component representatives

Structuring the Organization: Example Practices - 6

Structure for a distributed or globally dispersed organization includes these options

- virtual core asset development teams with leaders in one location and dedicated core asset developers in other locations
- core asset development teams in one location and product development teams in other locations
- distributed product line management and architecture team with core assets and products built by distributed teams who abide by an “open source philosophy” within their organization

Structuring the Organization: Practice Risks - 1

Choosing a structure that is inappropriate for the given organizational context

Lack of ample feedback and communication mechanisms

One size fits all, for all time

Reorganizing too often

Structuring the Organization: Practice Risks - 2

Ignoring key factors in implementing structural change

- existing organizational stress
- implementation history
- sponsorship
- resistance management
- culture
- change agent skills

Setting up an organizational structure without an accompanying operational concept

Structuring the Organization: For Further Reading

[Bosch 2000b]

[Brownsword 1996a]

[Jacobson 1997a]

Discussion

1. Should there be a single testing unit for both core assets and products, or one for each?

2. Conditions change over time. What mechanisms would you put in place to tell you when it was time to adopt a new organizational structure?

Session 2 Contents

Organizational Management Practice Areas

Launching and Institutionalizing

Structuring the Organization

Building a Business Case

Funding

Building a Business Case - 1

A *business case* is a tool that helps you make business decisions by predicting how they will affect your organization.

A business case is used

- initially, to justify pursuing a new business opportunity or approach
- later, to examine new or alternative angles on the opportunity

A business case addresses the following key questions:

- What specific changes must occur?
- What are the benefits of those changes?
- What are the costs and risks of those changes?
- How do we measure success?

Building a Business Case - 2

The business case documents how closely aligned the opportunity is with established business goals.

The goal of a business case is to provide management with a sufficient understanding of the approach and adequate data to determine if the return on investment (ROI) is sufficient to justify a proposed venture.

A business case should address these tasks [Humphrey 2000a]:

- deciding what to do
- estimating the likely costs and potential risks of all alternatives
- estimating the likely benefits contrasted with the current business practice
- developing a proposal for proceeding
- closing the deal

Building a Business Case: Aspects Peculiar to Product Lines - 1

A business case in a product line context can

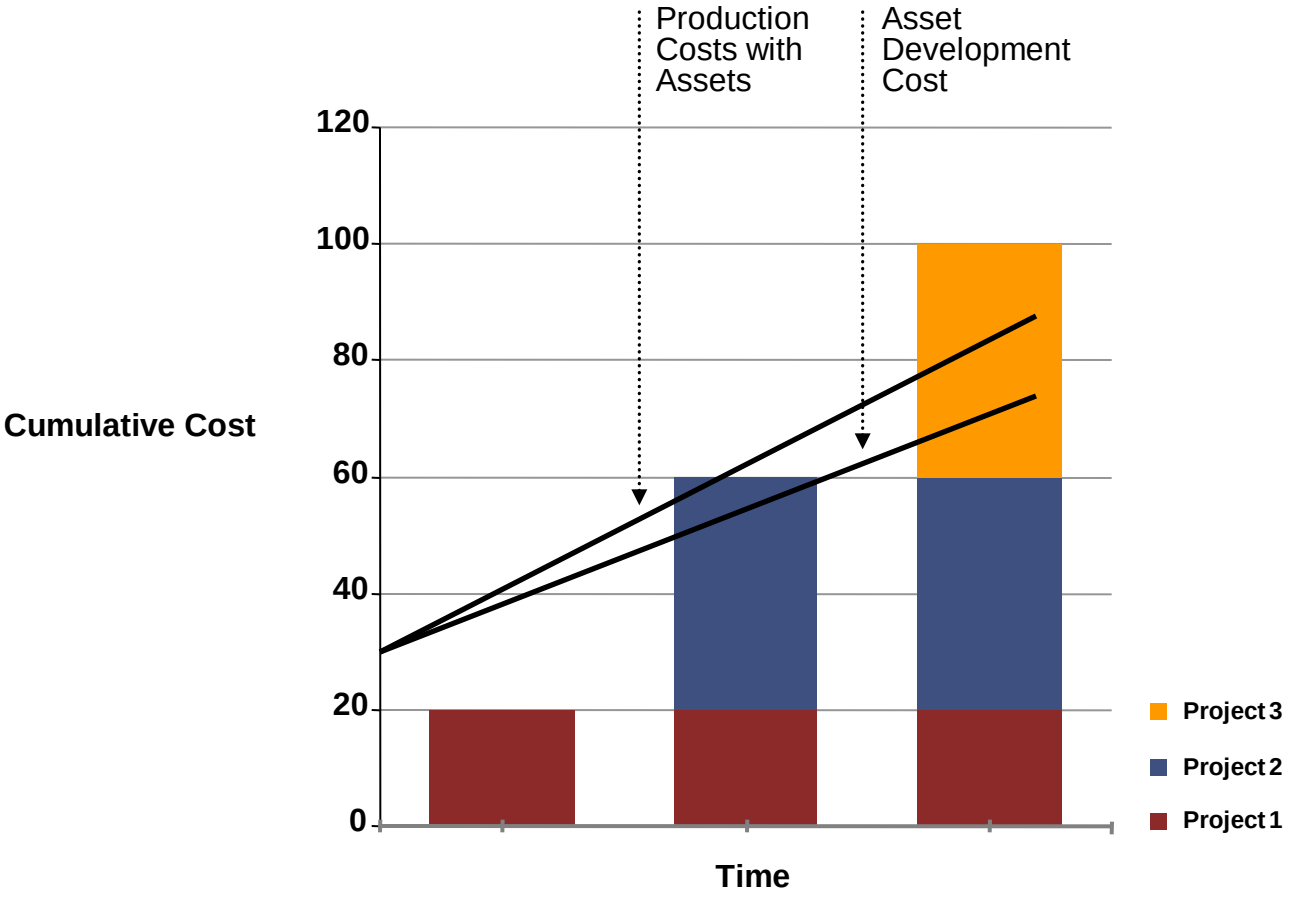
- justify the effort to adopt a product line approach
- help decide which products to include in the product line

The business case includes the product line goals that will, in turn, drive the data to be collected and the measures to be tracked for the product line.

Organizations that adopt a software product line approach should know what they expect to gain and what they will have to change. The business case helps them to predict the payoff and the risk.

The business case reflects the facts and assumptions from domain knowledge, product line scope, market analysis, and current technology trends.

Building a Business Case: Aspects Peculiar to Product Lines - 2



Basic product line economics play into the business case.

Building a Business Case: Aspects Peculiar to Product Lines - 3

The business case for the entire product line is, itself, a valuable core asset that should be documented, maintained, and revisited periodically to make sure that the organization's goals are still being adequately served.

As products in the product line are developed, the role of the business case evolves within the organization to become a more tactical document.

In addition, the business case supports decisions to direct or redirect resources during the product development and evolution phases.

Building a Business Case: Example Practices - 1

Build a "Business case lite:" Every cost and benefit need not be measured, only the major ones. "We won't survive otherwise" is (with a little added clarification/justification) a very compelling business case.

Estimate costs and benefits: Use "standard" product line costs, risks, and benefits as the basis for a business case template. Then, fill it in based on data specific to your company/industry.

Use Models: Cost-estimation or economic models can help to quantify the business case.

- COPLIMO: Constructive Product Line Investment Model, based largely on COCOMO (COnstructive COst MOdel)
- SIMPLE: Structured Intuitive Model For Product Line Economics

Building a Business Case: Example Practices - 2

SEI Structured Intuitive Model For Product Line Economics (SIMPLE)

- Economic models that use the organization's current cost to develop a product as the basis for comparison: C_{prod}
- Product line costs can be grouped into four cost functions:
  - C_{org} – cost of converting current organization to a product line organization
  - C_{cab} – cost of creating core assets
  - C_{unique} – cost of creating the unique portion of each product
  - C_{reuse} – cost of understanding the core assets used in a product
- These functions can be used to define functions that are used to model product development costs in the product line.

Building a Business Case: Example Practices - 3

SIMPLE (cont)

The cost of building the n products using the current approach is

$$n * C_{prod}()$$

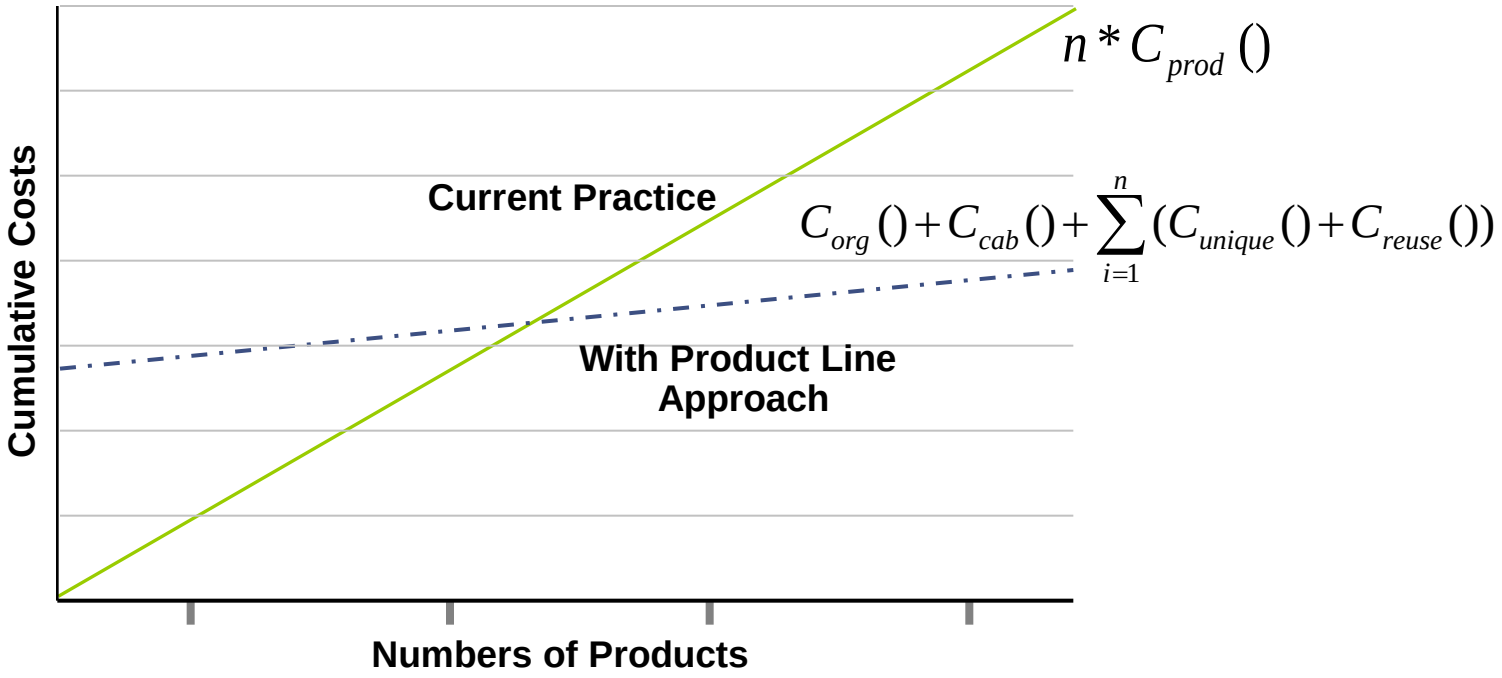
The cost of building the product line of n products is

$$C_{org}() + C_{cab}() + \sum_{i=1}^{n} (C_{unique}() + C_{reuse}())$$

The power of a product line is shifting as much cost as possible out of the summation and into the fixed cost expressions.

Building a Business Case: Example Practices - 4

SIMPLE (cont)



Weiss, D. M. & and Lai, C. T. R.
Software Product-Line Engineering: A Family-Based Software Development Process.
Reading, MA: Addison-Wesley, 1999.

Building a Business Case: Practice Risks

An inadequate business case (or lack of any business case) can set up a product line organization for failure.

An inadequate business case can result from

- insufficient data
- unreliable historical data
- approaches that fail to work across organizational boundaries
- uncertain market conditions
- management indecision
- a shift in organizational goals and needs (trying to hit a moving target)

Building a Business Case: For Further Reading

[Baldwin 2002a]

[Boehm 2003a]

[Clements 2002c, p. 226]

[Cohen 2003a]

[Ganesan 2006a]

[Humphrey 2000a]

[Reifer 1997a]

[SEI 2007h]

[Weiss 1991, pp. 45-49]

Discussion

1. What happens if the predictions in a business case are too optimistic?

2. How does a business case relate to product line scoping?

Session 2 Contents

Organizational Management Practice Areas

Launching and Institutionalizing

Structuring the Organization

Building a Business Case

Funding

Funding - 1

Software development efforts have to be financed; this practice area addresses how.

Funding sources and models vary according to the organizational culture and the nature of the software product being developed.

- Multiple copies of the software product are to be marketed, the organization usually appropriates development funds to a business unit, or the business unit appropriates its own funds.
- New products often get funded initially out of research and development allocations.
- For products made specifically to serve the needs of one customer, the customer usually provides the funds.
- The funding of product maintenance may come from a different source than that of the development financing.

Funding - 2

Whatever the source, somehow the funds are procured to support what it takes to develop and then to evolve the software product.

Good estimates are required so that an adequate amount is allocated, thereby providing a stable funding source through product completion.

Funding: Aspects Peculiar to Product Lines - 1

Investment is required for any organizational change—a move to product lines is no different. It will require funds for

- training
- different processes
- different management practices
- different tools
- establishing the production strategy and production method
- establishing the core asset base
- performing the initial analysis (such as achieving an understanding of the relevant domains, scoping, requirements engineering, architecture definition, and so on)
- establishing a production infrastructure

Funding: Aspects Peculiar to Product Lines - 2

Funding must be sufficient so that the core assets can be of high quality and have the appropriate applicability.

Funding provides ongoing investment to

- keep the core assets current
- update the analysis
- modernize the infrastructure

Funding must be stable and enduring so that the assets can be maintained and the associated product line practices and tools can be supported and improved.

The funding profile depends on the product line approach taken.

The magnitude of the funding required should be defined in the business case for the product line.

Funding: Example Practices - 1

Specific funding strategies include

- product-specific funding (individual customer, for example)
- direct funding from corporate sponsor/program
- product line organization's discretionary funds
- borrowing funds from corporate sources
- first product (project) funds the effort
- multiple projects band together to share costs
- taxing of participating projects
- product-side tax on customers
- fee based on asset usage
- prorated cost recovery

Funding: Example Practices - 2

Funding Strategies \ Activities to be Funded		Planning and Analysis	Product Line Development: Infrastructure Development	Product Line Development: Asset Development	Product Line Sustainment and Evolution	Product Development Sustainment and Evolution
1	Product-specific funding (individual customer, for example)			◒		●
2	Direct funding from corporate sponsor / program	●	◒	◒		
3	Product line organization's discretionary funds	○	●	○	○	
4	Borrowing funds from corporate sources	●	●	●	○	○
5	First product (project) funds effort	◒	◒	◒	○	●
6	Multiple projects banded together to share costs	●	◒	●	◒	○
7	Taxing of participating projects		○	○	●	
8	Product-side tax on customers			○	●	
9	Fee based on core asset usage				●	

Funding: Practice Risks

Inadequate attention to the funding model for a product line will result in a core asset base and products whose owners compete in unhealthy ways for the finite resources available.

A poor funding model can result from any of the following:

- the inflexibility of the organization's fiscal infrastructure
- waning management commitment
- externally imposed fiscal constraints
- a lack of strategic focus
- inadequate funding

Session 4 Summary

Organizational management practice areas are those necessary for orchestrating the entire software product line effort.

There are twelve organizational management practice areas.

Launching and institutionalizing is a special case of organizational change—one that involves technical and business dimensions.

There are choices of organizational structures for product lines; organizational structure and operational concept go hand in hand.

A business case is needed to justify the product line approach and continuously inform the scope.

A funding model for software product lines is both nontrivial and key.

Carnegie Mellon University

Software Engineering Institute

Part 3: Putting the Practice Areas into Action
Session 1: Case Studies: Cummins, Inc.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Course Structure

Course Introduction

Part 1: Software Product Line Fundamentals

Part 2: Software Product Line Practice Areas

Part 3: Putting the Practice Areas into Action

Wrap-Up

Session 1 Objectives

This lecture will acquaint participants with the

- usefulness of product line case studies
- context for the Cummins software product line
- practices that were used to launch and sustain the product line
- management practices employed on the effort
- results achieved by the Cummins effort
- lessons learned during the Cummins effort
- current Cummins status

Session 1 Contents

Case Studies

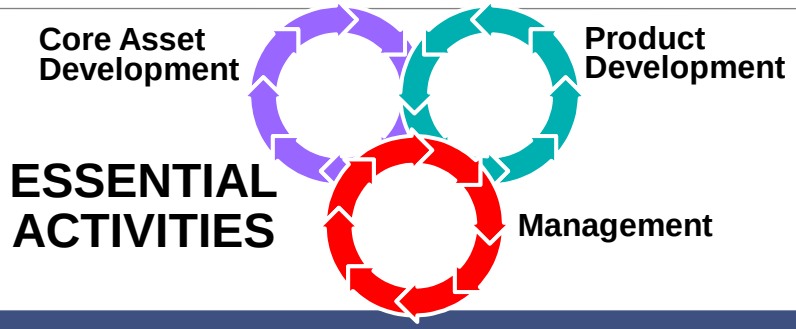
Cummins Background

Launching the Cummins Product Line

Practice Areas of Particular Interest

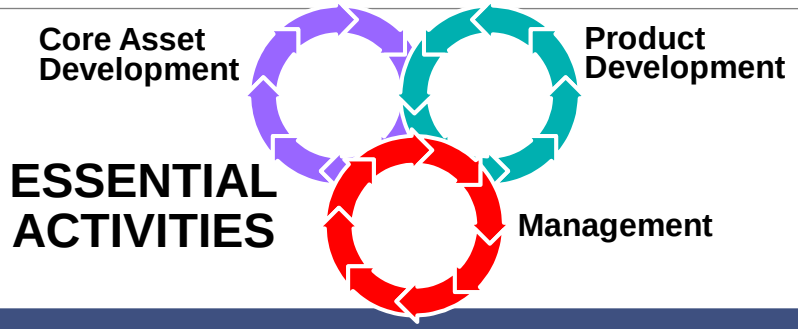
Cummins' Results

Review of Framework Version 5.0



PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management
Architecture Definition	Configuration Management	Building a Business Case
Architecture Evaluation	Make/Buy/Mine/Commission Analysis	Customer Interface Management
Component Development	Measurement and Tracking	Developing an Acquisition Strategy
Mining Existing Assets	Process Discipline	Funding
Requirements Engineering	Scoping	Launching and Institutionalizing
Software System Integration	Technical Planning	Market Analysis
Testing	Technical Risk Management	Operations
Understanding Relevant Domains	Tool Support	Organizational Planning
Using Externally Available Software		Organizational Risk Management
		Structuring the Organization
		Technology Forecasting
		Training

Review of Framework Version 5.0



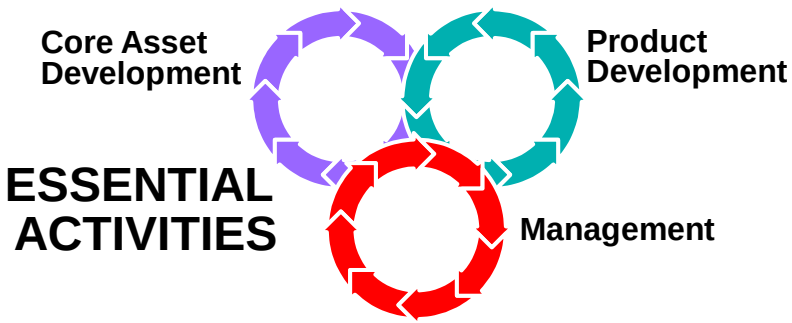
PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management
Architecture Definition	Configuration Management	**Building a Business Case**
Architecture Evaluation	Make/Buy/Mine/Commission Analysis	**Customer Interface Management**
Component Development	***Measurement and Tracking***	**Developing an Acquisition Strategy**
Mining Existing Assets	***Process Discipline***	Funding
Requirements Engineering	Scoping	Launching and Institutionalizing
Software System Integration	**Technical Planning**	Market Analysis
Testing	Technical Risk Management	**Operations**
Understanding Relevant Domains	**Tool Support**	Organizational Planning
Using Externally Available Software	**Key to differences from V4:**	Organizational Risk Management
	New Name and Substantial Change	**Structuring the Organization**
	Substantial Change	Technology Forecasting
		Training

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!

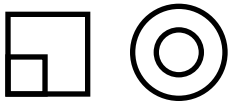
Help to Make It Happen



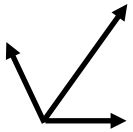
PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management

GUIDANCE

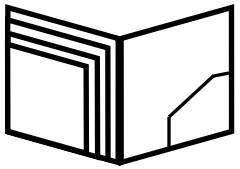
Case Studies



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

Session 1 Contents

Case Studies

Cummins Background

Launching the Cummins Product Line

Practice Areas of Particular Interest

Cummins' Results

Cummins, Inc.

World's largest manufacturer of commercial diesel engines above 50 hp

25,000 employees

350 controls and electronics engineers

$7B annual sales

Domain with Complex Variation

Today's diesel engines are driven by software.

- Micro-control of ignition timing is needed to achieve an optimum mix of power, economy, and emissions.
- Conditions change dynamically as a function of road incline, temperature, load, and so forth.
- Must also respond to statutory regulations that often change.
- Reliability is critical! Multimillion dollar fleets can be put out of commission by a single bug.
- Software is about 130KSLOC – C, assembler, microcode.
- They have different sensors, platforms, and requirements.

In 1993, Cummins Had a Problem

The market was demanding new products:

- Six engine projects were underway.
- Another 12 were planned.

Each project had complete control over its development process, architecture, and choice of language. Two were trying to use object-oriented methods.

Ron Temple (VP in charge) realized that he would need another 40 engineers to handle the new projects, which was out of the question.

Temple realized this was no way to do business. In May 1994, he halted all the projects.

Session 1 Contents

Case Studies

Cummins Background

Launching the Cummins Product Line

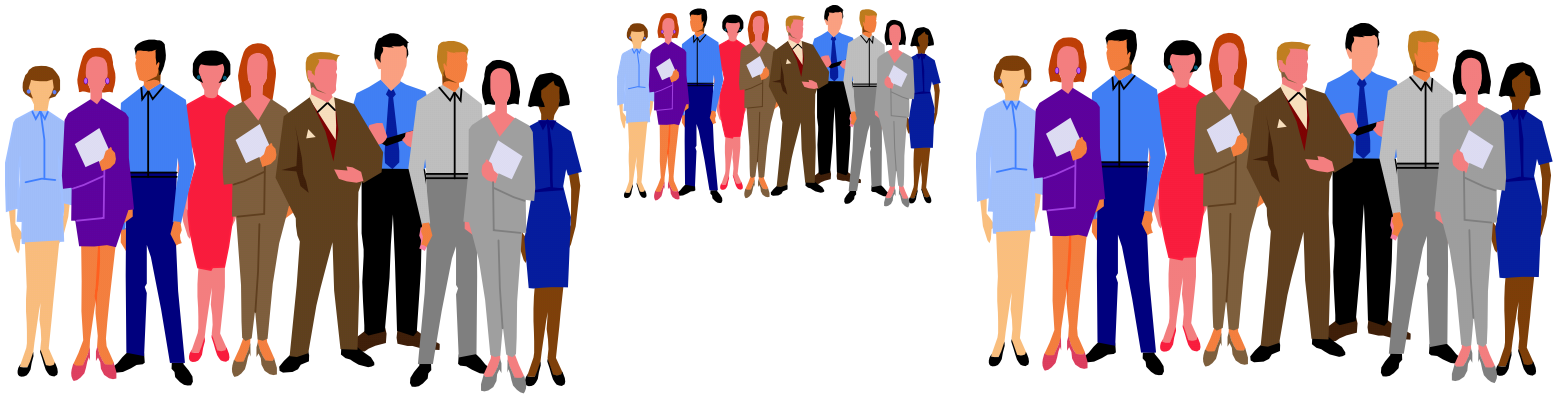
Practice Areas of Particular Interest

Cummins' Results

First Steps

Temple split up the staff on the leading project:

- One half built core assets – generic software, documentation, and other assets that every product could use.
- The other half became a pilot project for using the core assets to turn out a product.
- The staff on the remaining projects trained or served as reviewers.



