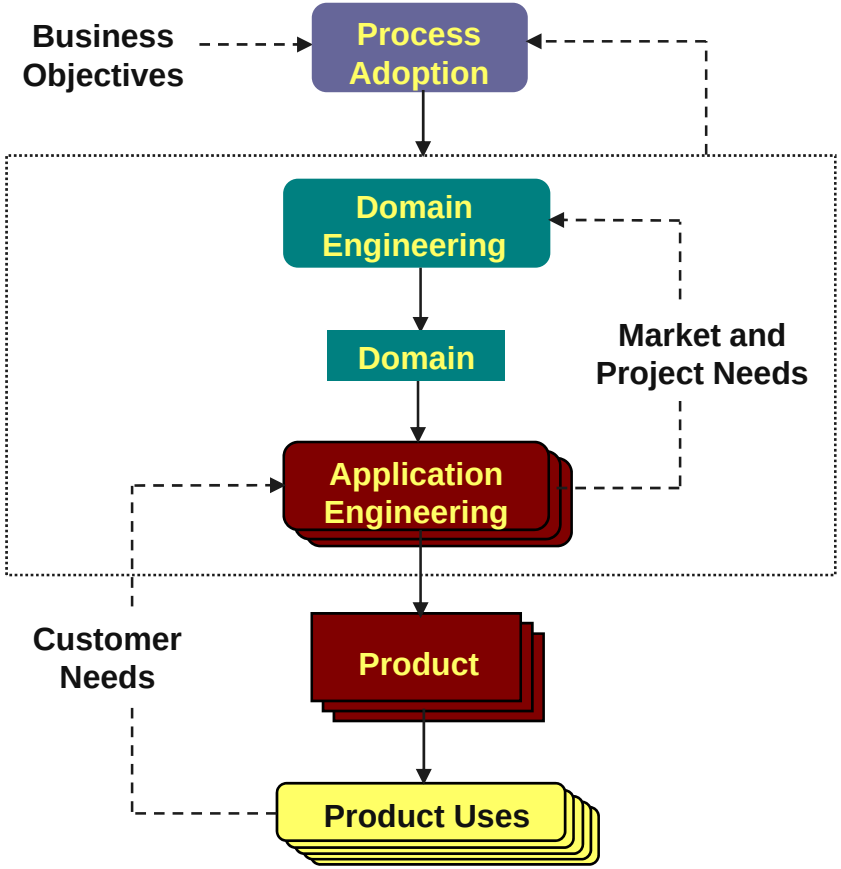
SPLC - 9 September 2008

Renewing the Product Line Vision

Grady Campbell

A Product Line Process: Domain-specific Engineering



Institute & improve a product line business

Develop and evolve a capability for building similar products

Build customized products for customers

source: **www.domain-specific.com**

The Original Product Line Vision

A framework for solving the problem of software productivity and quality:

Domain-specific: A focus on a domain represented as a family of products, all being similar but differing in well-defined ways

Streamlined production: Product building reduced to the resolution of decisions that correspond entirely to the ways in which family members can differ

Adaptable assets: Dependence on the mechanical derivation of tailored components from applying decisions to an adaptable form of reusable assets for the construction of all work products

Model-based analyses: The use of model-based analyses for help in understanding which decision choices provide the best product

source: **Reuse-driven Software Processes Guidebook (1993)**

Definitions

Domain - The knowledge (product family) and expertise (process) required to build a particular type of product

Model - A representation of a [product] that is sufficient to provide approximate answers to a designated set of questions about the represented [product]

Abstraction - A concept that denotes criteria for membership in a set (i.e., the characteristic function for a subset)

Family - (1) A set consisting of all instances that satisfy an associated defining abstraction (2) In mathematics, a set of functions that can be generated by varying the parameters of a general form

Product Family - A set of products that provide similar solutions to an envisioned set of similar problems

Product Line - A set of products having similar capabilities to address differing needs of customers in an organization's targeted market

Tracing the Product Line Concept

NRL Software Cost Reduction: disciplined software engineering methods, semi-formal requirements, information hiding module design {1981-82}

Spectrum: decision-model for product specification; adaptable assets; decision-based mechanical derivation of product {1984-88}

Dijkstra and Parnas: program family concept {1972, 1976}

Domain analysis concept: limiting scope requires market focus {1980's}

Synthesis/Reuse-driven (Software) Processes: commonalities and variabilities, total-product focus, family of processes, system product lines {1990-94}

Domain-specific Engineering: integrated adoption process, adaptability of non-text forms {1996-2002}

Neglected Aspects of the Vision

1. The decision model as product discriminator
2. Adaptable components for abstraction-based reuse
3. A domain-specific process for best fit to organizational needs
4. Total product generation to minimize producer effort
5. Model-based validation and verification to ensure quality and fitness for use of products
6. A comprehensive adoption-improvement framework for systematic organizational transition

The Role of Decisions

Engineering is a decision-making process (different decisions result in a different product).

A product family shows how different ways of resolving a set of decisions lead to different products.

Decisions represent:

- Customer choices (needs and constraints)
- Unresolved engineering tradeoffs

A focus on similar problems (a family) enables standardization, reducing number, variety, and complexity of decisions.

A “decision model” enables condensing the customer-developer dialog to its essentials, those decisions that are sufficient to distinguish among the members of a product family and identify a particular product in a domain

Decision Model Perspectives

Based on commonalities and variabilities, is there an optimal decision set that allows a customer to specify/select any properly derivable product?