Building a Business Case

At the same time, Temple made the business case to his management, which in this case was direct and simple:

> *We simply cannot continue to do business one project at a time.*

Projected gains cited in the business plan included

- a cycle-time reduction—hope to shave a year off
- the ability to turn out more products without having to hire 40 engineers
- product quality improvements
- a risk reduction
- higher customer satisfaction

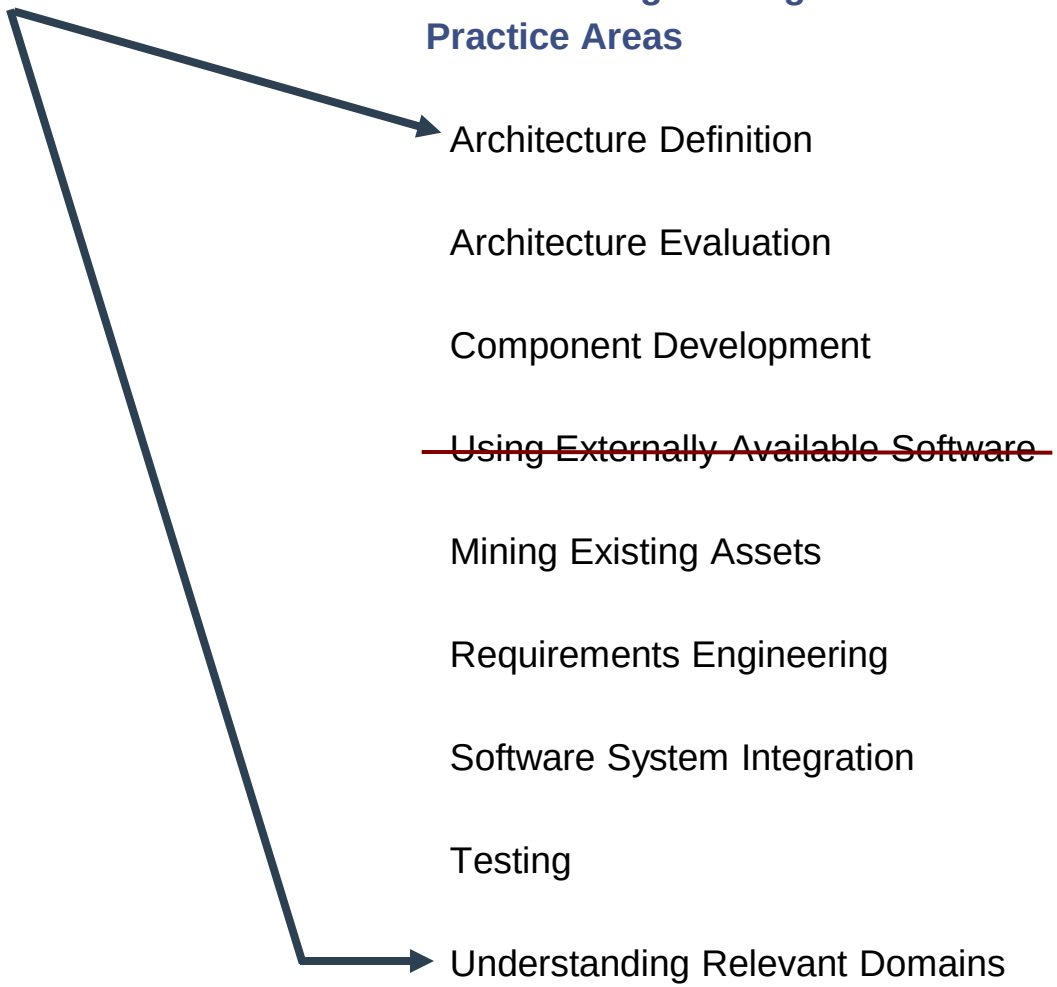
Launching Steps

Cummins saw rolling out the product line as a six-step job:

1. Establish and implement the core asset implementation plan. Select and task the core group.
2. Select legacy software for the starting point.
3. Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
4. Establish the workflow and tool set to be used. Establish detailed project plans.
5. Pilot the first application. Improve the workflow and tool set.
6. Roll out subsequent products, and continue to improve the workflow and tool set.

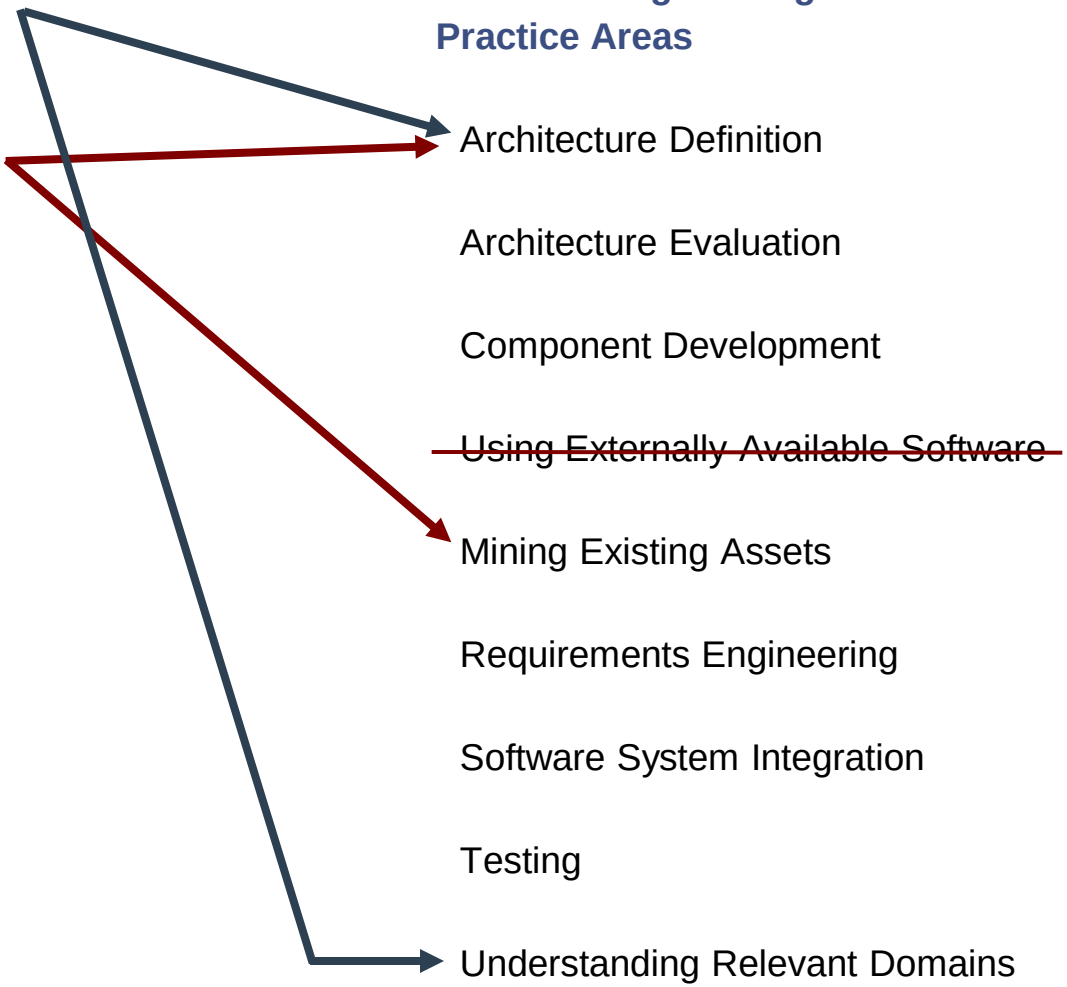
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.



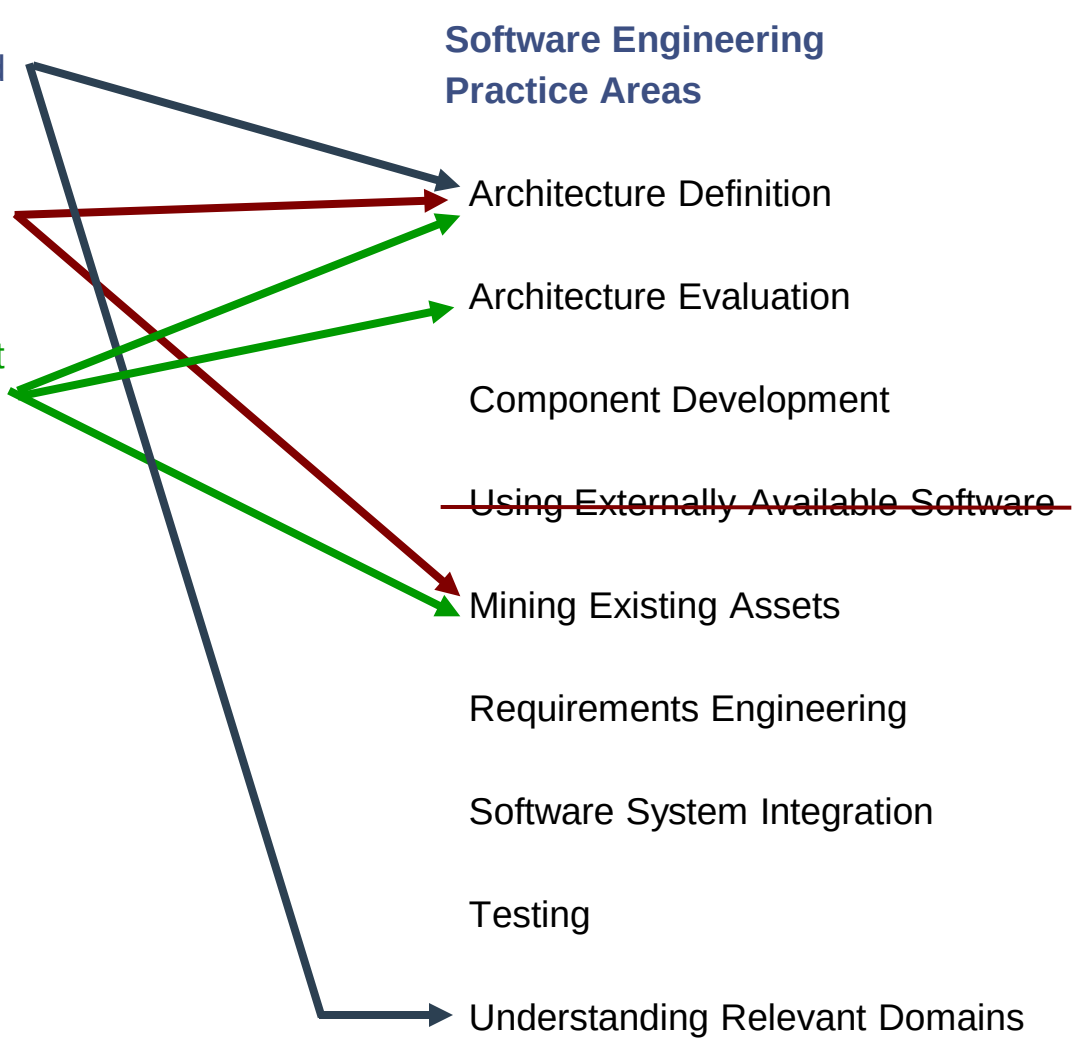
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.



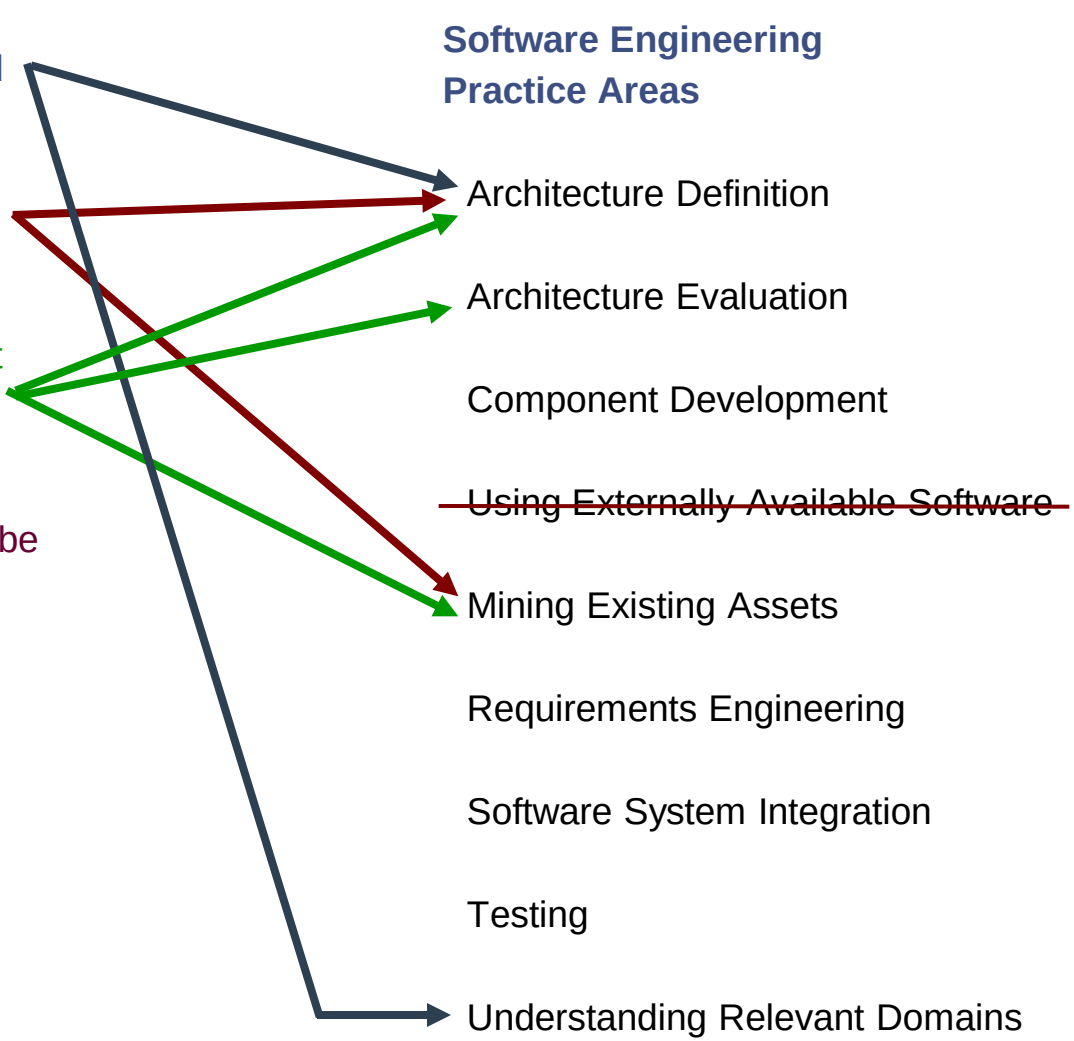
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.



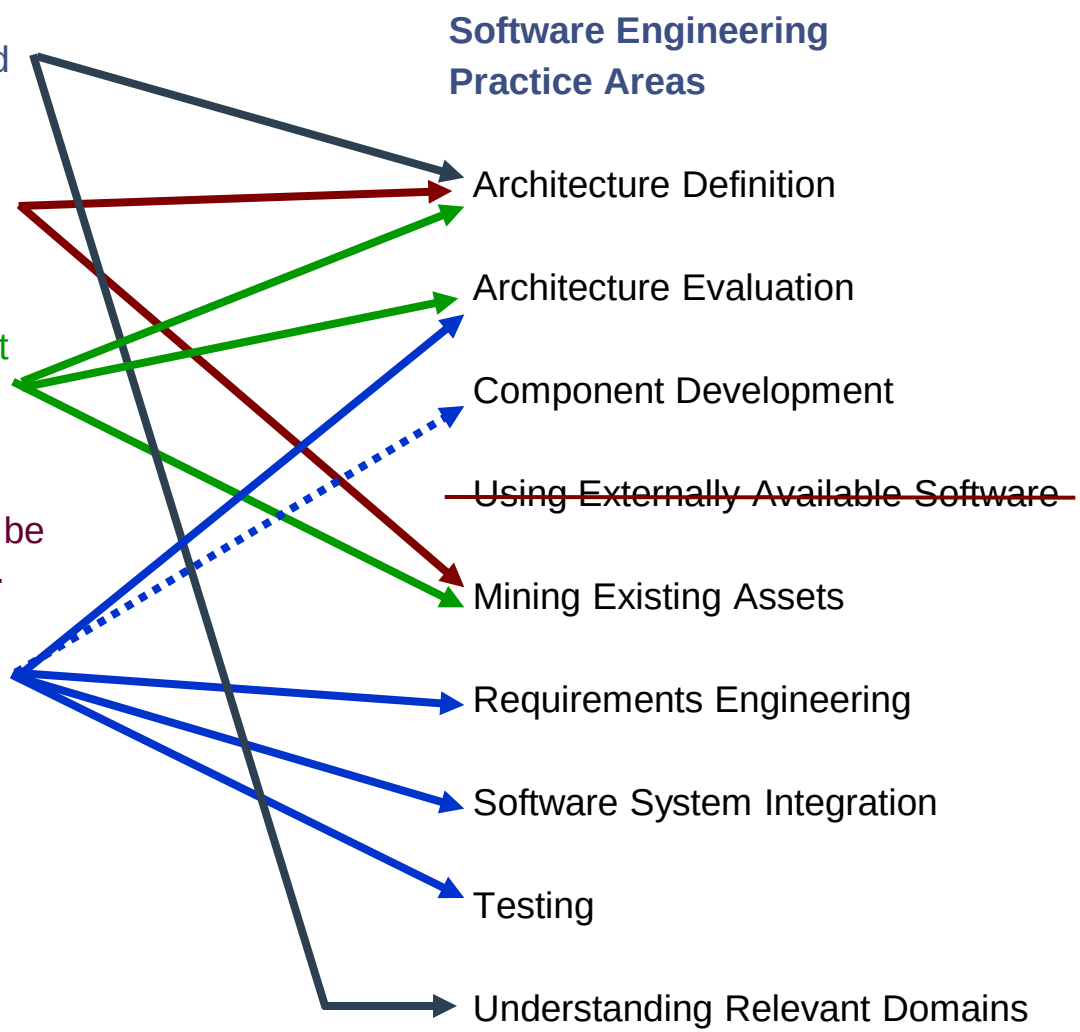
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.



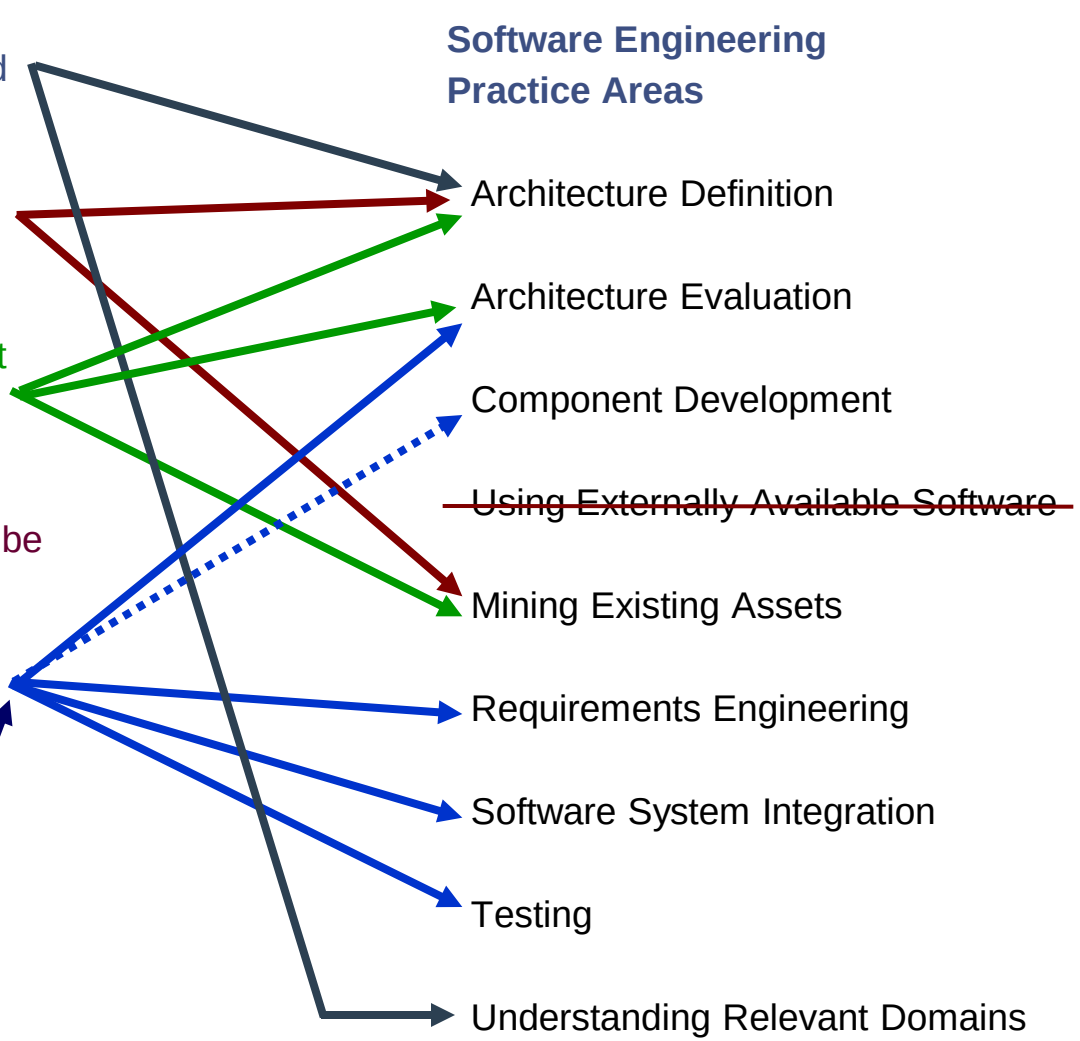
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.
- Pilot the first application. Improve the workflow and tool set.



Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.
- Pilot the first application. Improve the workflow and tool set.
- Roll out subsequent products, and continue to improve the workflow and tool set.

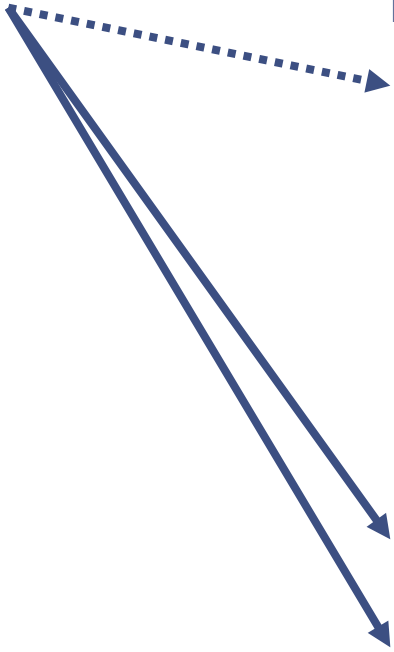


Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.

Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.



Technical Management Practice Areas

Configuration Management

Measurement and Tracking

~~Make/Buy/Mine/Commission Analysis~~

Process Discipline

Scoping

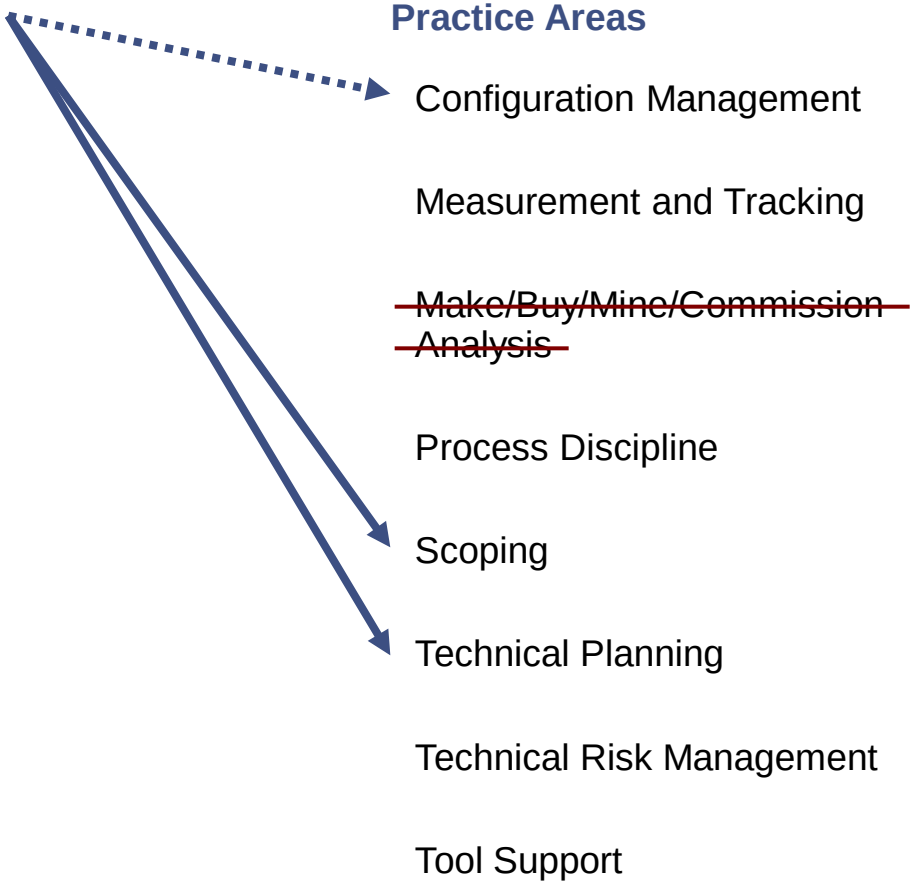
Technical Planning

Technical Risk Management

Tool Support

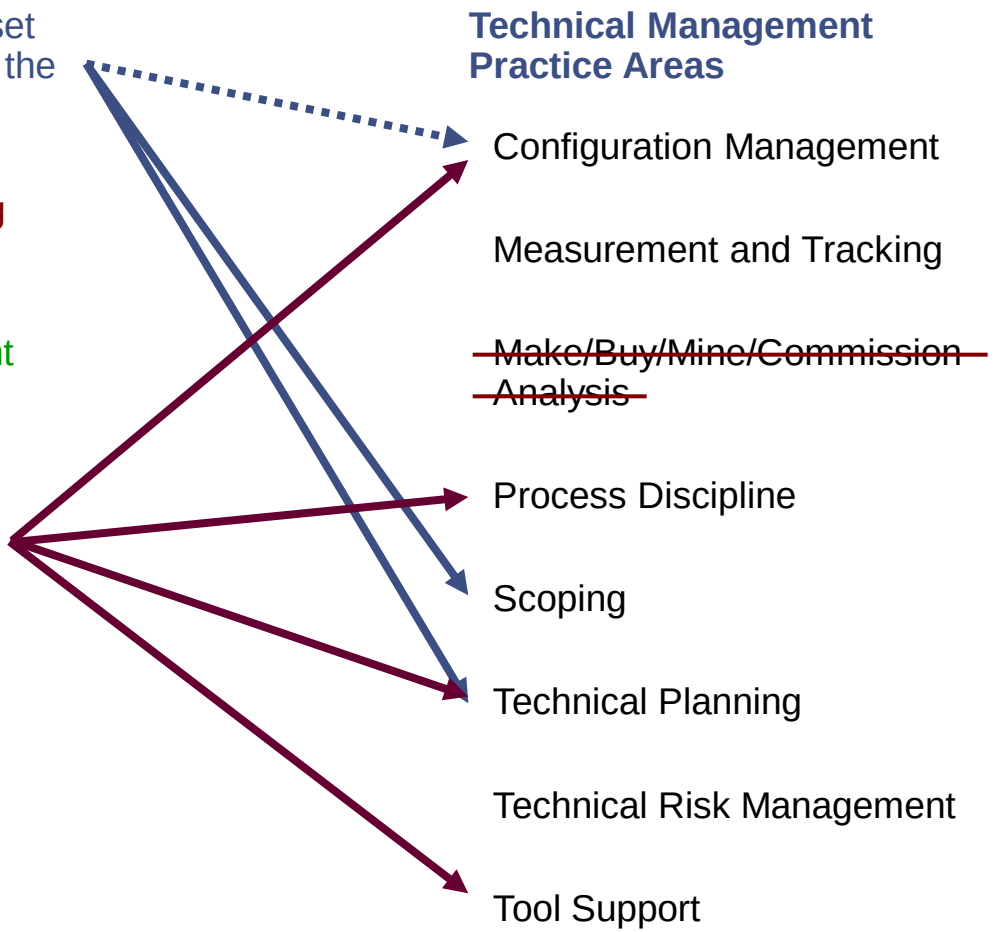
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.



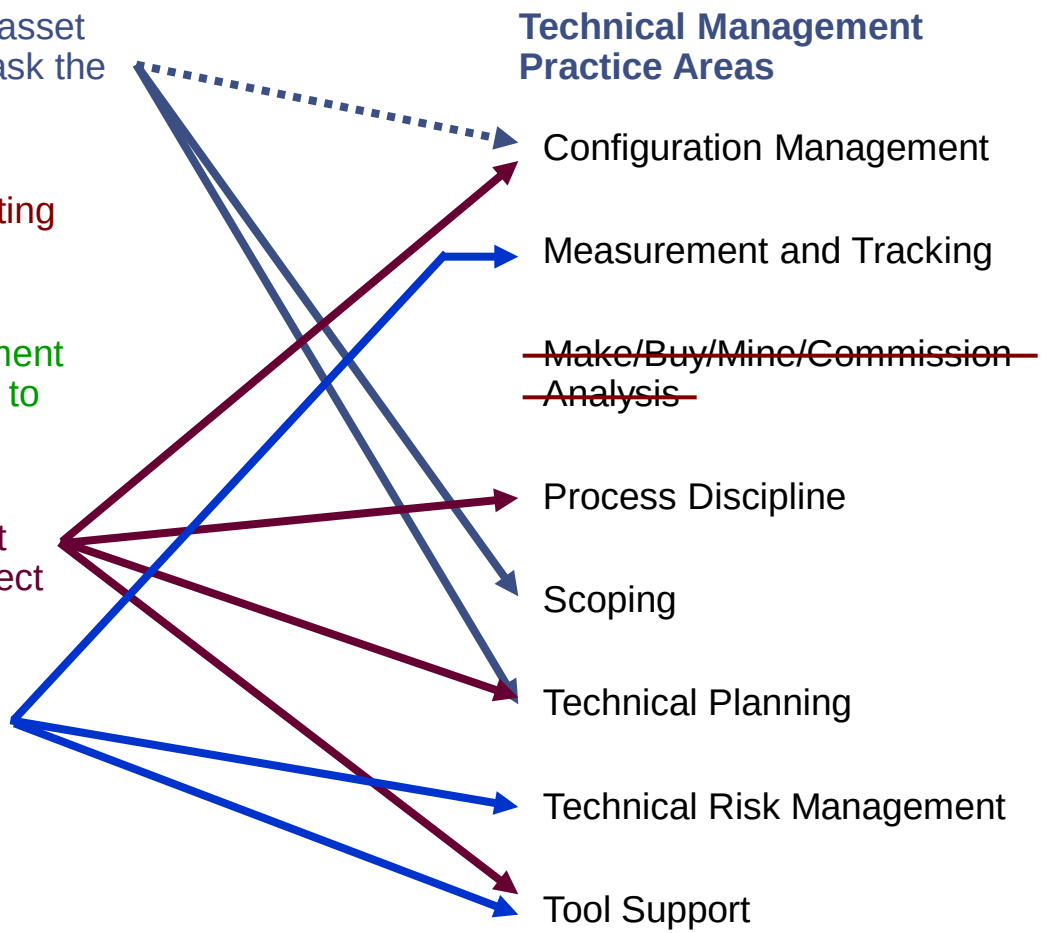
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.



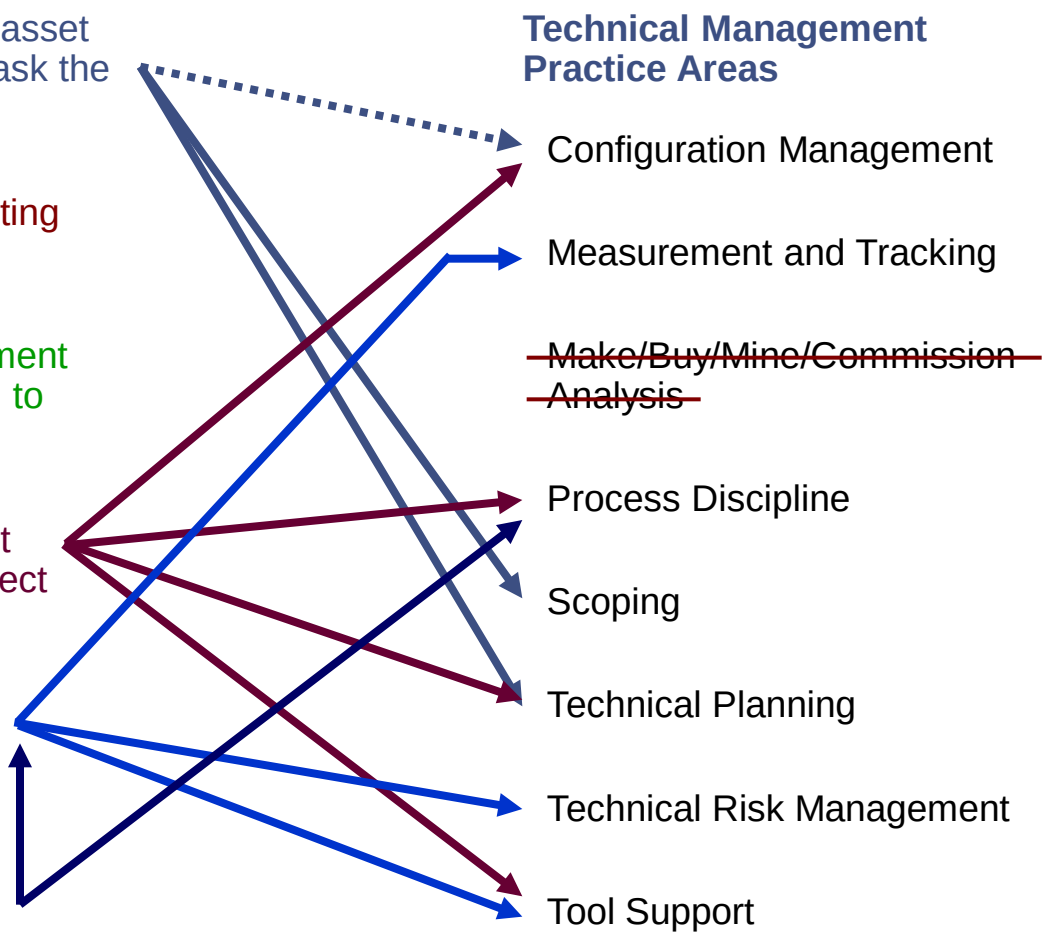
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.
- Pilot the first application. Improve workflow and tool set.



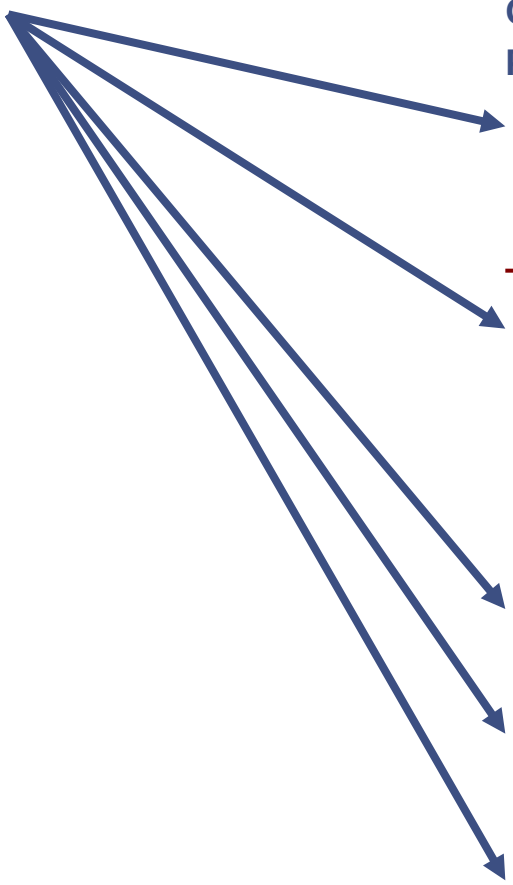
Launching Steps as Practice Areas

- Establish and implement the core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.
- Pilot the first application. Improve workflow and tool set.
- Roll out subsequent products, continue to improve the workflow and tool set.



Launching Steps as Practice Areas

- Establish and implement core asset implementation plan. Select and task the core group.



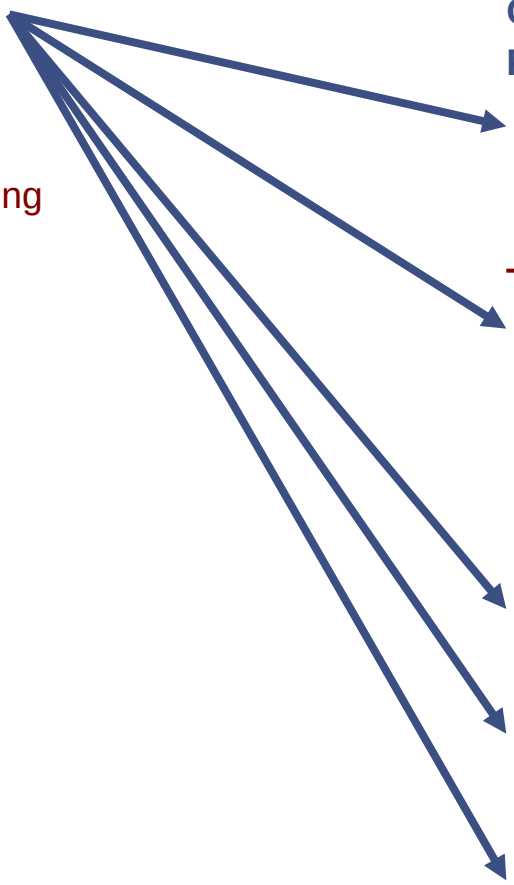
Organizational Management Practice Areas

- Building a Business Case
- Customer Interface Management
- ~~Developing an Acquisition Strategy~~
- Funding
- Launching and Institutionalizing*
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

* all activities

Launching Steps as Practice Areas

- Establish and implement core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.



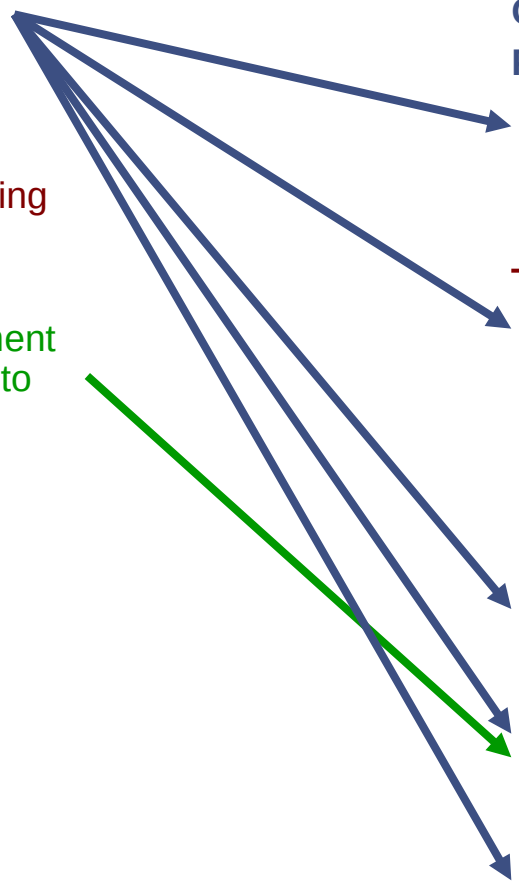
Organizational Management Practice Areas

- Building a Business Case
- Customer Interface Management
- ~~Developing an Acquisition Strategy~~
- Funding
- Launching and Institutionalizing*
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

* all activities

Launching Steps as Practice Areas

- Establish and implement core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.



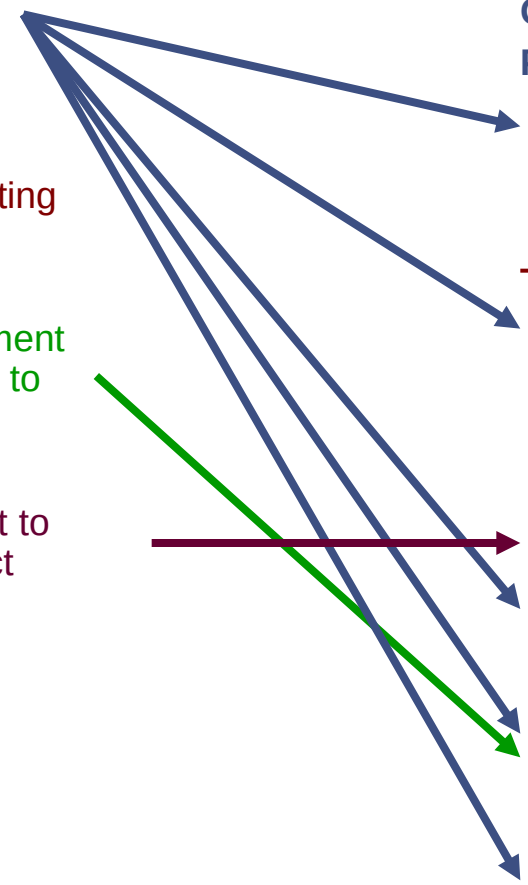
Organizational Management Practice Areas

- Building a Business Case
- Customer Interface Management
- ~~Developing an Acquisition Strategy~~
- Funding
- Launching and Institutionalizing*
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

* all activities

Launching Steps as Practice Areas

- Establish and implement core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.



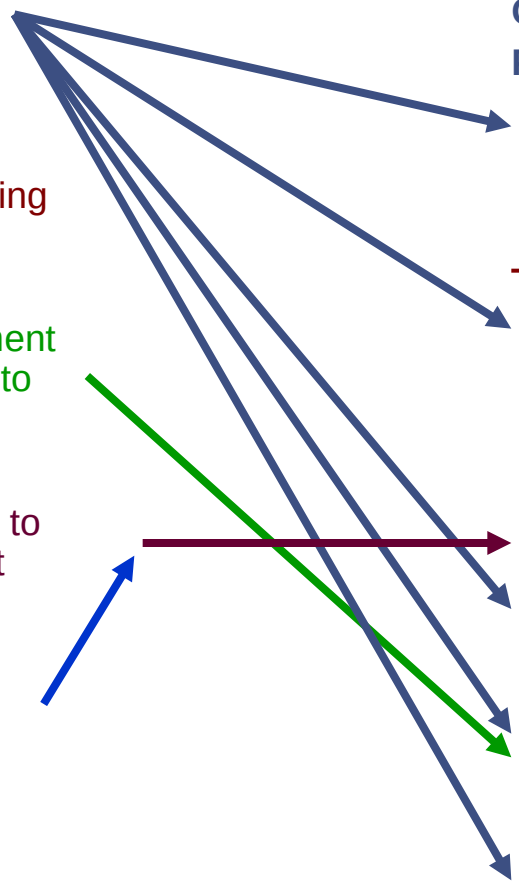
Organizational Management Practice Areas

- Building a Business Case
- Customer Interface Management
- ~~Developing an Acquisition Strategy~~
- Funding
- Launching and Institutionalizing*
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

* all activities

Launching Steps as Practice Areas

- Establish and implement core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.
- Pilot the first application. Improve the workflow and tool set.



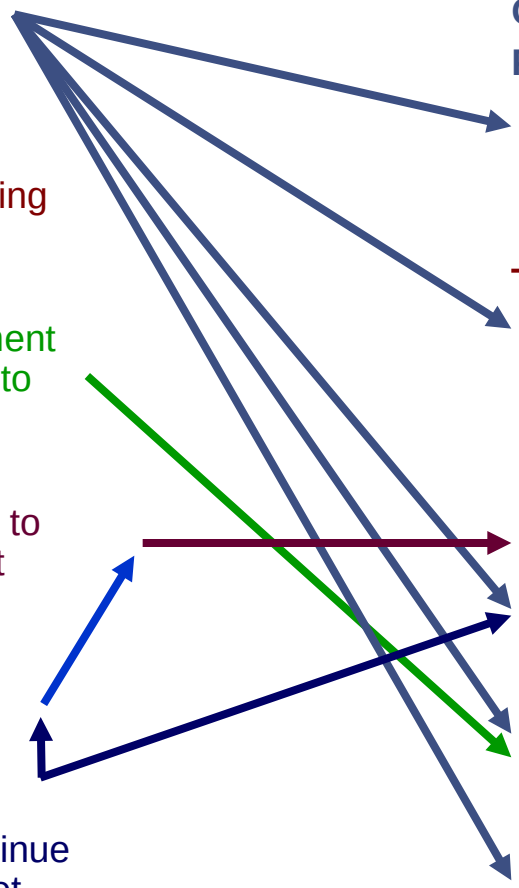
Organizational Management Practice Areas

- Building a Business Case
- Customer Interface Management
- ~~Developing an Acquisition Strategy~~
- Funding
- Launching and Institutionalizing*
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

* all activities

Launching Steps as Practice Areas

- Establish and implement core asset implementation plan. Select and task the core group.
- Select legacy software for the starting point.
- Establish core asset and development teams. Modify the legacy software to make it meet the architecture.
- Establish the workflow and tool set to be used. Establish detailed project plans.
- Pilot the first application. Improve the workflow and tool set.
- Roll out subsequent products, continue to improve the workflow and tool set.



Organizational Management Practice Areas

- Building a Business Case
- Customer Interface Management
- ~~Developing an Acquisition Strategy~~
- Funding
- Launching and Institutionalizing*
- Market Analysis
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Technology Forecasting
- Training

* all activities

Session 1 Contents

Case Studies

Cummins Background

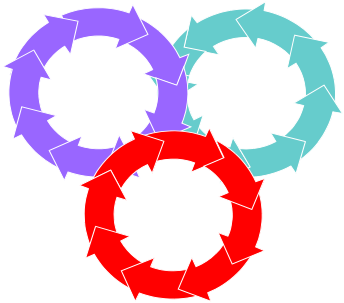
Launching the Cummins Product Line

Practice Areas of Particular Interest

Cummins' Results

Building the Core Assets - 1

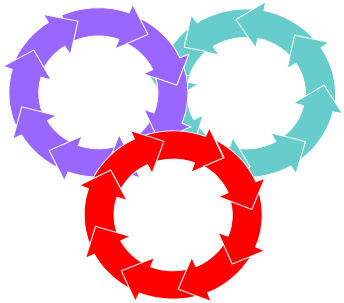
Mining Existing Assets



- Cummins chose a single system under development as the source of its software core assets.
- That system's architecture was the most modular and well documented.
- It had demonstrated the necessary reliability and stability.

Building the Core Assets - 2

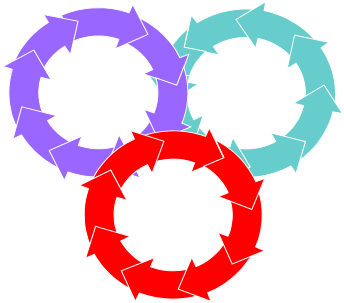
Requirements Engineering/ Understanding Relevant Domains



- Cummins used a combination of traditional requirements analysis and lightweight domain analysis.
- *Features* (customer-visible capabilities) were the basic unit.
- Domain experts in different market segments were interviewed. They identified features and points of variability.
- Features were labeled “unique” (assigned to product teams) or “common” (assigned to the core asset team).
- Features were assigned likelihood, delivery timing, and relative priority.

Building the Core Assets - 3

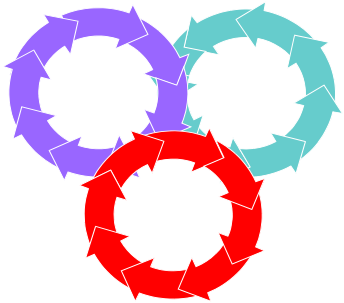
Architecture Definition/ Software System Integration



Cummins relied on modularity and the ability to combine different feature-implementing modules.

- Wrappers were used to “rehabilitate” legacy components and provide parameter-based variability.
  - highly parameterized code. 300 parameters are available for the customer to set after delivery.
- Generators were used to implement the variability.
- Build tools were used to automate builds.
  - “...build tools more than paid for themselves by removing human error from tedious, time-consuming work. They also provided a structure to enforce the use of the architectural variability mechanisms.”

Structuring the Organization



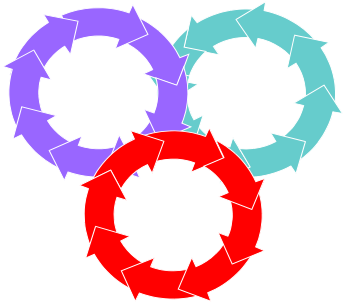
At Cummins, engines are roughly divided into automotive, industrial, and power generation, with one business unit for each.

To prevent stove-piping, each business unit (BU) has only partial responsibility; none is self-sufficient.

- The core asset group is part of the industrial BU.
- The automotive BU has some of the resources for manufacturing commonality.

Cooperation is ingrained in the culture: BU managers have “personal commitments” to one another. Everyone understands that nobody succeeds unless everybody succeeds.

Funding

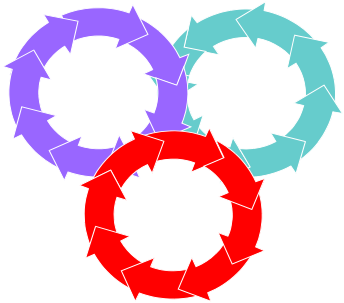


At Cummins, funding policy helps makes the organizational structure work.

In addition to paying for creating the core assets, Cummins' funding policy assures that they are *used*.

- The core asset group's budget is established.
- Each BU is billed a portion of that budget in proportion to its sales.
- Thus, each BU pays for core assets whether it uses them or not. Thus, there is every incentive to use the core assets.

Organizational/Technical Planning



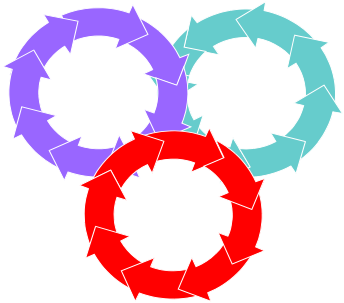
Cummins makes the organizational structure work.

The core asset group does not set priorities. They are set by the business units at weekly meetings where progress is reported.

"Face-to-face meetings are critical..."

- "…because you have to look the other managers in the eye when you say what your project's priorities are."
- This is where "personal commitments" are made.
- Such meetings discourage cheating. No manager wants to admit he had the resources to bypass the core asset group and build a component himself. In Cummins' culture, that's a sign of an overfunded project!

Other Practice Areas at Cummins



Architecture Evaluation

- frequent, unstructured reviews of architecture

Testing

- unit testing, subsystem integration testing, system testing, deployment testing, regression testing, acceptance or field testing
- Core asset team performs generic testing.
- Product teams perform product testing.

Risk Management

- Every contract includes risk management in stated activities.
- standard “identify/analyze/plan/track/control” model
- addressed during technical reviews, management reviews, and weekly coordination meetings

Session 1 Contents

Case Studies

Cummins Background

Launching the Cummins Product Line

Practice Areas of Particular Interest

Cummins' Results

Cummins' Results

In early 1995, the product was launched on time (relative to revamped schedule) with high quality. Others followed—on time and with high quality.

Achieved a product family capability with a breathtaking capacity for variation, or customization

- nine basic engine types
- 4-18 cylinders
- 3.9 - 164 liter displacement
- 12 kinds of electronic control modules
- five kinds of microprocessors
- 10 kinds of fuel systems
- diesel fuel or natural gas

Quantitative Results - 1

Twenty product groups were launched, which account for over 1,000 separate engine applications.

Seventy-five percent of all software, on average, comes from core assets.

Product cycle time has plummeted. The time to first engine start went from 250 person-months to a few person-months. One prototype was built over a weekend.

Software quality is at an all-time high, which Cummins attributes to the product line approach.

Quantitative Results - 2

Customer satisfaction is high. Productivity gains enable new features to be developed (more than 200 to date).

Projects are more successful. Before product line approach, 3 of 10 were on track, 4 were failing, and 3 were on the edge. Now, 15 of 15 are on track.

There is a widespread feeling within Cummins that developers are more portable and hence more valuable.

Quantitative Results - 3

Supported Components	1992	1993	1994	1995	1996	1997	1998
Electronic control modules (ECMs)	3	3	4	5	5	11	12
Fuel Systems	2	2	3	5	5	10	11
Engines	3	3	5	5	12	16	17
Features per ECM	60	80	180	370	1100	2200	2400

Achieving this flexibility without the product line approach would have required 3.6 times the current staff.

Quantitative Results - 4

Today's largest teams are smaller than yesterday's smallest teams. Two-person teams are not unusual.

Cummins' management has a history of embracing change—carefully targeted change.

- Managers estimate that process improvement alone has brought a benefit/cost ratio of 2:1 to 3:1.
- Managers estimate that the product line approach has brought a benefit/cost ratio of 10:1.

The product line approach let the company quickly enter and then dominate the *industrial* diesel engine market.

Lessons Learned - 1

Cummins credits its success to an effective, well-placed champion (Temple).

A culture of cross-organizational cooperation reinforced by policy was a recurring theme.

Mining small assets is often the most feasible task, but it delivers the smallest payoff. Resolving this tradeoff is the key to successful mining.

Developers recognized that configuration management is critical and strongly suggest keeping traces from requirements - test cases.

Lessons Learned - 2

The product builder's guide started out as a "guide" that was hundreds of pages long and was descriptive rather than prescriptive. Compliance was not enforced. All of these were mistakes that were later corrected.

Cummins tracks cost, reuse, and conformance metrics of various kinds. "Metrics are a challenge. Institutionalize them up front."

Engineering the requirements across business units was critical.

Current Status - 1

The original product line that Cummins built is now called the Core I product line.

The Core I product line was a success beyond expectations, but it did have shortcomings **and** Cummins' organizational context changed a bit over the years.

The original Core I team did a self-diagnosis against the 29 practice areas and noted deficiencies.

Cummins launched a Core II product line

- not from emergency business needs, as was Core I
- but rather from a mature realization that the organization could do better

Current Status - 2

Core II includes

- a new core asset base
- newly derived products
- a new product line process
- a new production method, strategy, and plan
- a new organizational structure
- a new operational concept
- a powerful, new toolset

Core II is meeting its goals.

- It is a much fuller and more mature product line capability.

Current Status - 3

The overall impact of a software product line approach on Cummins as measured by Core II results includes

- freed up resources (time, money, and people) to invest in new technologies and state-of-the-art tools and simulation capabilities
- an all-time high in product quality
- continuously shrinking time to market
- an ability to handle increased breadth and complexity of products
- an ability to outpace its market rivals

Product lines have now become institutionalized at Cummins.

- There is no longer a need for a product line champion.
- The product line approach is ***community practice***—everyone is a champion of the product line approach.

Session 1 Summary

Cummins' product line launch story echoes product line success themes seen on other successful efforts, namely

- a compelling business case
- deep domain expertise
- a rich legacy base
- a dedicated champion
- organizational cohesion
- courage to try new engineering approaches

Cummins' product line institutionalization story shows how an organization can translate product line adoption into a continuous, competitive edge.

Carnegie Mellon University

Software Engineering Institute

Part 3: Putting the Practice Areas into Action
Session 2: Case Studies: Control Channel Toolkit

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 2 Objectives

This lecture will acquaint participants with the

- CCT context
- practices that were used to launch CCT
- CCT core assets and the processes that generated them
- management practices employed on the CCT effort
- results achieved by the CCT effort
- lessons learned by the CCT effort

Session 2 Contents

Background

The Control Channel Toolkit (CCT) is a software asset base for ground-based spacecraft command and control systems that was commissioned by one organization (the National Reconnaissance Office [NRO]) and built under contract by another (the Raytheon Company).

Control Channels - 1

Ground-based command and control is employed throughout the life cycle of a space vehicle project.

- Ground-based spacecraft command and control systems are also called control channels.
- A typical control channel is 500,000 LOC.

Control channels provide ground processing to support spacecraft so that operations staff can

- monitor spacecraft functions
- configure spacecraft service and payload systems
- manage spacecraft orbits and attitudes
- perform mission planning

Control Channels - 2

Some control channel processing is real time or near real time and is called ***execution*** processing.

The remainder is batch or off-line processing and is called ***planning*****.**

Control Channels: Execution Processing

The control channel

- receives raw telemetry from the spacecraft and the antenna ground system through its front-end processing equipment and converts it into a client-usable form
- distributes the telemetry to client processes to make real-time assessments of the spacecraft's health
- permits client processes to initiate command requests
- archives telemetry, command, and other system data
- models the on-board processor instruction and data loading and its execution

Control Channels: Planning

The control channel performs planning functions to

- estimate and propagate the spacecraft orbit and attitude
- calculate maneuvers to change orbit or attitude
- schedule future contacts and resource needs

These functions include both launch and early-orbit support as well as on-orbit operations.

The CCT Concept

CCT (begun in August 1997) emerged from earlier reuse efforts (DCCS, 1994; SSCS, 1996).

CCT was originally conceived to be a “toolkit” that would consist of

- a set of reusable software components
- tools to help integrate them into complete systems

The toolkit concept evolved into a software product line asset base.

What Is CCT?

CCT is a software product line asset base consisting of

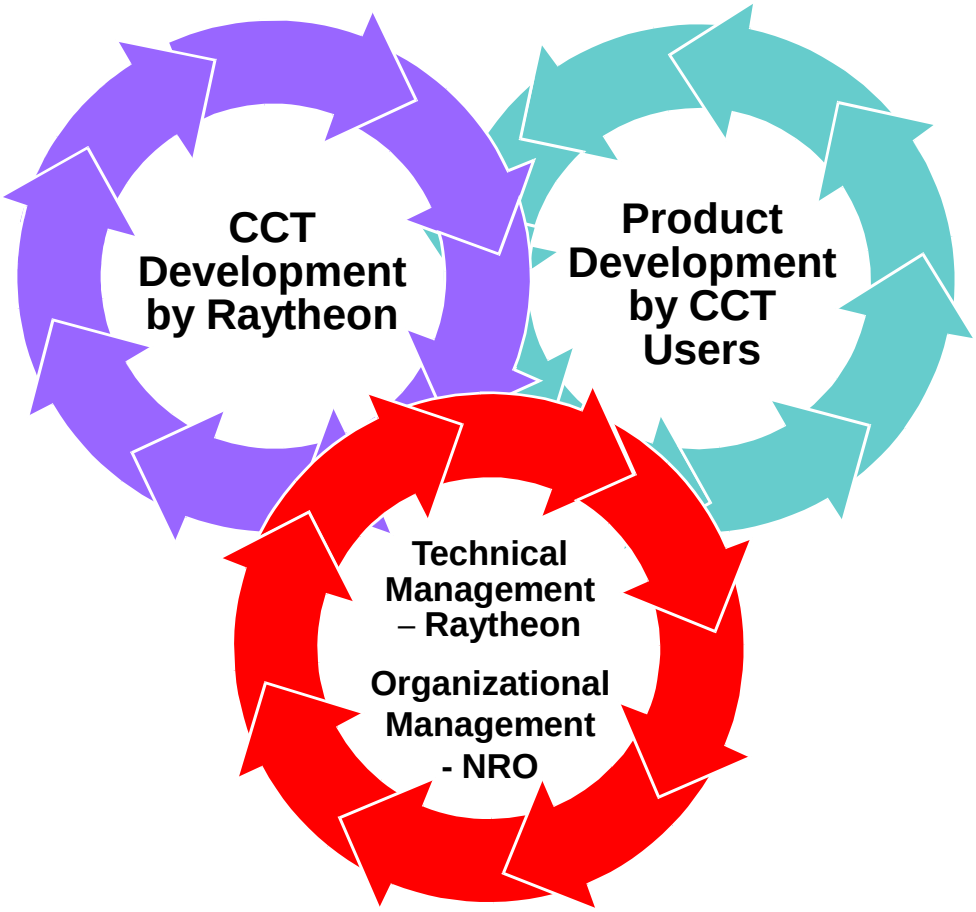
- generalized requirements
- domain specifications
- a software architecture
- a set of reusable software components
- test procedures
- a development environment definition
- a guide for reusing the architecture and components

CCT Users

CCT users build products.

CCT users are individual government contractors commissioned by a government office to build spacecraft command and control systems using the CCT software assets.

The Three Essential Activities and CCT



Organizational Profiles: The NRO

The NRO

- designs, builds, and operates defense reconnaissance satellites
- provides intelligence products that can warn of potential trouble spots around the world, help plan military operations, and monitor the environment

Organizational Profiles: Raytheon

The NRO selected Raytheon* as the prime contractor for the CCT development.

Raytheon

- had broad experience in satellite ground control systems
  - was knowledgeable in orbit dynamics, coordinate system transformations, vehicle configuration variability, and so forth
- was accustomed to following defined processes for both software project management and software development

* *Actually the NRO selected the Hughes Electronics Corporation, which subsequently merged with the Raytheon Company.*

Organizational Profiles: CCT Development Team

The Raytheon CCT team

- had 20-25 members through architecture implementation
- had 45 members during component engineering
- was part of the Denver, CO office
- had few subcontractors

Session 2 Contents

Contextual Background

Launching CCT

Engineering the CCT Core Assets

Managing the CCT Effort

Early Benefits from CCT

Lessons and Issues

Launching Practice Areas

The feasibility study team, and the initial CCT team that followed, exercised practices in the following software product line practice areas:

- Building a Business Case
- Developing an Acquisition Strategy
- Structuring the Organization
- Technical Planning
- Organizational Planning
- Operations

Together these practices areas made up the Launching part of the “Launching and Institutionalizing” practice area.

Building a Business Case for CCT - 1

The business goals for CCT were to

- reduce life-cycle costs
- reduce development risks
- promote interoperability
- provide flexibility
  - Flexibility meant accommodating multiple implementation contractors and enabling the integration of both commercially available and legacy assets.

Building a Business Case for CCT - 2

The feasibility study team examined an array of government and commercial spacecraft command and control systems, including the three under primary investigation. They determined

- original program estimates as baselines
- the commonality of requirements
- risks and mitigation strategies
- the overall cost savings for systems using CCT from1997 to 2009
  - development: 18.2% savings
  - sustainment: 27.8% savings

Building a Business Case for CCT - 3

The feasibility study concluded that

- There was a credible CCT development schedule.
- CCT would ensure significant cost savings for the three programs over a 10-year period.
- Risks that could preclude CCT's success were manageable.
- CCT supported the overall U.S. government reuse vision.

Program sponsors were sold on the vision of greatly reduced risk and the promise of cost savings in the future.

Acquisition Strategy

The CCT asset base was commissioned by the U.S. government.

Three government programs formed a partnership to acquire software that could be used across those programs, as well as others.

The U.S. government retains total rights to the architecture and components.

Raytheon is free to develop commercial derivations.

Structuring the Organization

There were four organizational groups ***(stakeholders)*** involved in the CCT effort:

1. the NRO
2. Raytheon (the commissioned developer)
3. organizations developing the systems for which CCT was launched (Spacecraft C2 System, DCCS, and SSCS)
4. future CCT users

Structuring the Organization: Raytheon - 1

Organizational Unit	Responsibilities
contractor program office	technical management of development effort; accountable to the NRO's CCT Program Office
program support	technical management support functions: for example, configuration management, quality assurance, business operations
domain engineering and architecture	requirements engineering; architecture definition
component engineering	component development
application engineering	testing of architecture that demonstrates the ability to build products from CCT
test engineering	component and assembly testing
training	training materials and internal training of CCT personnel
sustainment engineering	fixing and enhancing of CCT baseline; working with application engineers and users to refine, deliver, and maintain the CCT assets

Structuring the Organization: Raytheon - 2

CCT development was structured into six overlapping increments with an *integrated development team* responsible for each.

- Members of the integrated teams came from a cross section of the organizational units (with the exception of the first increment that involved only the Domain Engineering and Architecture groups).
- The phases and the integrated teams made the iteration explicit.

Structuring the Organization: NRO

The NRO's organizational unit for CCT was the CCT Program Office consisting of

- a small team of government management and technical personnel who managed CCT development at the organizational level
- a small number of technical personnel from two government research centers (the SEI and the Aerospace Corporation)

The NRO also created a variety of *working groups* to ensure effective communication and guide CCT over the extended life cycle.

- CCT Working Group – coordinated project-level decisions for the CCT program and ran the stakeholder working groups
- Stakeholder Working Groups – provided direction for the CCT component and sustainment engineering groups
- Architecture Working Group – advised the CCT Working Group on issues related to the CCT architecture

Organizational and Technical Planning - 1

The NRO and Raytheon partnered in the planning of CCT.

- The development period was from August 1997 to December 1999.
- The development plan addressed
  - the NRO's organizational management needs
  - Raytheon's technical management needs for plotting and tracking
  - Spacecraft C2's needs for the CCT assets

Organizational and Technical Planning - 2

- Other plans included
  - Configuration management
  - Risk management
  - Quality assurance
  - Testing
  - Transition
  - Sustainment

All CCT plans were

- accessible at the CCT Web site
- updated religiously

Operations - 1

Operations for core asset development were governed by the plans set in motion:

- The integrated teams for each of the six development increments carried out the development tasks.
- The CCT effort was process driven and took advantage of what was available (legacy assets, proven algorithms, and COTS products).
- There were frequent technical interchange meetings, weekly status meetings with the CCT Program Office, and extensive use of the CCT Web site.

Operations - 2

Operations for core asset development were governed by the plans set in motion: (cont)

- The NRO held a modest number of reviews.
- Metrics were defined, and corresponding data was collected.
- A proof-of-concept prototype for mining was developed.

The NRO developed the *CCT Program Concept of Operations* (CONOPS) to document explicitly

- how the CCT effort would connect with the product line it was intended to support
- procedures and criteria for working with potential CCT users

Session 2 Contents

Contextual Background

Launching CCT

Engineering the CCT Core Assets

Managing the CCT Effort

Early Benefits from CCT

Lessons and Issues

CCT Software Engineering Processes - 1

The CCT development effort consisted of six major software engineering processes:

1. domain engineering and architecture – defined and specified the product line and created the architecture
2. component engineering – created the components
3. application engineering – created an application as a test product and tested "reusability" of the assets
4. test engineering – validated CCT
5. sustainment engineering – managed the maintenance and evolution of the assets
6. training – created training materials for CCT team and users

Note: These processes map to the practice areas, but CCT parlance is used to be faithful to the experience

CCT Software Engineering Processes - 2

These processes

- proceeded through six increments, delivering successively more complete versions of the CCT assets
- incorporated many of the practices of Jacobson's incremental and iterative reuse-based approach and Kruchten's "4+1 view" model
- evolved during the course of the development (e.g., an architecture evaluation step and the architecture model as a separate deliverable were added)
- resulted in the major CCT assets
  - domain specification
  - architecture model
  - reference architecture
  - reuse guide
  - components

Domain Analysis - 1

Though a formal domain analysis method was not used, the following essential tasks of domain analysis were performed:

- capture and analyze common requirements and their variation across several systems
- synthesize them into a set of common requirements for the product line
- capture the essential terminology

Domain Analysis - 2

Product line requirements were classified into two domains:

- Execution
- Planning

Within these two domains, component categories were identified (e.g., the handling of telemetry streams and orbit estimation) as were the common services that support these categories (e.g., persistent storage services and event notification services).

This partitioning provided the basis for the logical view of the CCT architecture.

Domain Analysis - 3

There were three documents that described the scope and the general requirements of the product line:

1. Domain Definition Document – described the common features and defined the scope in terms of the mission and system characteristics that CCT supports
2. Generalized Requirements Specification – documented the CCT requirements, capturing common capabilities to be provided and expressing variability in terms of variation points
3. CCT Domain Specification Document – contained use case diagrams and descriptions for the common requirements

Architecture - 1

The architecture follows the “4+1” view model and was described in UML, as supported by the Rational Rose tool.*

Architecture definition and component engineering were closely coupled.

The process for defining the software architecture evolved considerably during the development of the CCT assets.

* *The tool did not support some CCT architectural representations that consequently had to be handled separately.*

Architecture - 2

Two key documents describe the CCT architecture and its use:

1. CCT Architecture Model – provides a complete description of the architecture (philosophical underpinnings, architectural drivers, architectural and design patterns,* and architectural views)
2. CCT Reuse Guide – documents the CCT assets that could be used to build a product in the product line and the programming approaches to use

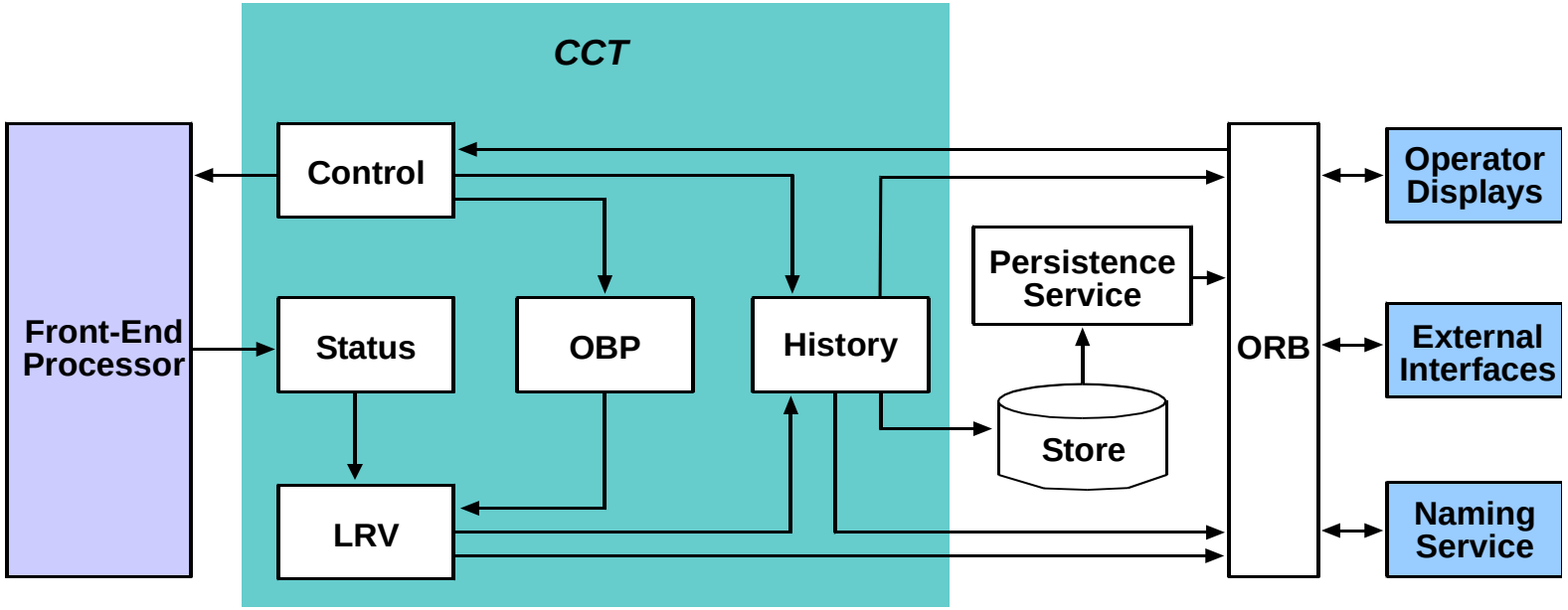
* *A notable feature of the CCT architecture is the abundant use of architectural and design patterns, which provided a vocabulary for describing and reasoning about the architecture.*

Architecture - 3

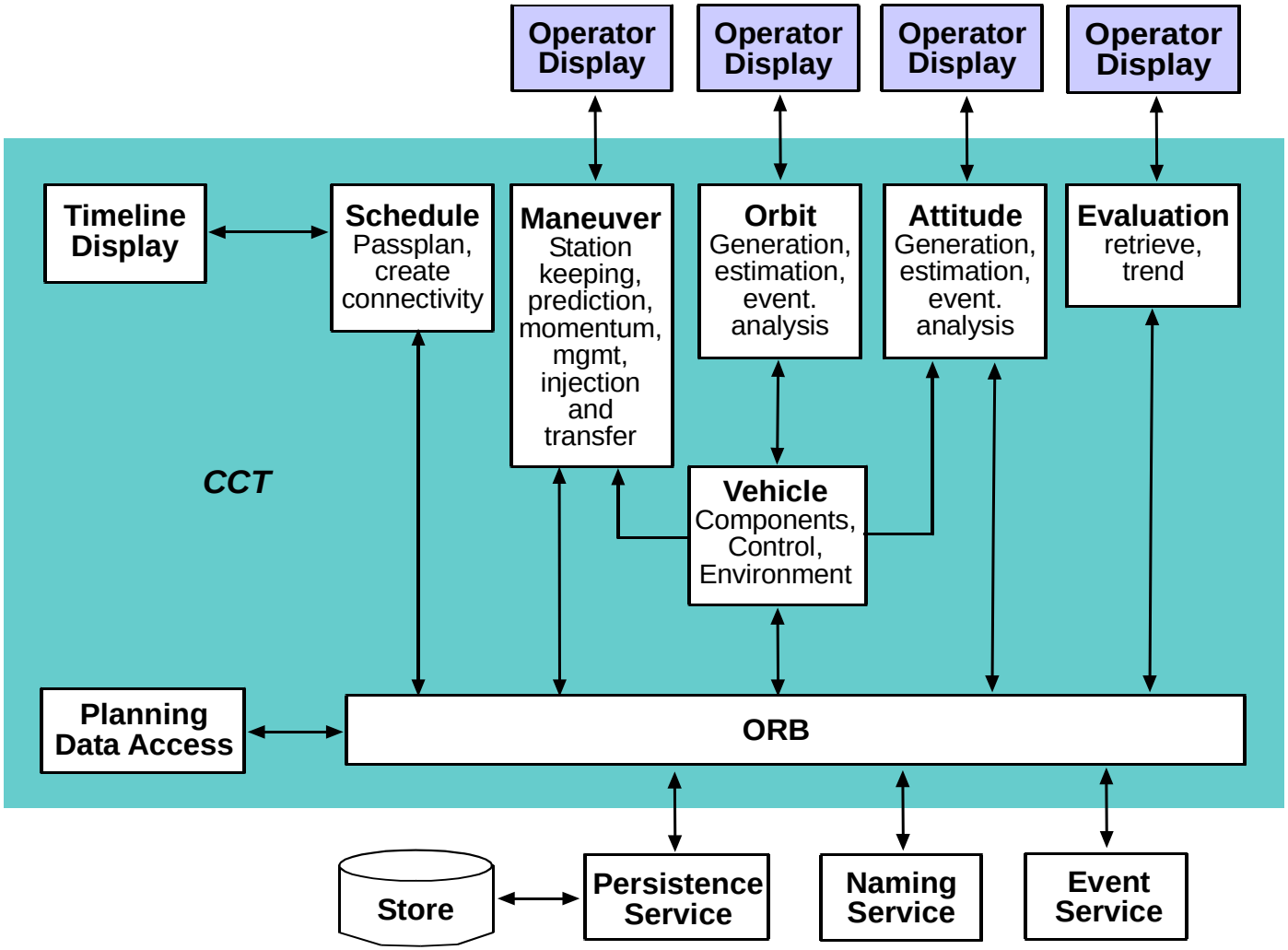
The CCT architecture

- uses the Common Object Request Broker Architecture (CORBA) for the inter-component communication infrastructure
  - CORBA facilities are used for common services.
- has two subsystems: *planning and execution*
- within the subsystems, divides functionality into component categories
  - Components in different categories might use each other in well-defined ways to achieve system functions.
  - Component categories across the two subsystems do not interact except by reading data from and writing data to shared files.

Architecture - 4: CCT Execution Architecture



Architecture - 5: CCT Planning Architecture



Architecture - 6: Variation Support

Each variation point identified during domain specification was supported in the architecture by one of six standard mechanisms:

1. dynamic attributes
2. template
3. inheritance
4. parameterization
5. function extension (callbacks)
6. scripting

Building Products from CCT Assets

The process of starting with the architecture and components and building applications included the following steps:

1. Determine COTS and legacy usage in the intended system.
2. Identify real-time user interface products and database implementations.
3. Select a CORBA vendor to provide the intra-system communication infrastructure.
4. Address security needs by adding security layers or secure gateways.
5. Determine how to extend and vary the CCT components.
6. Package the components into executable applications and allocate to nodes on the intended system's physical network.

Architecture Evaluation

Both the planning and execution architecture underwent explicit evaluation using the SEI Software Architecture Analysis Method (SAAM).

- The SAAM is a scenario-based architecture evaluation approach that specifically examines modifiability and that involves the system (in this case, the product line) stakeholders in the evaluation process.

For CCT, the SAAM produced scenarios of usage for ground-based spacecraft command and control systems.

- Some were aimed at illuminating the production plan by which systems are built from the CCT assets.

Neither the planning nor the execution architecture revealed any modifiability problems that would require more than a few person-months to correct.

Component Engineering

The components in each component category were implemented:

- according to the development plan in planned increments (with a few exceptions)
- to handle the variation points already specified
- to have the interfaces and interconnection mechanisms called for by the software architecture
  - Descriptions of component interfaces were included in the Architecture Model and restrictions were specified in the Reuse Guide.

Testing - 1

Application engineering and test engineering together built the system test architecture and executed detailed test procedures.

The system test architecture was a test application that was incrementally built and used to perform the formal testing of CCT components.

The test engineers validated CCT's reusability, including the architecture and other assets:

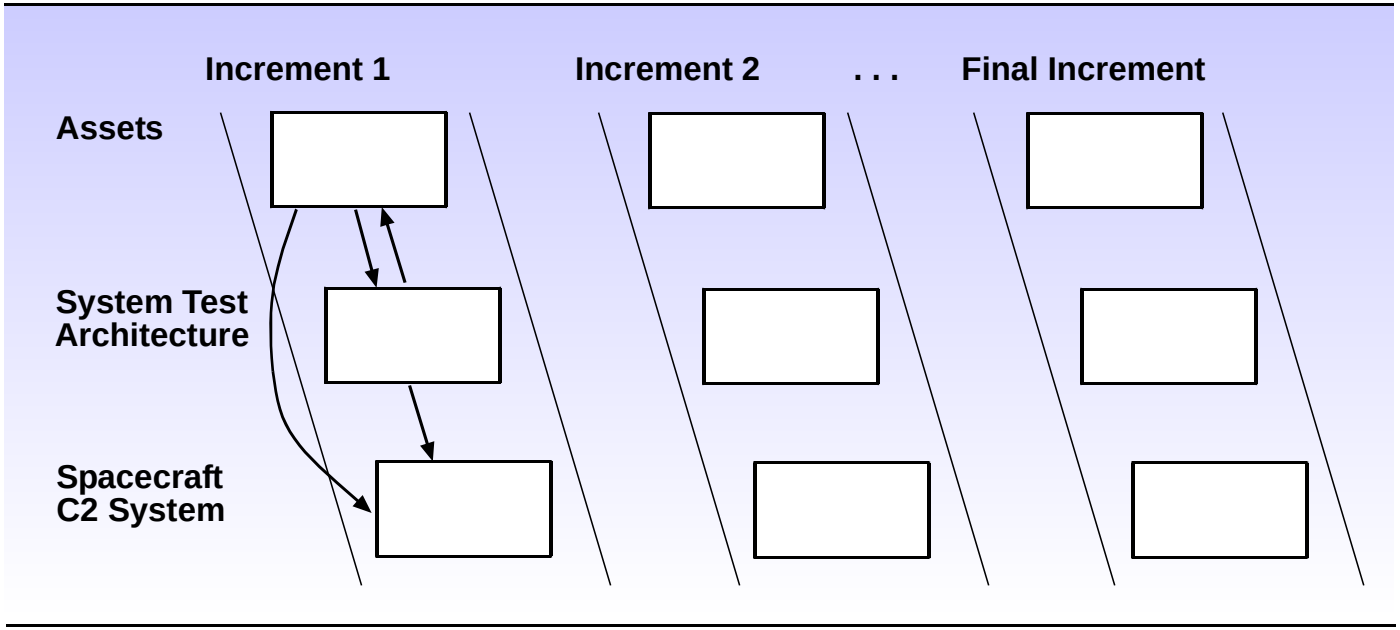
- They built test plans and cases based on the use cases in the Domain Specification.

Testing - 2

CCT defined four levels of testing:

1. Level 0: unit testing (informal, performed by component engineering)
2. Level 1: requirements verification of components and component categories (formal, performed by test engineering)
3. Level 2: integration testing using the test architecture (informal, performed by application engineering
4. Level 3: system-level performance requirements verification (formal, performed by test engineering)

Testing - 3



Testing - 4

The Test and Integration Plan described the test methodology in generic terms for the first increment and then in more specific terms, on an increment-by-increment basis.

Discrepancy reports documented and tracked all problems discovered during testing.

The test engineering process validated CCT's reusability.

The test engineering group participated throughout the CCT development cycle and in all the standing meetings.

Documentation

Because it was a product line, the CCT program produced some documents that are not found in typical government procurement cycles, including

- CCT Domain Definition
- Generalized Requirements Specification
- CCT Domain Specification
- CCT System Test Architecture
- CCT Portability Demonstration Plan
- CCT Sustainment Support Plan
- CCT Program Concept of Operations
- CCT Architecture Model
- CCT Reuse Guide

Session 2 Contents

Contextual Background

Launching CCT

Engineering the CCT Core Assets

Managing the CCT Effort

Early Benefits from CCT

Lessons and Issues

Management Practices - 1

Raytheon's process sophistication and discipline resulted in the superb execution of CCT's technical management activities.

Practices from the following practice areas were stitched into the fabric of the CCT development process:

- Technical Planning
- Configuration Management
- Technical Risk Management
- Process Discipline
- Measurement and Tracking
- Make/Buy/Mine/Commission Analysis
- Scoping
- Testing

Management Practices - 2

The NRO management chain responsible for CCT all supported strategic reuse.

- They were open to the changes that a product line approach necessitated.
- They provided unwavering support.
- The CCT program manager was deeply committed to the effort, flexible, innovative, and willing to listen to advice from the experts he involved on the government side.

The Raytheon CCT program manager was structured, well organized, technically well grounded, and a superb communicator who turned others in his organization into CCT believers.

Together, the NRO and Raytheon managers (Ohlinger and Shaw) provided a protective shield and a supportive environment for the CCT team; they **led** the CCT product line effort.

Session 2 Contents

Contextual Background

Launching CCT

Engineering the CCT Core Assets

Managing the CCT Effort

Early Benefits from CCT

Lessons and Issues

The CCT Development Effort

CCT is a groundbreaking effort within its government communities.

CCT was completed **on schedule** and **within budget** in December 1999 with **no outstanding risks** and **no outstanding actions.**

First CCT-Based Product - 1

Because of CCT, the Spacecraft C2 Program is enjoying reduced development costs, schedules, number of required workers, and product risks, as well as increased product flexibility.

- **quality:** There was one-tenth the typical number of discrepancy reports for a system of this type, and the problems identified were all local.
- **performance:** Using CCT improved performance over the results predicted without CCT.

First CCT-Based Product - 2

- **integration time:** Incremental builds were completed in weeks rather than months.
- **code volume:** Total SLOC developed by Spacecraft C2 was 76% less than originally planned.
- **productivity:**
  - development staff of 15 versus 100+
  - 50% reduction of overall costs
  - 50% reduction of overall development time
  - flexibility in meeting requests for modifications

First CCT-Based Product - 3

CCT was treated like a COTS product initially; training was required, and then development proceeded on the basis of a domain specification, interface definitions, and the Reuse Guide.

Interviews with Spacecraft C2 developers revealed

- high morale
- low attrition
- praise for the CCT assets
- greater professional satisfaction with the product line approach than with the traditional single-systems approach

Beyond the First Customer

The NRO is using the CCT assets in other programs.

The NRO and SEI jointly developed a business case based on the CCT experience for NRO-wide strategic reuse through product lines.

Raytheon is using the CCT assets in other systems and the CCT processes and tools in other efforts.

Other commercial organizations have access to CCT assets.

Session 2 Contents

Contextual Background

Launching CCT

Engineering the CCT Core Assets

Managing the CCT Effort

Early Benefits from CCT

Lessons and Issues

Lessons Learned

Many lessons were learned during the CCT effort. Some are issues that nag other product line efforts:

- Tool support is inadequate.
- Domain analysis documentation is important.
- An early architecture focus is best.
- Product builders need more support.
- CCT users need reuse metrics.
- It pays to be flexible.
- Cross-unit teams work.
- A real product is a benefit.

Current Status - 1

The lessons from CCT have been applied across Raytheon.

- The whole product line approach has been replicated.
- CCT spawned a Collaborative Product Reuse (CPR) strategy within this division of Raytheon.

Raytheon is now at version 8.0 of CCT.

- There is still the original execution and planning structure.
- There is a single maintenance effort that keeps track of multiple baselines.
- CCT still uses CORBA, but there is a move to services; CCT lends itself to a service-oriented approach.

Current Status - 2

There are formal and informal users of CCT.

- All formal users are classified systems. The benefits continue to be substantial.
- The original production plan is followed. The users use the CCT Reuse Guide.
- Parts of CCT have been ported to another product line.
- Parts of CCT and the CCT philosophy have been ported to other systems.
- No contractor other than Raytheon has used CCT.

Leaving out the man-machine interface (MMI) has had adverse effects.

- CCT has no “face” to it to demonstrate its capabilities to other potential customers.

Session 2 Summary

CCT echoes product line successful launch themes seen on other successful efforts, namely

- deep domain expertise
- a rich legacy base
- process maturity
- a dedicated champion (in this case two)
- a strong architectural vision
- an incremental development and refinement approach

Carnegie Mellon University

Software Engineering Institute

Part 3: Putting the Practice Areas into Action
Session 3: Case Studies: Salion, Inc.

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 3 Objectives

This lecture will acquaint participants with the approach Salion, Inc. took in its product line effort.

Salion, Inc.

A small organization: 21 people

Maker of software for suppliers who sell complex products via proposals:

- **Salion Revenue Process Manager** - helps suppliers manage opportunities. It contains a workflow engine and Web-based communication tools to help a supplier organization manage the collaboration of design and pricing. It keeps track of a proposal's status and assists in the assembly of the final document.
- **Salion Knowledge Manager** - helps triage and analyze requests for proposals (RFPs), with decision support capabilities and analysis of bid performance, win/loss rates, and pricing; helps choose the best candidates from among all the available opportunities; uses historical information to prioritize opportunities and improve response rates
- **Salion Business Link** - extends collaboration between the supplier and the customer, and between the supplier and subsuppliers

Salion's Specialized but Important Market - 1

"It should take us one day or less to turn a quote around. For some reason, it takes five weeks. This process is out of control."
— Director of Engineering, Tier 1 automotive supplier

"We recently rushed a late quote out the door that we thought we had priced with a 'nice margin.' In reality, the quote was for a part that we had been selling at twice the price we quoted. Luckily, our customer only asked for one year of retroactive rebates."
— Director of Sales, Tier 1 automotive supplier

Salion's Specialized but Important Market - 2

"We just spent $100,000 on an opportunity that we had no chance of winning. We bid on the same business two years ago and our price was 50% too high. We have no way to capture or analyze our historical sales and bidding performance, so we make the same mistakes over and over."
— Tier 2 automotive supplier

"We spent $600,000 in overnight shipping costs last year."
— Tier 1 automotive supplier

Variations Handled by Salion, Inc. - 1

Customers run different combinations of products.

There are three installation options:

1. run on customer's hardware (installed)
2. run on Salion's dedicated hardware (hosted)
3. run on Salion's shared hardware

Variations Handled by Salion, Inc. - 2

Each customer will have a unique workflow, a unique set of input screens and other user-interface concerns, and a unique set of reports he/she wants to generate.

Each customer will have unique “bulk load” requirements, involving the transformation of existing data and databases into forms compatible with Salion’s products.

An automotive industry trade group has defined a business-to-business transaction framework encompassing some 120 standard objects to be used to transfer information from organization to organization. Not every customer will want to make use of all 120 objects.

How Salion Builds Its Product Line

Salion first produced a “standard” product as its entry into the market.

That product formed the basis for Salion’s software product line and the basis for each new customer-specific product it fielded.

The standard product was more than an engineering model from which “real” systems were produced; it was also sold.

Typical product: 40 modules, 150K SLOC

Customization Versus Configuration

Salion builds subsequent products by

- *customizing* elements of the “standard” product
- *configuring* elements of the “standard” product

Early on, Salion tried to make many elements configurable:

- forms manager
- customized reports manager

The results were wasted effort, wrong guesses, and bloated software. Now, Salion customizes these aspects.

Tool support plays an important role in managing these variations:

- 3,333 files for 3 products
- 88 files represent variations

Practice Areas Especially Germane to the Salion Effort

Scoping

Understanding Relevant Domains

Process Discipline

Architecture Definition

Using Externally Available Software

Tool Support

Market Analysis

Customer Interface Management

Operations

Measurement and Tracking

Structuring the Organization

Salion's Product Line Benefits

Seven developers produce and support sophisticated, highly secure, high-availability, COTS-intensive systems.

At the time the SEI wrote the Salion case study, Salion had produced its 12th 30-day release, and all releases were on schedule.

Building the standard product took 190 person-months.

- Building the first customer product took just 15 person-months with 97% reuse.
- Building the second product took 30% less effort.

Salion's approach gives it a superb position to answer investors' question: "How are you going to scale?"

- normal answer: Rewrite the product to make it robust, increase development staff, and bring on the QA staff.
- Salion's answer: Nothing. We can scale right now, as we are.

Session 3 Summary

Salion’s software product line story is that of a small, nimble organization that from the beginning recognized that a reactive product line approach was the way to achieve flexibility in an application domain in which the future could not be predicted reliably.

Discussion

Contrast the product line approaches taken by Cummins, Raytheon (on CCT), and Salion, and comment on the appropriateness of each for the organization in question.

Carnegie Mellon University

Software Engineering Institute

Part 3: Putting the Practice Areas into Action
Session 4: Software Product Line Practice Patterns

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

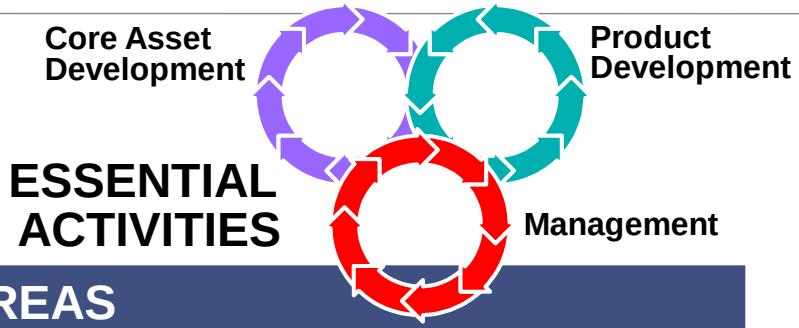
Session 4 Objectives

This lecture will introduce participants to

- the concept of software product line practice patterns
- the product line practice template
- sample software product line practice patterns and how they can be used
- the current software product line practice pattern collection

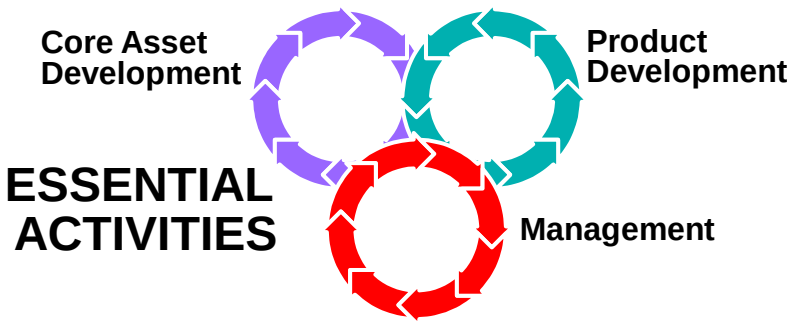
Session 4 Contents

Framework Version 5.0



PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management
Architecture Definition	Configuration Management	Building a Business Case
Architecture Evaluation	Make/Buy/Mine/Commission Analysis	Customer Interface Management
Component Development	Measurement and Tracking	Developing an Acquisition Strategy
Mining Existing Assets	Process Discipline	Funding
Requirements Engineering	Scoping	Launching and Institutionalizing
Software System Integration	Technical Planning	Market Analysis
Testing	Technical Risk Management	Operations
Understanding Relevant Domains	Tool Support	Organizational Planning
Using Externally Available Software		Organizational Risk Management
		Structuring the Organization
		Technology Forecasting
		Training

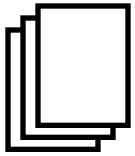
Help to Make It Happen



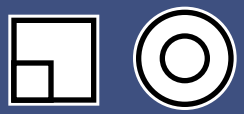
PRACTICE AREAS

Software Engineering	Technical Management	Organizational Management

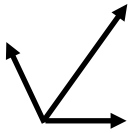
GUIDANCE



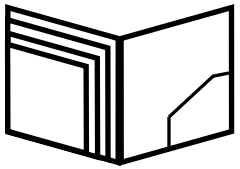
Case Studies



Patterns



Probe



Curriculum

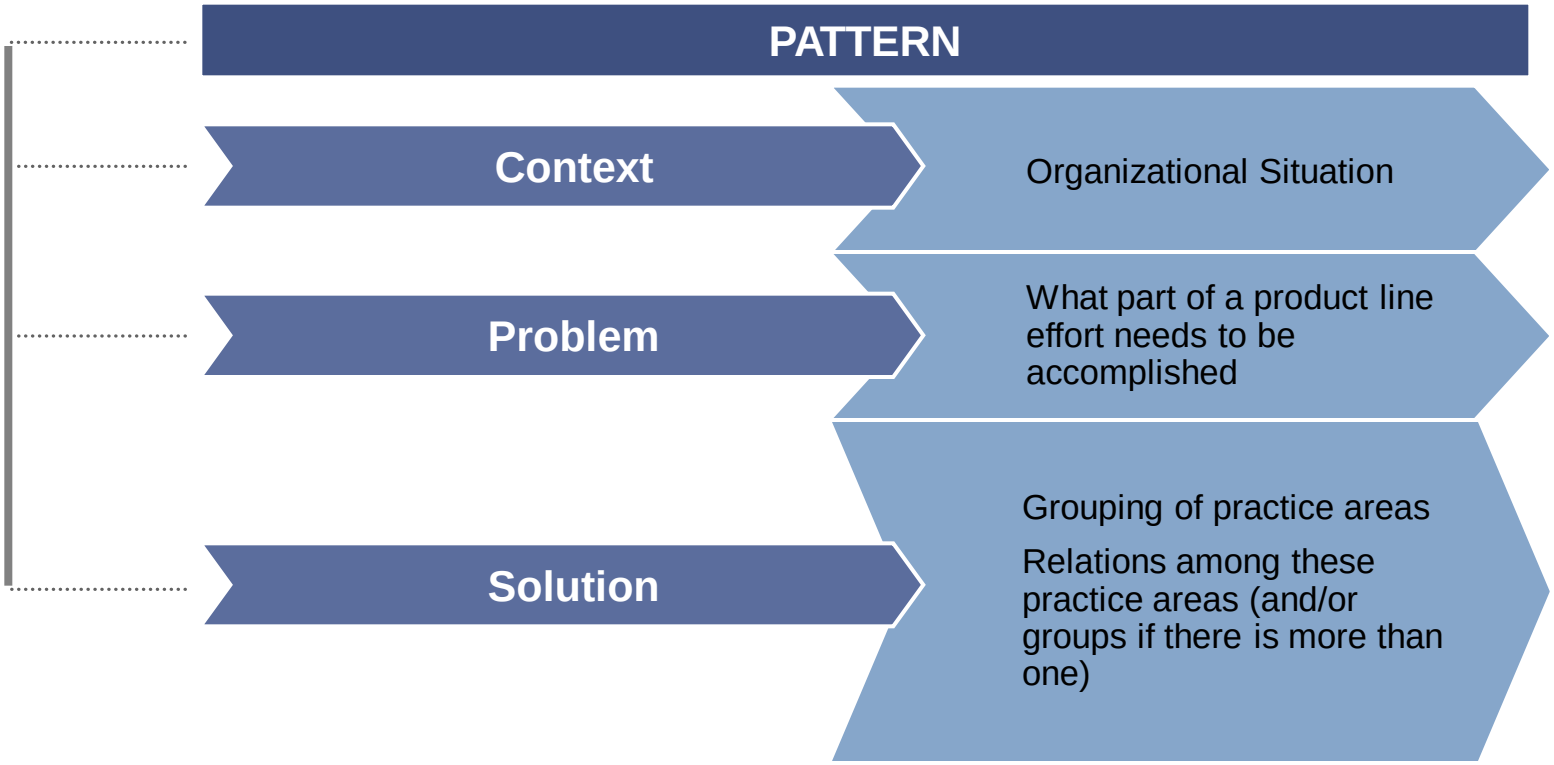
Patterns Can Help

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



How Do Product Line Practice Patterns Help?

Product line practice patterns

- address recurring product line problems
- codify existing, well-proven software product line experience
- identify and specify abstractions that are broader in scope than single practice areas
- provide an additional common vocabulary for understanding product lines
- are a means of documenting new and ongoing product line efforts
- help manage complexity
- can be combined to build complex product line solutions

Session 3 Contents

The Value of Patterns

Pattern Descriptions

Example Patterns

- What to Build
- Factory
- Each Asset
- Product Parts
- Product Builder
- Assembly Line
- Process
- Cold Start

Pattern Collection

Pattern Descriptions

Patterns need to be described

- in an easily understandable way
- with sufficient detail to permit you to recognize the pattern context and problem and to implement the solution
- consistently

Template for Software Product Line Practice Patterns

Name: a unique and intuitive pattern name and a short summary of the pattern

Example: one or more scenarios to help illustrate the context and the problem

Context: the organizational situations in which the pattern may apply

Problem: what part of a product line effort needs to be accomplished

Solution: the basis for the practice area pattern grouping underlying the pattern

Static: lists the practice areas in each group

Dynamics: a table, diagram(s), or possibly scenario(s) describing the relations among the practice areas in each group and/or among the groups if there is more than one

Application: any suggested guidelines for applying the pattern

Variants: a brief description of known variants or specializations of the pattern

Consequences: the benefits and any known limitations of the pattern

A Word About Pattern Dynamics

Software product line activities are iterative.

- Any relation that shows a dynamic interaction between two practice areas has to be iterative.
- The relations will vary with the pattern.
- Whatever the relation, it is always iterative.
- The arrow shown below denotes the shifting of active emphasis but NOT sequential activity.

practice area A ⟶ practice area B

Session 4 Contents

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.

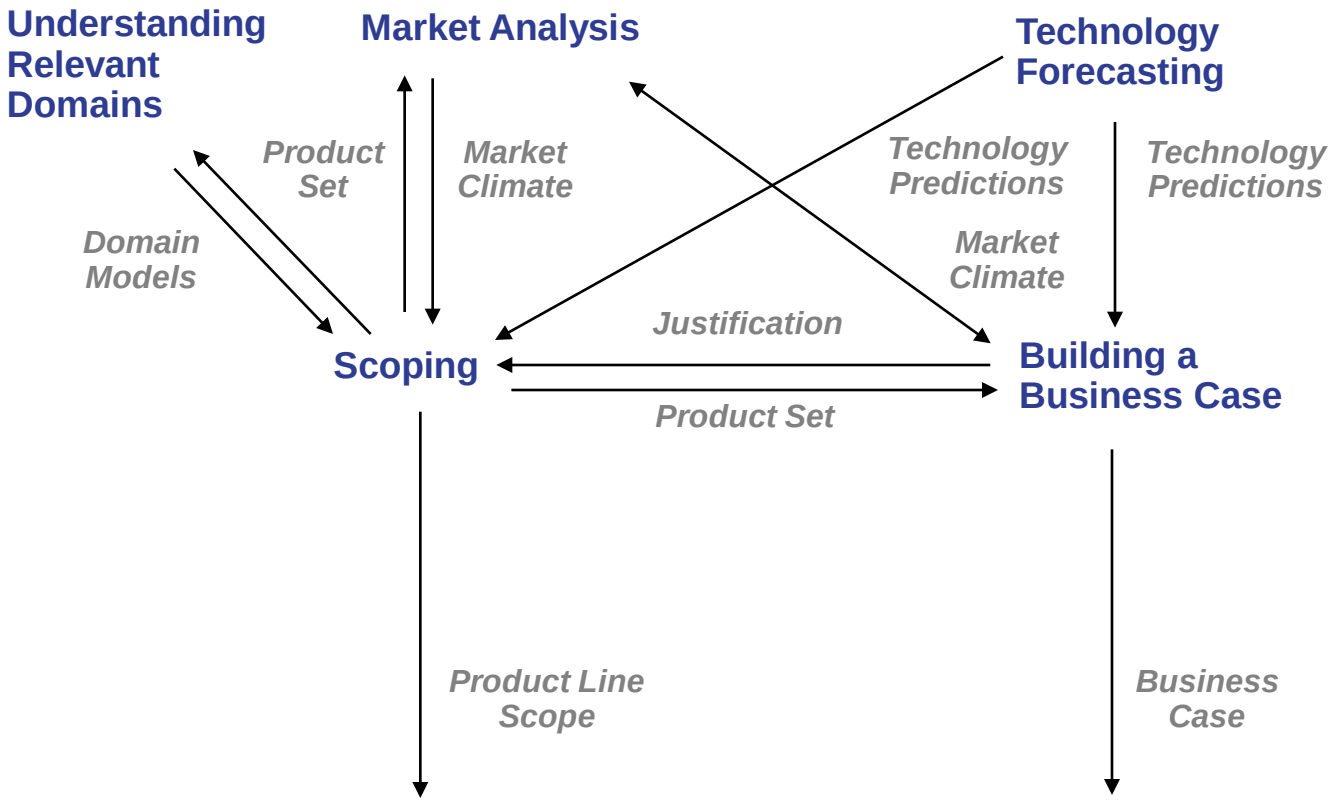
What to Build Pattern - 2

Static:

The practice areas that address the solution and that provide the structure for the *What to Build* pattern are

- Market Analysis
- Understanding Relevant Domains
- Technology Forecasting
- Building a Business Case
- Scoping

What to Build Pattern - 3



Dynamic Structure

What to Build Pattern - 4

Application:

- Practices from “Market Analysis,” “Understanding Relevant Domains,” and “Technology Forecasting” can be conducted in parallel by separate groups.
- The pattern is especially applicable to those well versed in the marketplace.
- It can be carried out by any size organization.
- It can (and should) be applied after a product line already exists.

Variants:

- Analysis pattern
- Forced March pattern

Consequences:

The *What to Build* pattern provides a handle on what practices are needed to figure out what is in the product line.

Session 4 Contents

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 products to include in the product line
- preparing the organization for a product line approach
- designing and providing the core assets that will used to construct products
- building and using the production infrastructure (plans, processes, tools)
- building products from core assets in a prescribed way
- monitoring the product line effort and making course corrections

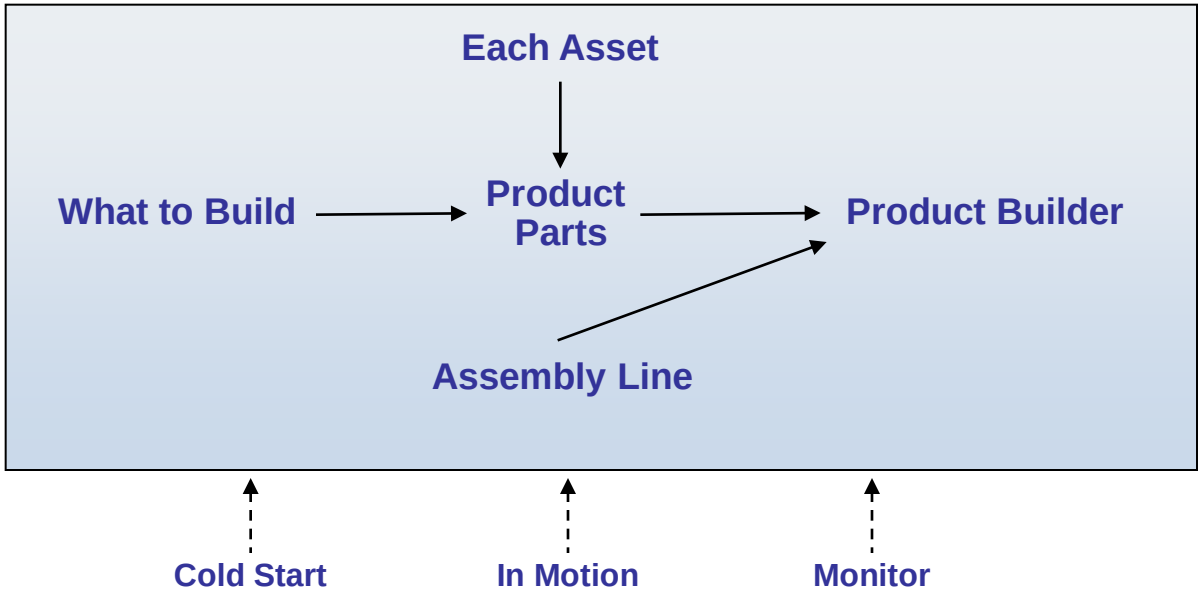
Factory Pattern - 2

Static:

The *Factory* pattern consists of the following other patterns:

- Assembly Line
- Each Asset
- Cold Start
- In Motion
- Product Builder
- Product Parts
- Monitor
- What to Build

Factory Pattern - 3



Dynamic Structure

Factory Pattern - 4

Application:

The *Factory* pattern gives a high-level view of a product line organization.

Variants:

- Adoption Factory pattern

Consequences:

The *Factory* pattern is a top-down view of the product line organization and a blueprint for a divide-and-conquer strategy.

Session 4 Contents

The Value of Patterns

Pattern Descriptions

Example Patterns

- What to Build
- Factory
- ***Each Asset***
- Product Parts
- Product Builder
- Assembly Line
- Process
- Cold Start

Pattern Collection

Each Asset Pattern - 1

Name:

The ***Each Asset*** pattern should be used whenever any asset in the core asset base is being developed.

Context:

The pattern user knows the asset to be developed, has the specifications or other necessary information for the asset, and knows who will complete the task. The person(s) to complete the task is knowledgeable in that area.

Each Asset Pattern - 2

Problem:

To use the proper set of practices to build the asset so that it will be an effective member of the product line's asset base

Solution:

The asset needs to be built in a way that is appropriate for an asset of its type and with the appropriate tools. Other necessities of asset building need to be addressed (e.g., work plan for the construction, data collection and tracking, attached process development, and testing).

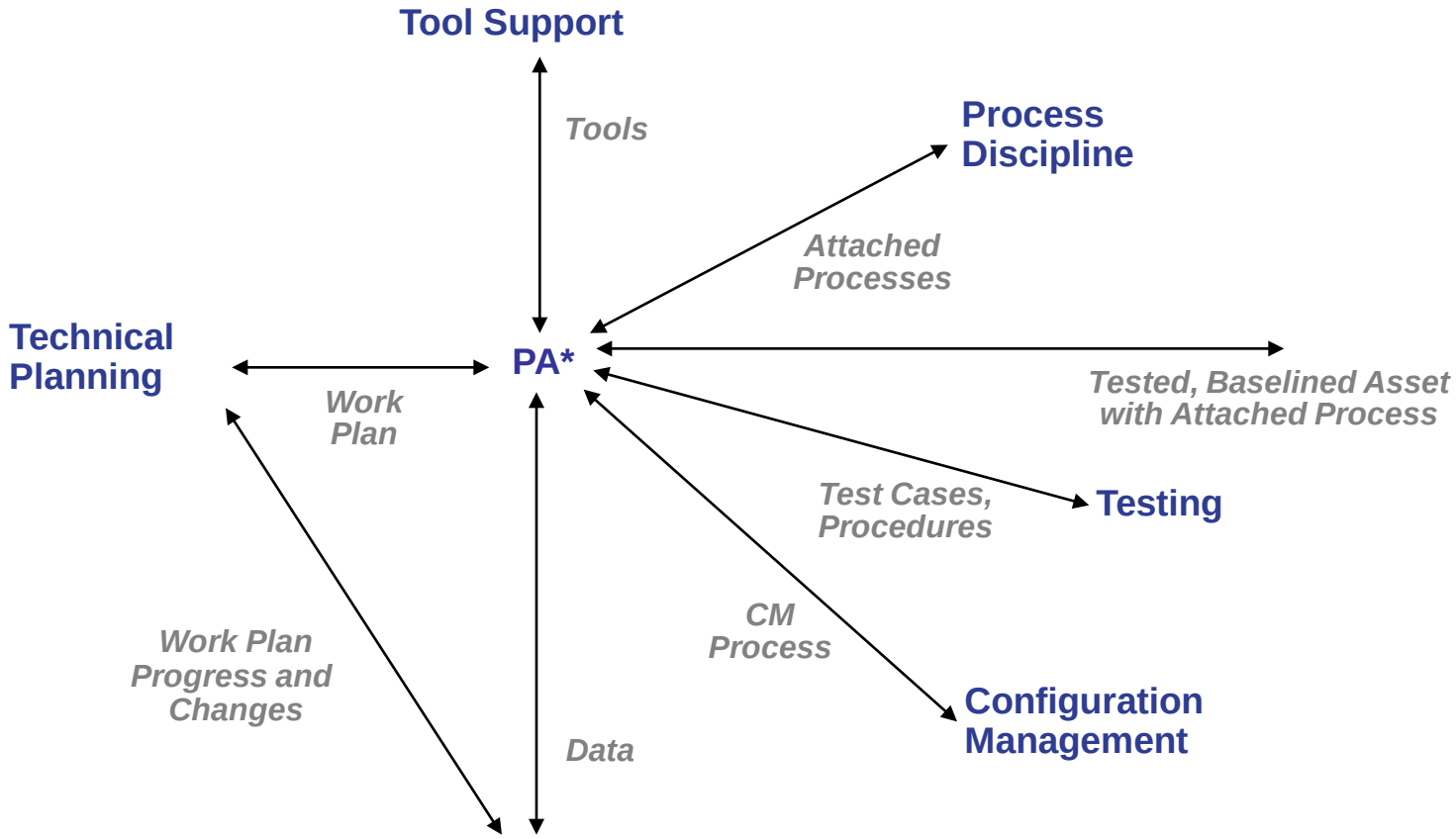
Each Asset Pattern - 3

Static:

The practice areas that address the solution and that provide the structure for the *Each Asset* pattern are

- the practice area that relates to the development of the asset in question - PA*
- Tool Support
- Technical Planning
- Process Discipline
- Testing
- Configuration Management
- Measurement and Tracking

Each Asset Pattern - 4



PA* = practices to develop the asset (e.g., "Architecture Definition" for the architecture, "Requirements Engineering" for requirements, etc.)

Dynamic Structure

Each Asset Pattern - 5

Application:

The *Each Asset* pattern is an atomic pattern that you can use for building each and every asset that is included in the core asset base.

Variants:

- Each Asset Apprentice pattern
- Evolve Each Asset pattern

Consequences:

The *Each Asset* pattern is useful in the following ways:

- It points the way in the construction of each and every core asset.
- It shows how the technical management and software engineering practice areas are blended to perform the essential activity of core asset development.
- It assumes known asset responsibility, asset specifications, and requisite knowledge.

Session 4 Contents

Product Parts Pattern - 1

Name:

The ***Product Parts*** pattern is a composite pattern. It consists of practice areas and other patterns that should be used to provide the core assets that will be part of the products in the product line.

Context:

An organization knows what products are to be included in the product line and has designated knowledgeable individuals or groups to develop* the core assets.

* *build, buy, mine, or contract*

Product Parts Pattern - 2

Problem:

To develop the core assets that will be joined together to form the products in the software product line.

Solution:

The core assets of interest to *Product Parts* are the product line requirements, the product line architecture, the components, and their test-related artifacts. Each of these core assets needs to be equipped with an attached process that describes how it will be used in the construction of products. The best source for each component needs to be determined. Individual components could be built in-house, mined from something the organization already has, bought if commercially available, or contracted out to someone else to build. Each core asset needs to be tested, and the suite of core assets needs to be integrated and tested.

Product Parts Pattern - 3

Static:

Four practice area patterns and seven practice areas address the solution and provide the structure for the *Product Parts* pattern.

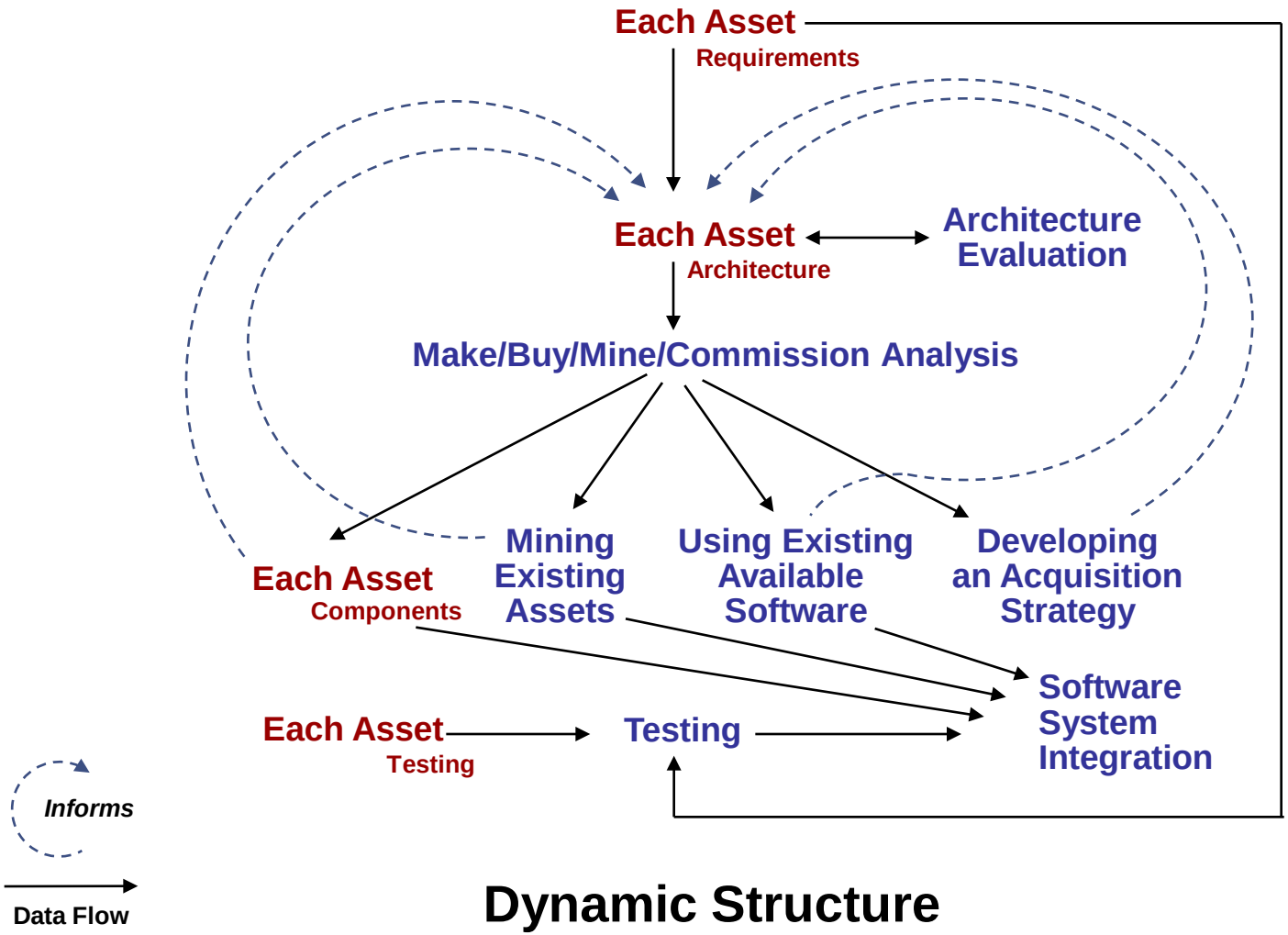
The patterns are

- Each Asset for requirements
- Each Asset for architecture
- Each Asset for components
- Each Asset for test-related artifacts

The practice areas are

- Architecture Evaluation
- Make/Buy/Mine/Commission Analysis
- Mining Existing Assets
- Using Externally Available Software
- Developing an Acquisition Strategy
- Software System Integration
- Testing

Product Parts Pattern - 4



Dynamic Structure

Product Parts Pattern - 5

Application:

- Individuals and teams with a combination of software engineering and technical management skills are required.
- The practices will get parceled out depending on the way the product line organization has been structured and the way the roles and responsibilities have been defined.

Variants:

- Green Field pattern
- Barren Field pattern
- Plowed Field pattern

Consequences:

The *Product Parts* pattern links together the practices that design and provide the parts for the products. It provides a roadmap for core asset development assuming you know the scope of the product line and have delineated responsibilities for core asset development.

Session 4 Contents

The Value of Patterns

Pattern Descriptions

Example Patterns

- What to Build
- Factory
- Each Asset
- Product Parts
- ***Product Builder***
- Assembly Line
- Process
- Cold Start

Pattern Collection

Product Builder Pattern - 1

Name:

The ***Product Builder*** pattern should be used whenever any product in the product line is being developed.

Context:

An organization has already established the production plan, the production capability, and the core asset base and has designated knowledgeable individuals or groups to develop a product that has been determined to be in the product line.

Problem:

To develop a product from the core assets using the production plan

Solution:

The production plan is followed using the established production capability to create an instance of the product line.

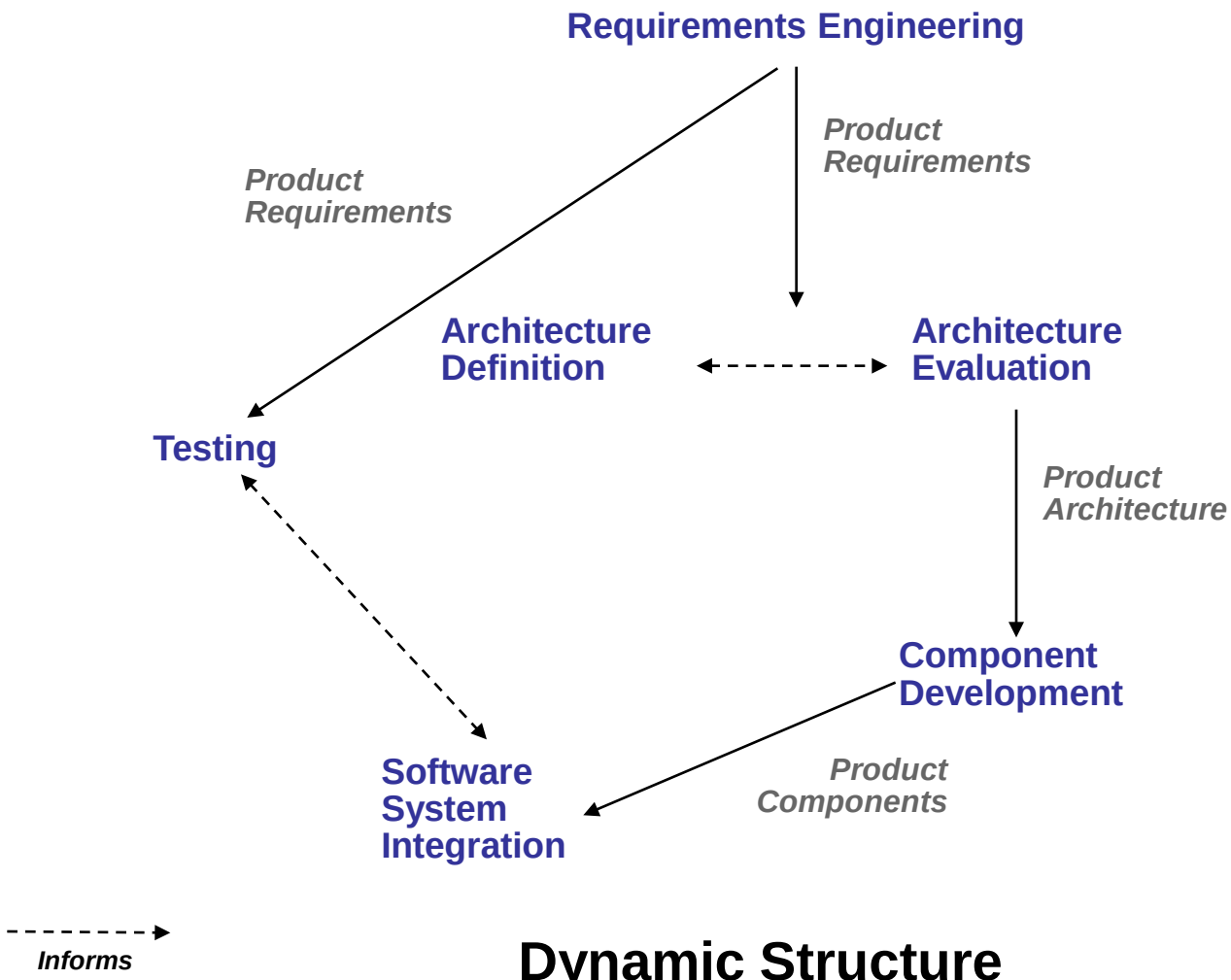
Product Builder Pattern - 2

Static:

The practice areas that address the solution and provide the structure for the *Product Builder* pattern are

- Requirements Engineering
- Architecture Definition
- Architecture Evaluation
- Component Development
- Testing
- Software System Integration

Product Builder Pattern - 3



Product Builder Pattern - 4

Application:

- involves following the production plan, which includes an attached process for the major assets
- is performed by a product development team

Variants:

- Product Gen pattern

Consequences:

The *Product Builder* pattern collects those practices that are needed to construct products.

It assumes

- production capability
- an existing asset base
- knowledgeable product developers

Session 4 Contents

The Value of Patterns

Pattern Descriptions

Example Patterns

- What to Build
- Factory
- Each Asset
- Product Parts
- Product Builder
- ***Assembly Line***
- Process
- Cold Start

Pattern Collection

Assembly Line Pattern - 1

Name:

The ***Assembly Line*** pattern should be used to set up and run the production capability of a software product line.

Context:

An organization has made a decision to launch a product line effort.

Problem:

To provide and use the tools and processes necessary to support the development of products from the product line's core assets

Solution:

The assembly line dictates how to assemble the products from their core asset parts and specifies which asset versions to use and where to find them, the schedule for the assembly, the tools, and the coordination of activities.

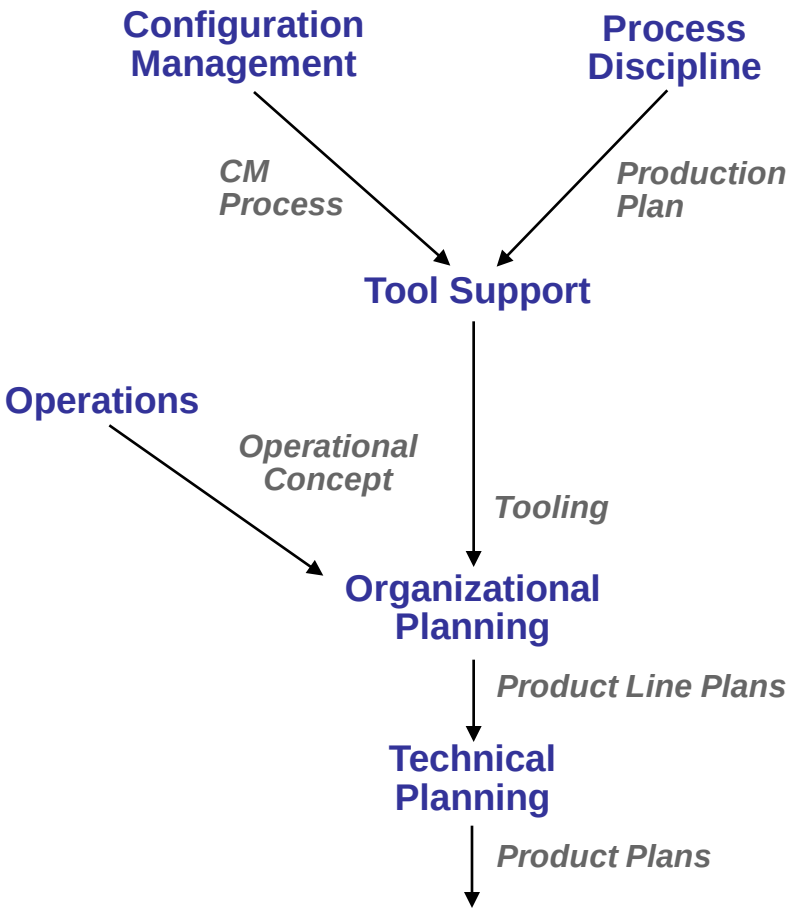
Assembly Line Pattern - 2

Static:

The practice areas that address the solution and that provide the structure for the *Assembly Line* pattern are

- Configuration Management
- Process Discipline
- Tool Support
- Operations
- Technical Planning
- Organizational Planning

Assembly Line Pattern - 3



Dynamic Structure

Assembly Line Pattern - 4

Application:

A large organization might charter a software support group carrying out the technical management practices in the *Assembly Line* pattern and assist the product managers and the product line manager.

In a small organization, the product line manager and one or two developers would perform the *Assembly Line* pattern practices.

Variants:

None known.

Consequences:

The *Assembly Line* pattern provides the handle on those practices that are needed to use the core assets routinely and efficiently.

Session 4 Contents

The Value of Patterns

Pattern Descriptions

Example Patterns

- What to Build
- Factory
- Each Asset
- Product Parts
- Product Builder
- Assembly Line
- ***Process***
- Cold Start

Pattern Collection

Process Pattern - 1

Name:

The ***Process*** pattern should be used to support all the product line activities that require processes.

Context:

An organization has made a decision to launch a product line effort.

Problem:

To develop the processes necessary to support both the development of core assets and the development of products from those assets

Solution:

A software product line approach requires processes for carrying out an assortment of activities so that those activities are performed routinely and with predictable results by one or more teams operating harmoniously.

Process Pattern - 2

Static:

There is a single group that contains practice areas and a pattern that address the solution and provide the structure for the *Process* pattern.

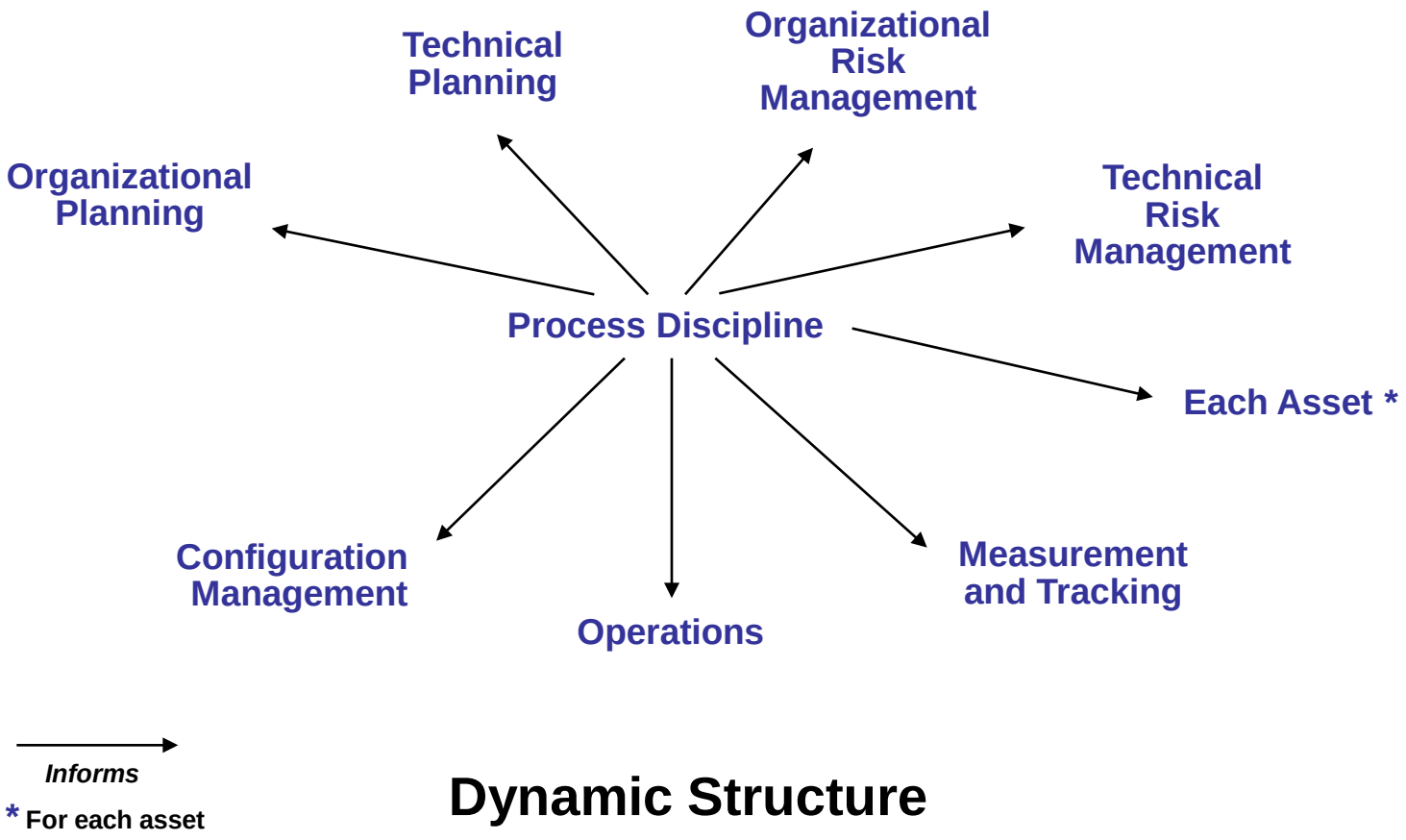
The pattern is

- Each Asset for all assets in the asset base

The practice areas are

- Configuration Management
- Measurement and Tracking
- Process Discipline
- Operations
- Organizational Planning
- Organizational Risk Management
- Technical Planning
- Technical Risk Management

Process Pattern - 3



Dynamic Structure

Process Pattern - 4

Application:

Often, in a medium-sized to large organization, a group is dedicated to process definition and improvement.

In a small organization, someone must have process definition capabilities because defined processes are important for the discipline required for product lines, even if those processes are lightweight ones for a small organization context.

Variants:

- Process Improvement pattern

Consequences:

The *Process* pattern is a building-block pattern that is required to support the product line operation. It is the scaffolding for the assembly line.

Session 4 Contents

Cold Start Pattern - 1

Name:

The ***Cold Start*** pattern should be used when an organization is launching a software product line for the first time.

Context:

An organization is launching its first software product line.

Problem:

To effectively prepare the organization for its first software product line production

Solution:

The person in charge must launch the effort—fund, staff, provide training, prepare customers, arrange for suppliers, develop a concept of operations, and create the requisite plans.

Cold Start Pattern - 2

Static:

The practice areas that address the solution and that provide the structure for the *Cold Start* pattern are

- Launching and Institutionalizing
- Funding
- Customer Interface Management
- Developing an Acquisition Strategy
- Operations
- Organizational Planning
- Organizational Risk Management
- Structuring the Organization
- Training

Cold Start Pattern - 3

Cold Start Pattern - 4

Application:

- People in charge apply the *Cold Start* pattern.
- Organizational management skills, authority, and leadership are required.
- Begin with the launching practices as a way to frame the other practices.

Variants:

- Warm Start pattern

Consequences:

The *Cold Start* pattern is very helpful to managers who have newly opted for a product line approach.

The *Cold Start* pattern relies on the authority, the commitment, and the organizational skills of the pattern user.

Session 4 Contents

The Value of Patterns

Pattern Descriptions

Example Patterns

- What to Build
- Factory
- Each Asset
- Product Parts
- Product Builder
- Assembly Line
- Process
- Cold Start

Pattern Collection

About the Pattern Collection

We have developed a useful, starter set of 12 patterns, some of which have variants.

The context for some of the patterns is universal, while for others, it is specific to organizational conditions.

There are relationships among some of the patterns.

Current Set Of Patterns

Pattern	Variants
Assembly Line	
Cold Start	Warm Start
Curriculum	
Each Asset	Each Asset Apprentice Evolve Each Asset
Essentials Coverage	
Factory	Adoption Factory
In Motion	
Monitor	
Process	Process Improvement
Product Builder	Product Gen
Product Parts	Green Field Barren Field Plowed Field
What to Build	Analysis Forced March

Pattern Coverage

The practice area patterns can help you put the practice areas into play in a manageable way, but they don't do all your work.

Moreover, using any of the practice area patterns does not constitute a waiver to exclude any of the 29 practice areas not covered.

All the practice areas are essential to a real mastery of software product lines.

Session 4 Summary

Product line practice patterns give common product line problem/solution pairs, where the problems are product line work to be done and the solutions are the groups of practice areas that must be applied to accomplish the work.

A collection of 12 product line practice patterns has been defined.

Product line practice patterns make the move to product lines more manageable.

Exercise 2

Relating the case studies to the *What to Build* pattern.

Carnegie Mellon University

Software Engineering Institute

Part 3: Putting the Practice Areas into Action
Session 5: SEI Product Line Technical Probe

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 4 Objectives

This session will acquaint participants with the

- SEI Product Line Technical Probe[SM] (PLTP[SM])
- PLTP process

[SM] Product Line Technical Probe and PLTP are service marks of Carnegie Mellon University.

Session 4 Contents

Product Line Technical Probe Overview

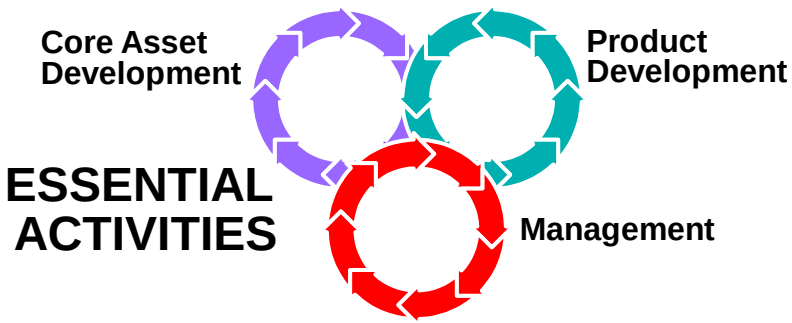
PLTP Participants

PLTP Process

PLTP Interview Questions

Other Diagnostics

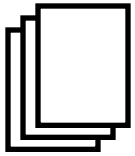
Help to Make It Happen



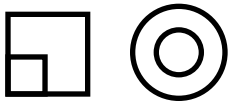
PRACTICE AREAS

Software Engineering	Technical Management	Organizational Management

GUIDANCE

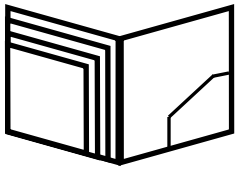


Case Studies



Patterns

Probe



Curriculum

One Way to Begin

When embarking on a new approach (such as a software product line approach), you need to know the answers to the following questions:

- Which practices lead to success?
- How does my organization stack up against those practices?

Answers to the first question may be found in the practice areas (as described in the SEI Framework for Software Product Line Practice[SM]).

Answers to the second question can result from conducting a software product line diagnostic such as the SEI Product Line Technical Probe (PLTP).

It will then be possible to chart a course from where you are to where you want to be.

[SM] Framework for Software Product Line Practice is a service mark of Carnegie Mellon University.

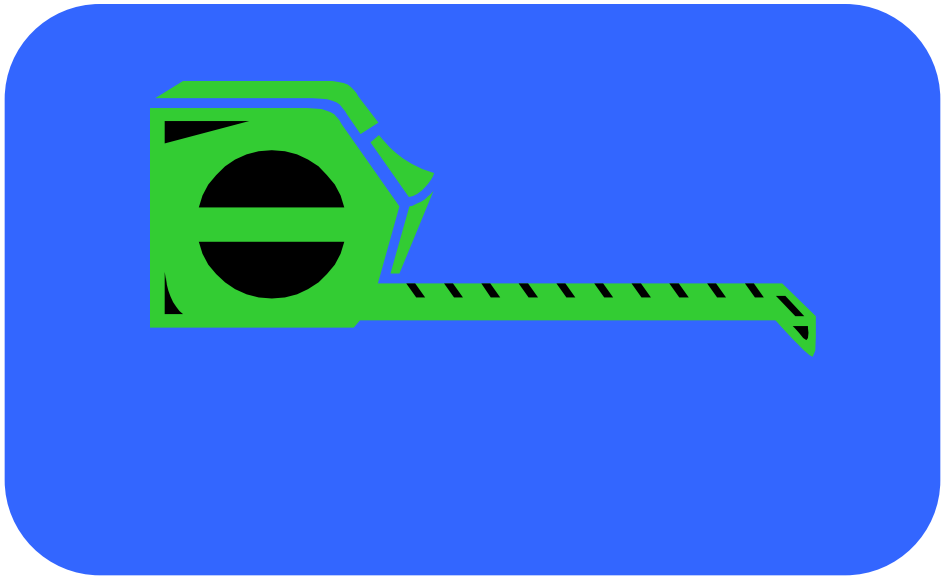
What Is a Product Line Technical Probe (PLTP)?

The 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.

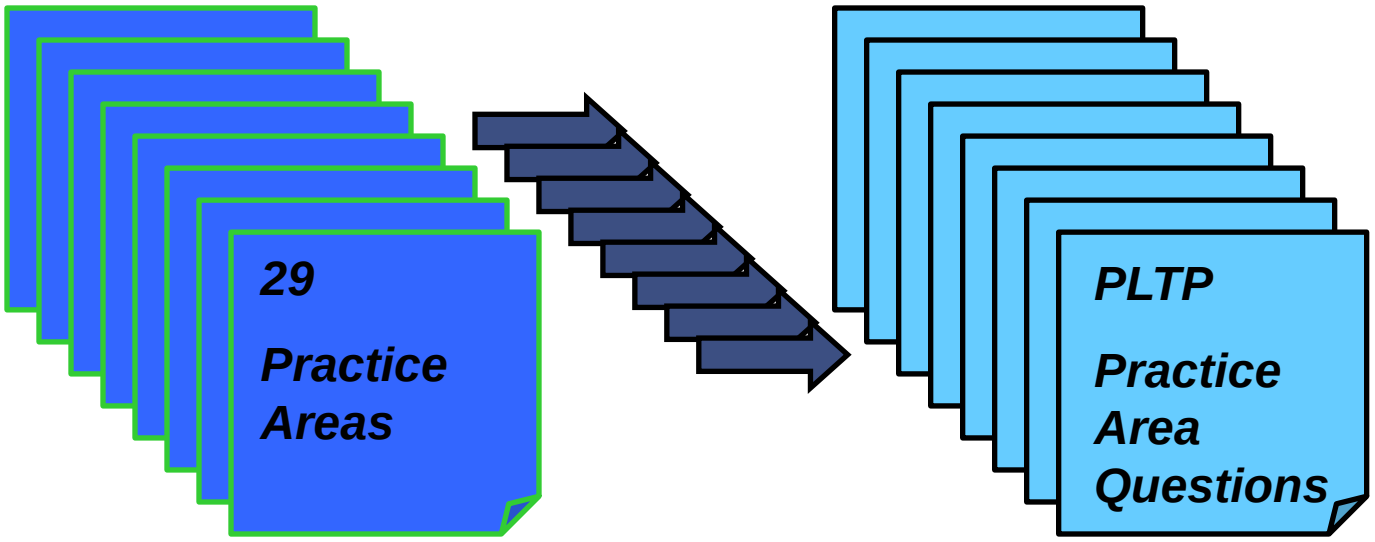
The Framework and the Product Line Technical Probe (PLTP)

The Product Line Technical Probe compares an organization's practices against the practice areas in the Framework.



Applying the Practice Areas in the PLTP

The PLTP process uses structured interviews based on questions derived from the 29 practice areas.



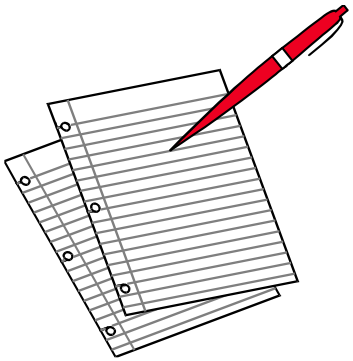
PLTP Outcomes

Set of findings that portray organizational

- strengths
- challenges

with regard to a product line approach

Findings can be used to develop an action plan with the goal of making the organization more capable of achieving product line success.



PLTP Applicability

When an organization

- is considering adopting a software product line approach
- has already initiated a software product line approach

Why Do a PLTP?

To take a baseline snapshot of the product line organization

To do a reality check

To avoid common pitfalls

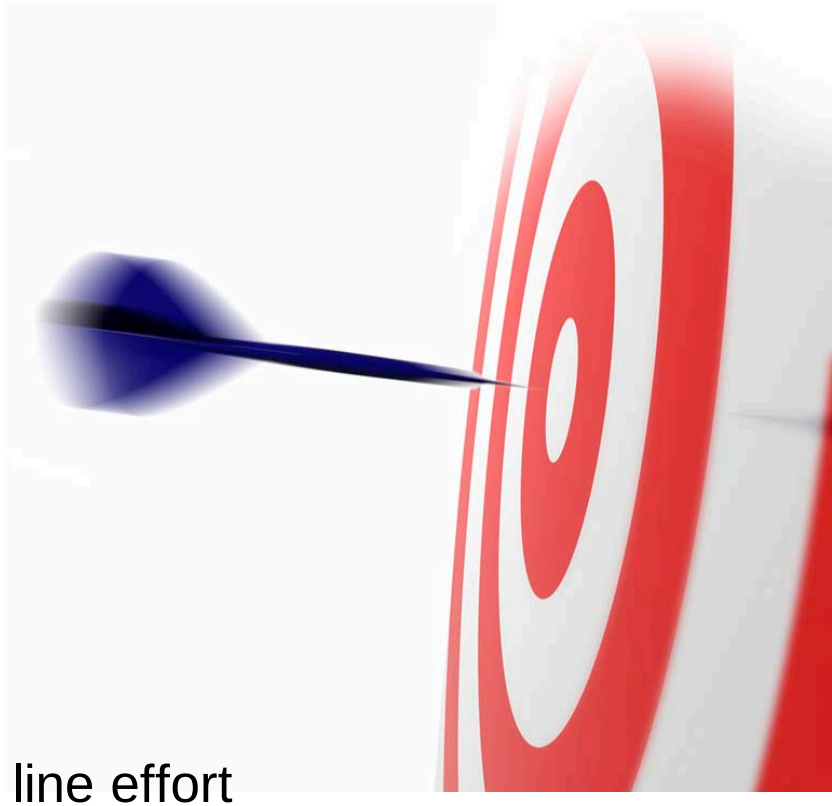
To capitalize on strengths

To shore up weaknesses

To identify and mitigate risks early

To get stakeholder buy-in

To gauge progress in an ongoing product line effort



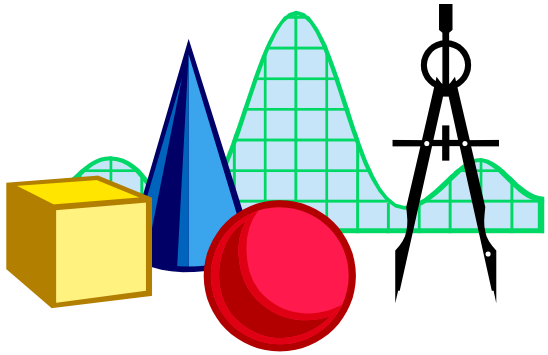
PLTP Basis

Mechanism

- SEI Software Risk Evaluation
- SEI Capability-Based Assessment
- SEI early product line evaluations

Content

- SEI Framework for Software Product Line Practice



Session 4 Contents

Product Line Technical Probe Overview

PLTP Participants

PLTP Process

PLTP Interview Questions

Other Diagnostics

Who Are the Participants?

Executives

Technical support staff

Managers

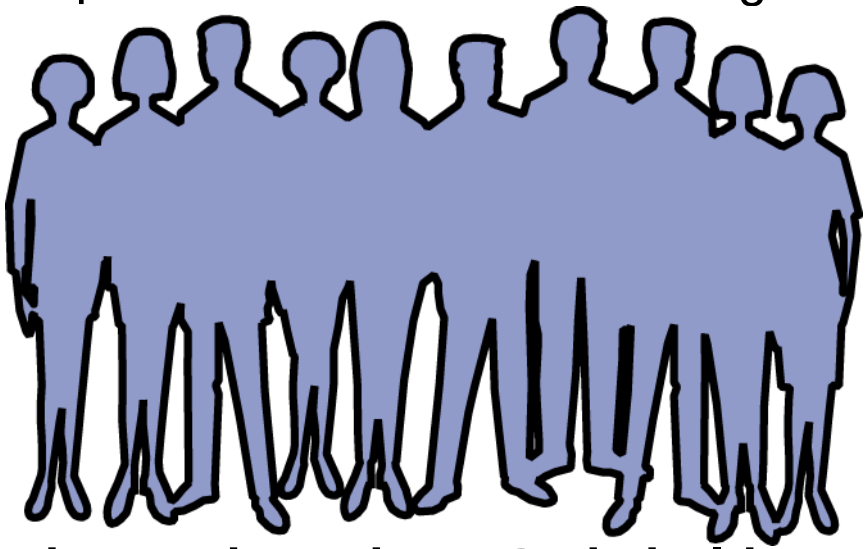
Marketers

Architects

Customers/End users

Developers

Engineers



The Product Line's Stakeholders

Session 4 Contents

Product Line Technical Probe Overview

PLTP Participants

PLTP Process

PLTP Interview Questions

Other Diagnostics

How Is a PLTP Executed?

Preliminary Phase

- one-day meeting at customer site
- probe preparation

Technical Probe Phase

- data gathering
- data consolidation and analysis
- reporting
- four days at customer site

Follow-On

- report writing
- optional: facilitated development of action plan to address findings
- optional: tailored assistance

Preliminary Phase

Determine organizational context

- goals for the product line effort
- status of the product line effort
- expectations for the PLTP

Plan technical probe execution

- determine the stakeholders for interview groups
- script interview questions from the PLTP question bank
- arrange the logistics
  - facilities
  - dates
  - schedule
  - point of contact

Preliminary Phase Meeting Topics

Overall Context

PLTP Context

Terminology Mapping

Process Context

Legacy Context

Management and Structural Context

Implementation Context

Documentation Context

Technical Probe: Data Gathering - 1

Series of structured interviews of small groups

- peer groups
  - no reporting relationships
- no preparation required
- all comments non-attributable
- questions derived from the SEI Framework for Software Product Line Practice
  - scripted based on Preliminary Phase of the PLTP

Technical Probe: Data Gathering - 2

Sample interview groups

- senior managers
- middle managers
- project managers
- technical team leads
- architects and/or senior designers
- systems engineers
- requirements analysts
- developers
- testers
- marketers
- internal customers
- support groups (quality assurance, configuration management, tool support, process group)

Technical Probe: Data Consolidation and Analysis

The PLTP team

- consolidates and compares data gathered against practice areas documented in the Framework
- identifies findings
  - strengths
  - challenges
  - other relevant information that impacts the organization's product line effort

Technical Probe: Reporting

The PLTP team reveals findings in a final presentation:

- delivered at the end of the Technical Probe phase
- to an audience designated by the organizational sponsor

Follow-On Phase

Written report delivered after the probe

Optional action planning workshop to address the PLTP findings

Optional tailored assistance in specific areas of the action plan

PLTP Final Report Contents

Context

- the SEI Framework for Software Product Line Practice
- the SEI Product Line Technical Probe (PLTP)
- the organization's context
- the PLTP applied to the organization

Summary of findings

- general observations
- overall strengths
- major challenges

Recommendations

Detailed findings

- findings and discussions for all 29 practice areas

Use of Software Product Line Practice Patterns in the PLTP

Software product line practice patterns address specific product line contexts and problems.

Patterns are used during the probe by

- finding the root cause of surfaced weaknesses
- classifying the results above the practice area level
- packaging and prioritizing the results
- packaging and prioritizing a course of action

Session 4 Contents

Product Line Technical Probe Overview

PLTP Participants

PLTP Process

PLTP Interview Questions

Other Diagnostics

Example Practice Area Questions - 1

Structuring the Organization

If a product line effort *is* already underway:

- Would you please describe your planned organizational structure for the product line effort? Include the roles, responsibilities, and size associated with each unit; for example,
  - architecture responsibility and authority
  - component engineering responsibility and authority
  - product development responsibility and authority
  - requirements engineering responsibility and authority
  - testing responsibility and authority
  - core asset evolution responsibility and authority
  - …

If a product line effort *is not* already underway:

- Would you please describe your plan for the overall organizational structure for the product line effort?
  - … (as before)

Example Practice Area Questions - 2

Requirements Engineering

If a product line effort *is not* already underway:

- How are requirements engineering activities typically planned?
- How are changes to requirements typically handled?
- How do you plan to tailor existing requirements engineering processes to address:
  - product line requirements
  - commonalities and variations in the requirements
  - product specific requirements
  - requirements changes
  - communication of changes to requirements
  - the traceability of requirements to relevant core assets

Example Practice Area Questions - 3

Requirements Engineering

If a product line effort *is* already underway:

- Would you please describe how requirements engineering activities for the product line are planned?
- Do you have a documented requirements engineering process for the product line? Please describe it?
- How are commonalities and variations in the requirements identified and modeled?
- How are requirements communicated to the architects and the component developers?

Session 4 Contents

Product Line Technical Probe Overview

PLTP Participants

PLTP Process

PLTP Interview Questions

Other Diagnostics

Other Product Line Diagnostics

Patterns can also be used to narrow the PLTP's focus (a focused PLTP is possible).

SEI Product Line Quick Look[SM] (PLQL[SM])

Bosch Product Line Potential Analysis [Fritsch 2004a]

European Union ITEA (Information Technology for European Advancement) BAPO (Business, Architecture, Process, Organization) Evaluation [van der Linden 2004a]

[SM] Product Line Quick Look and PLQL are service marks of Carnegie Mellon University.

Session 5 Summary

Product line diagnostics can help an organization troubleshoot its own product line effort.

The SEI Product Line Technical Probe (PLTP) is a diagnostic method for examining an organization's readiness to adopt, or ability to succeed with, a software product line approach.

The 29 practice areas of the Framework for Software Product Line Practice serve as a reference model for the PLTP both in data collection and analysis.

The PLTP follows a structured process based on proven mechanisms and includes a series of structured interviews of small peer groups within the organization, followed by data analysis.

The PLTP results include a characterization of an organization's strengths and challenges relative to its product line effort.

The product line practice patterns are used in the PLTP.

Carnegie Mellon University

Software Engineering Institute

Part 3: Putting the Practice Areas into Action
Session 6: Product Line Adoption Roadmap

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Session 6 Objectives

This session will introduce participants to a product line adoption roadmap that can be used to plan and track product line adoption and progress.

Session 6 Contents

Need for Adoption Support

Adoption Factory Pattern

Product Line Adoption Support

The tremendous benefits of taking a software product line approach are well documented.

Nonetheless, there are still considerable barriers to product line adoption.

The “Launching and Institutionalizing” practice area lays out what needs to occur when an organization adopts a product line approach.

A generic roadmap to product line adoption would be useful.

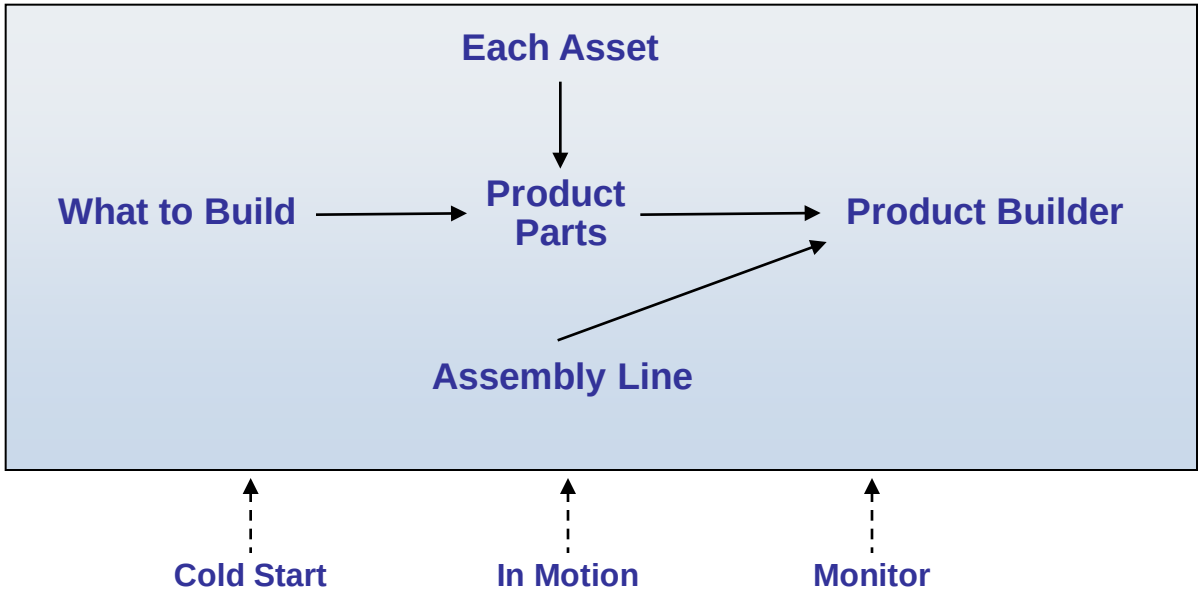
The ***Factory*** pattern can serve as the basis for such a roadmap.

Session 6 Contents

Need for Adoption Support

Adoption Factory Pattern

Factory Pattern Revisited



Dynamic Structure

A Variant for Adoption

The *Factory* pattern is already a high-level view of the entire product line organization and a blueprint for a divide-and-conquer strategy.

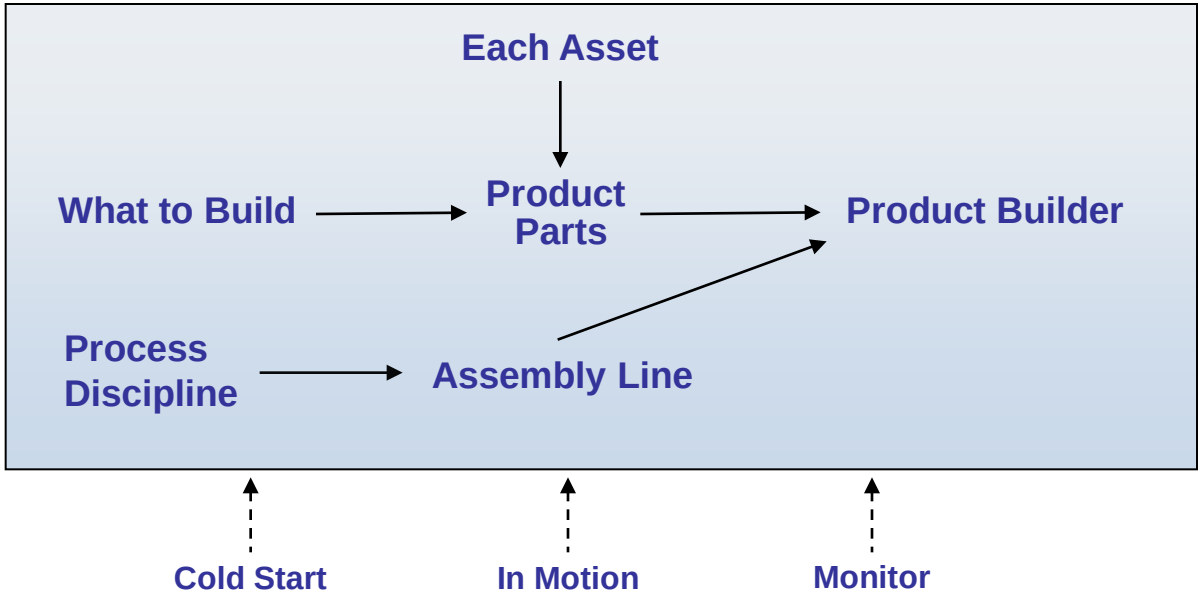
It is deficient as an adoption roadmap because

- The “Process Discipline” practice area is part of the *Assembly Line* pattern but is so fundamental that it should be called out separately.
- The *Factory* pattern lacks perspectives on timing and focus areas as well as a detailed mapping to practice areas.

To make it useful as an adoption roadmap, a variant called the ***Adoption Factory*** pattern was created where

- The “Process Discipline” practice area was added as a separate element.
- A number of different perspectives or views were created.

Adoption Factory Pattern

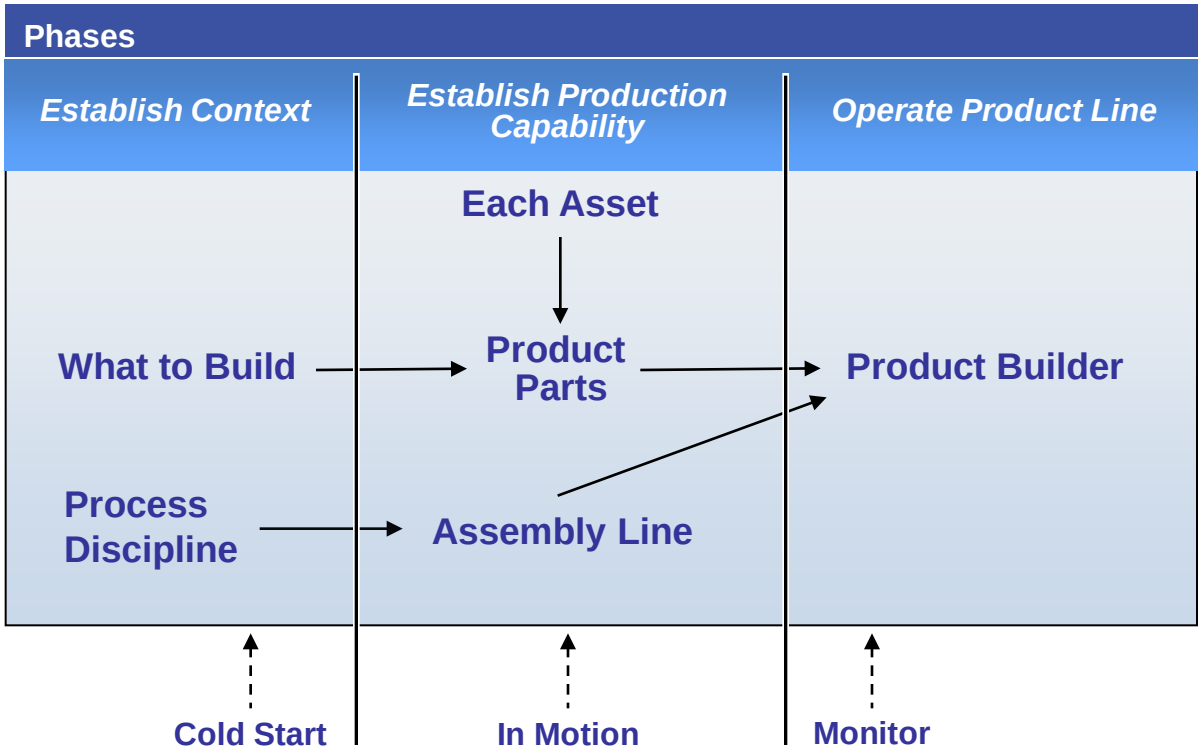


Informs and information flow

Supports

Dynamic Structure

Adoption Phases View

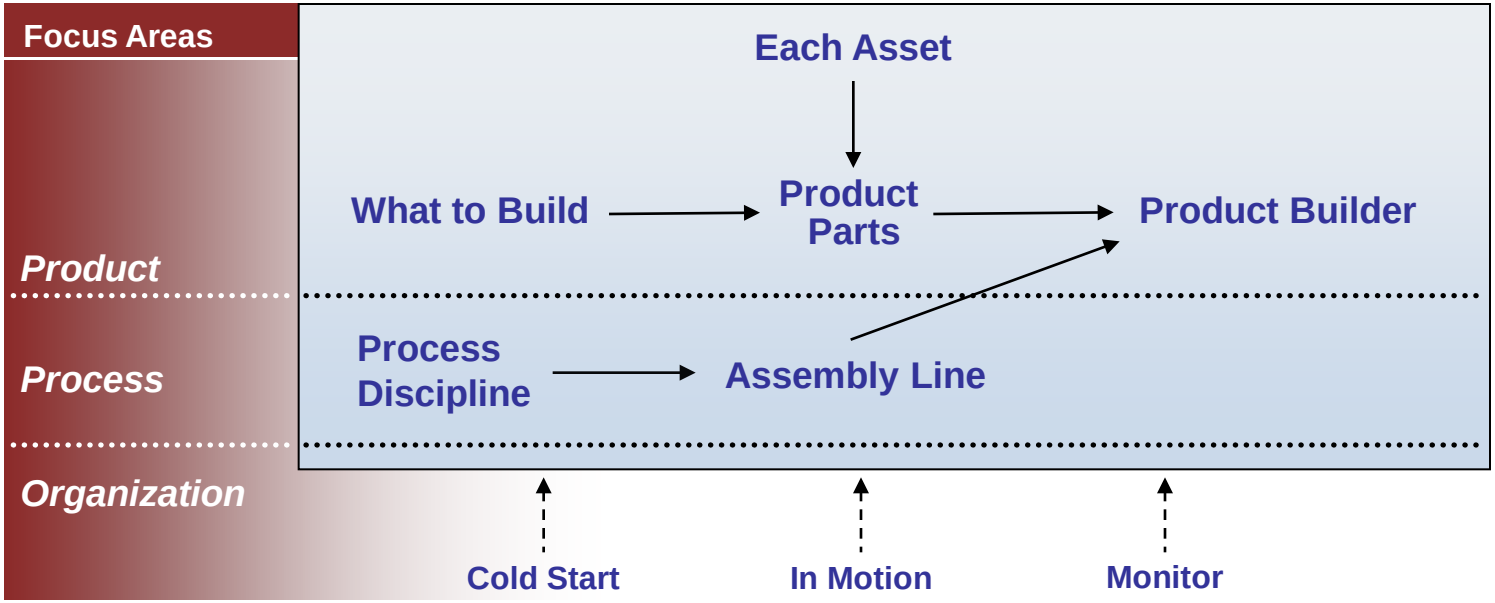


Informs and information flow

Supports

Adoption Factory Pattern

Focus Areas View

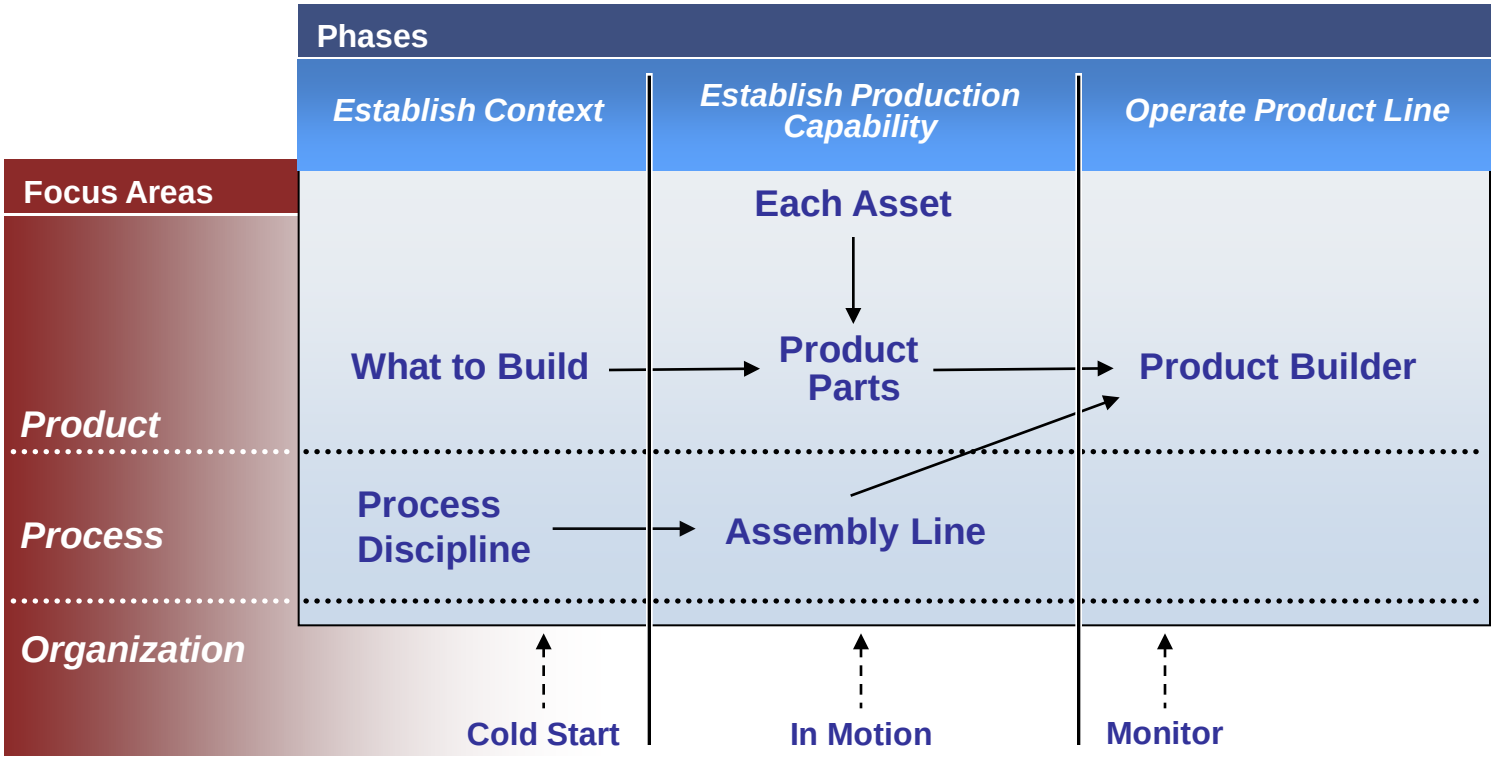


Informs and information flow

Supports

Adoption Factory Pattern

Phases and Focus Areas View



Informs and information flow

Supports

Adoption Factory Pattern

Practice Areas View

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

Using the Adoption Factory Pattern - 1

To use the *Adoption Factory* pattern as a roadmap

- Elaborate the practice areas associated with its subpatterns.
- Plan to master these practice areas in a continuous way that begins at the phase where they first appear.

The *Adoption Factory* pattern applies regardless of the adoption strategy chosen—proactive, reactive, incremental.

The *Adoption Factory* pattern also has “Roles” and “Outputs” views.

Using the Adoption Factory Pattern - 2

You can also use the *Adoption Factory* pattern to gauge where in the adoption process by phase your organization is and benchmark your activities by measuring yourself against the practice areas in that phase.

- We use the *Adoption Factory* pattern in the analysis part of the PLTP and also in framing recommendations.
- You can use the *Adoption Factory* pattern as an easily understood adoption vocabulary that can be shared across an organization and that marks organizational progress.

Cautions:

- Because of the inherent iteration in product line practices, any position in the roadmap indicates heightened awareness, not strict linear progression.
- The roadmap does not address organizational change mechanisms.
- The roadmap needs to be tailored to meet any organization-specific details and circumstances.

For Further Reading

Northrop: *Software Product Line Adoption Roadmap* [Northrop 2004a]

Session 6 Summary

Organizations need product line adoption support to lower the barriers to adoption.

A variant of the *Factory* pattern, called the *Adoption Factory* pattern, provides a useful product line adoption roadmap that can be used to plan and track product line adoption.

Carnegie Mellon University
Software Engineering Institute

Course Wrap-Up

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense

Course Structure

Course Introduction

Part 1: Software Product Line Fundamentals

Part 2: Software Product Line Practice Areas

Part 3: Putting the Practice Areas into Action

Wrap-Up

What Is a Software Product Line?

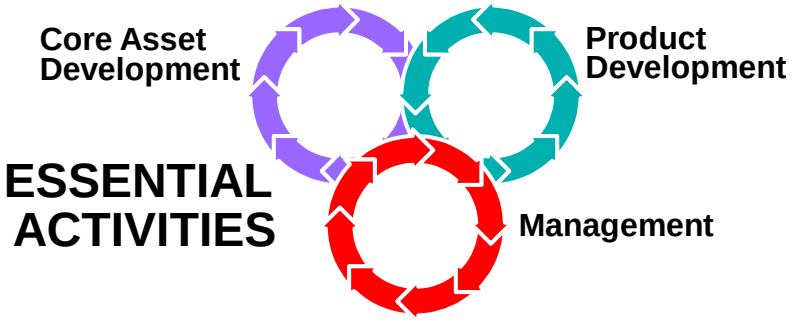
A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.

In a Nutshell

Software product lines epitomize the concept of strategic, planned reuse.

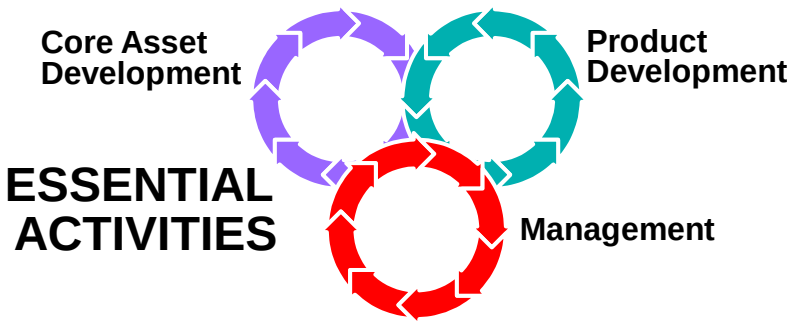
The product line concept is about more than a new technology. It is a new way of doing one's software business.

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



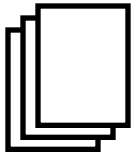
PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management

Help to Make It Happen

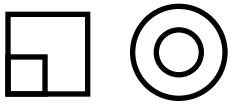


PRACTICE AREAS		
Software Engineering	Technical Management	Organizational Management

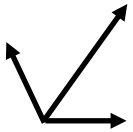
GUIDANCE



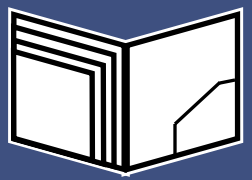
Case Studies



Patterns



Probe



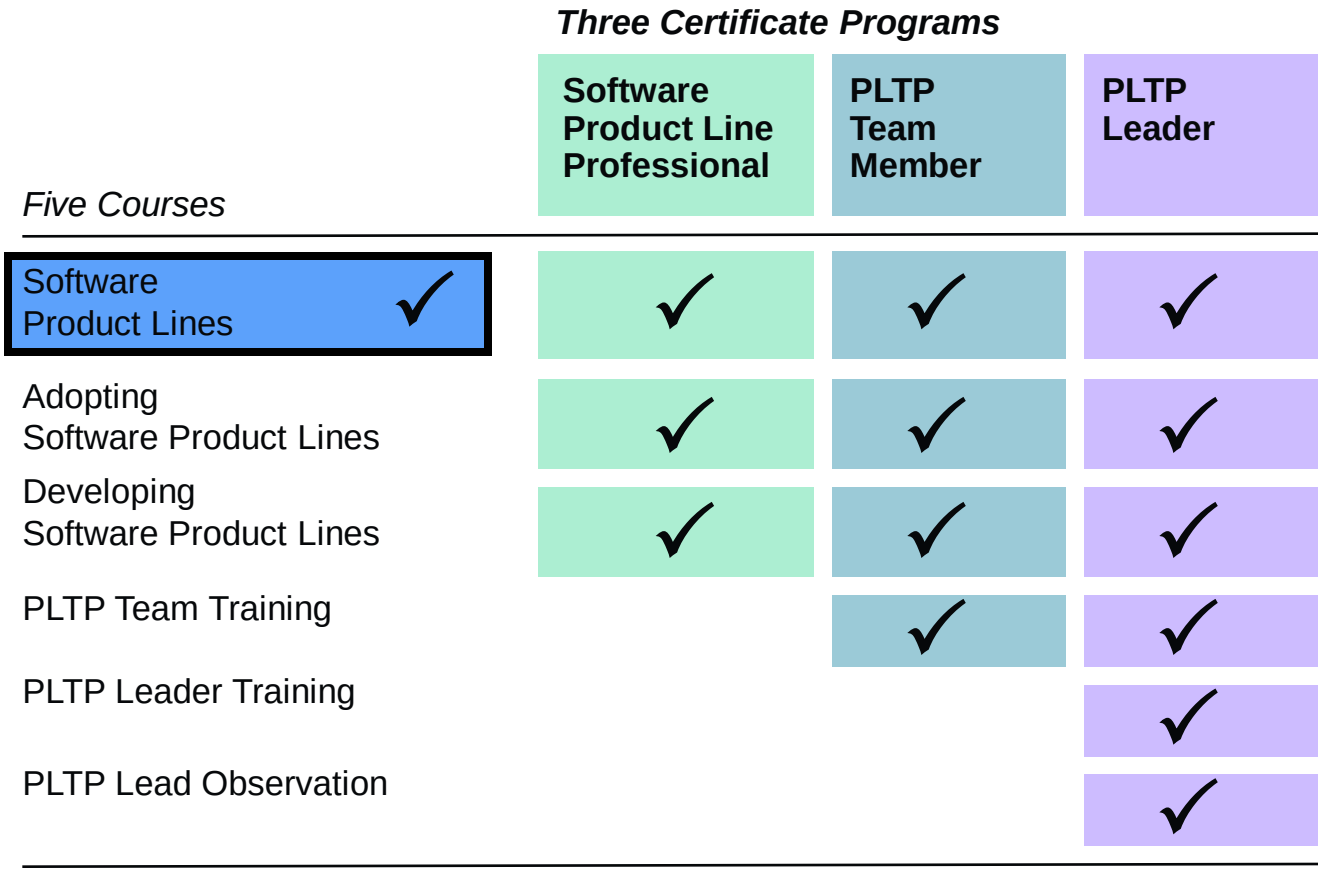
Curriculum

The SEI Software Product Line Curriculum

Five Courses	Software Product Line Professional	PLTP Team Member	PLTP Leader
	Three Certificate Programs		
Software Product Lines	✓	✓	✓
Adopting Software Product Lines	✓	✓	✓
Developing Software Product Lines	✓	✓	✓
PLTP Team Training		✓	✓
PLTP Leader Training			✓
PLTP Lead Observation			✓

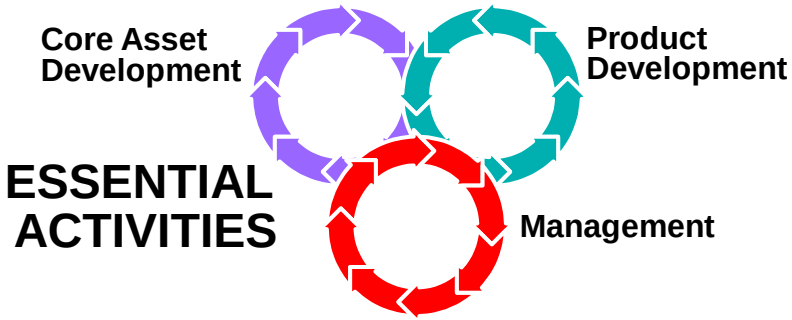
✓ **: course required to receive certificate**

The SEI Software Product Line Curriculum



✓ : course required to receive certificate

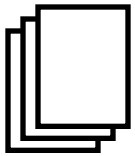
The Entire Picture

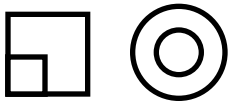


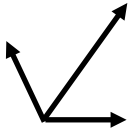
PRACTICE AREAS

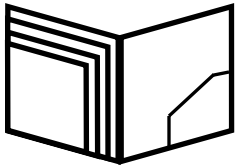
Software Engineering	Technical Management	Organizational Management

GUIDANCE


Case Studies


Patterns


Probe


Curriculum

ADOPTION FACTORY

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

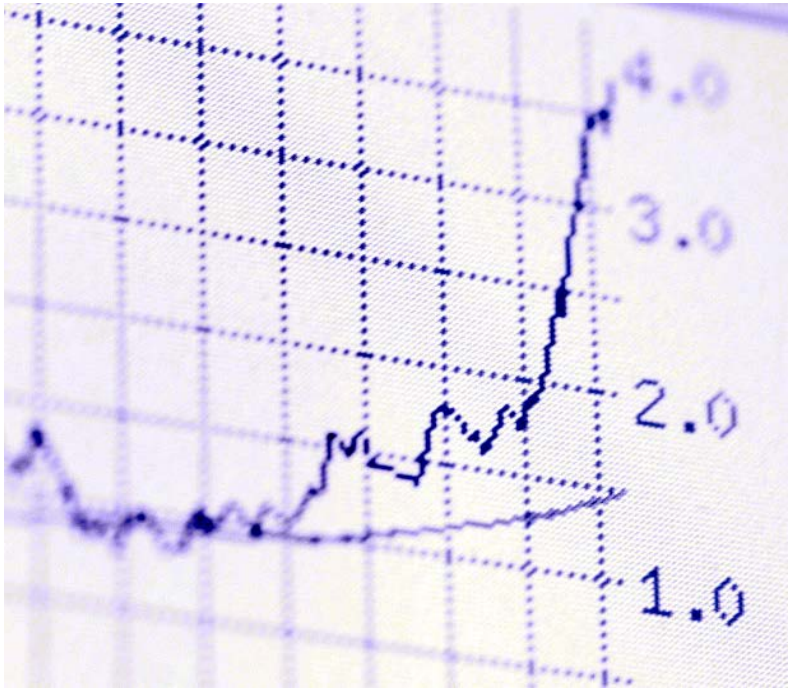
A legacy base from which to build

Architectural excellence

Process discipline

Management commitment

Loyalty to the product line as a single entity



The Truth About Software Product Lines

Unlike earlier reuse approaches, software product lines constitute a strategic business **and** technical reuse agenda.

To adopt software product line practices, an organization must adapt its

- technical practices
- management practices
- organizational structure and personnel
- business and acquisition approaches

and must embrace a software architecture-centric approach.

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
- prepare itself to institutionalize product line practices

Adoption and Institutionalization

Innovators and early adopters demonstrated the feasibility and the benefits of software product lines:

- CelsiusTech
- Cummins, Inc.
- Hewlett-Packard
- Motorola
- Nokia

The SEI and others have tried to lower the adoption barrier by codifying practices, writing case studies, perfecting methods useful in product line approaches, and engendering a software product line community.

Many organizations are now handsomely achieving their business goals using a software product line approach.

Widespread Use of Software Product Lines

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

- mobile phones
- shipboard command and control systems
- satellite ground-station systems
- avionics systems
- command and control/situational 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
- fish farm management software

The Time Is Right

Rapidly maturing, increasingly sophisticated software development technologies including *component technology, aspect-oriented technology, model-driven development, open source, and service-oriented approaches.*

A global realization of the *importance of architecture*

A universal recognition of the need for *process discipline*

Role models and case studies that are in the literature and trade journals

Books, conferences, workshops, and education programs on software product lines

Company and intercompany *product line initiatives*

A rising recognition of the ***significant benefits*** that are possible

Summary of SEI Contributions

Models and Guidance

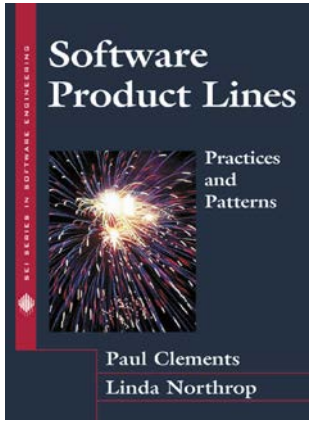
- *A Framework for Software Product Line Practice*[SM]
- product line practice patterns
- product line adoption roadmap

Methods and Technology

- product line analysis
- architecture definition, documentation, evaluation (ATAM®), and recovery
- production planning
- Structured Intuitive Model for Product Line Economics (SIMPLE)
- Product Line Technical Probe[SM] (PLTP[SM])
- Product Line Quick Look[SM] (PLQL[SM])
- 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
  product line workshops 1997 – 2005
  Army product line workshops 2007, 2009

Technical Reports, Publications, and Web Site

New Challenges and SEI Research

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 and collaborative development approaches
- globalization
- predictable assembly
- ultra-large-scale systems

Our Research: adapting software product line concepts to exploit new technologies and serve new contexts

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

Software Product Lines:
Reuse That Makes Business Sense.

Contact Information

Linda Northrop
Director
Research, Technology, and
System Solutions Program
Telephone: +1 412-268-7638
Email: lmn@sei.cmu.edu

U.S. mail:
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

World Wide Web:
www.sei.cmu.edu/productlines

Customer Relations
Email: customer-relations@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257