Is there a decision-based process and presentation of the problem-solution (an application model) that supports effective customer decision making (envisioning and evaluating alternatives)?

Given a decision model and conforming application model, is there sufficient information and means to generate an acceptable product without intervention?

Given a derived product, is there the means to trace needed changes or revisions back to decisions and underlying domain knowledge?

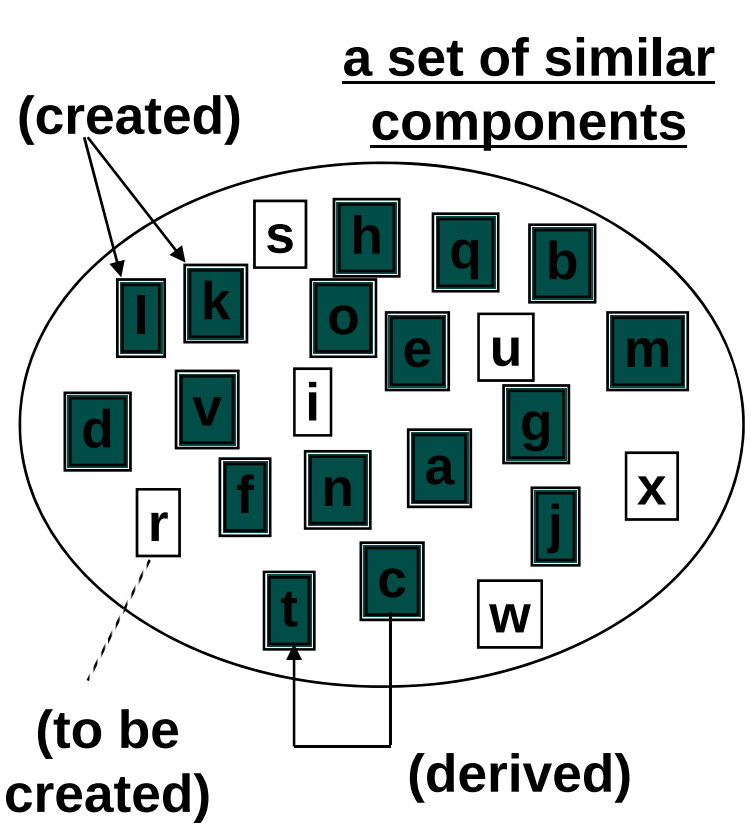
A Basic Tenet for Effective Reuse

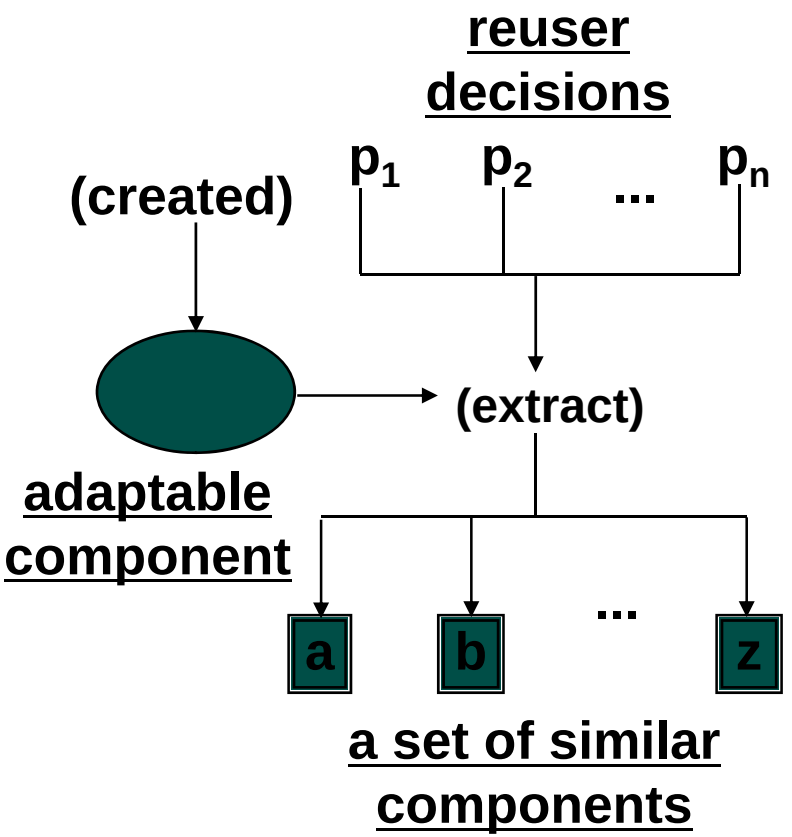
The only sound basis for reuse is an envisioned set of similar products: a family

- Similarity comprises
  - Commonality: the basis for standardization of work products and process
  - Variability: the flexibility needed to accommodate different needs
- Adaptability requires
  - An explicit representation of similarity
  - A characteristic set of deferred decisions that distinguish among the members of a family

An adaptable component expresses many versions of a work (the instances of a family) in a single unified representation.

Adaptable Components





The Elements of an Adaptable Component

An abstraction: What is the intended purpose of these components? (formulating a family)

Parameters (traceable to decisions): Why is there a need for more than one of these components? How are they different from each other?

A definition: Given a set of parameter values, what are the steps to derive a corresponding instance component?

Alternative formulations:

- Descriptive (metaprogramming notation plus generator)
- Prescriptive (domain-specific generator)
- Dynamic-Interpretive or runtime branching (virtual instantiation)

Adaptable Component Perspectives

What technology is needed for building and using adaptable software?

How is a component family or its instances verified (review, testing, formal methods)?

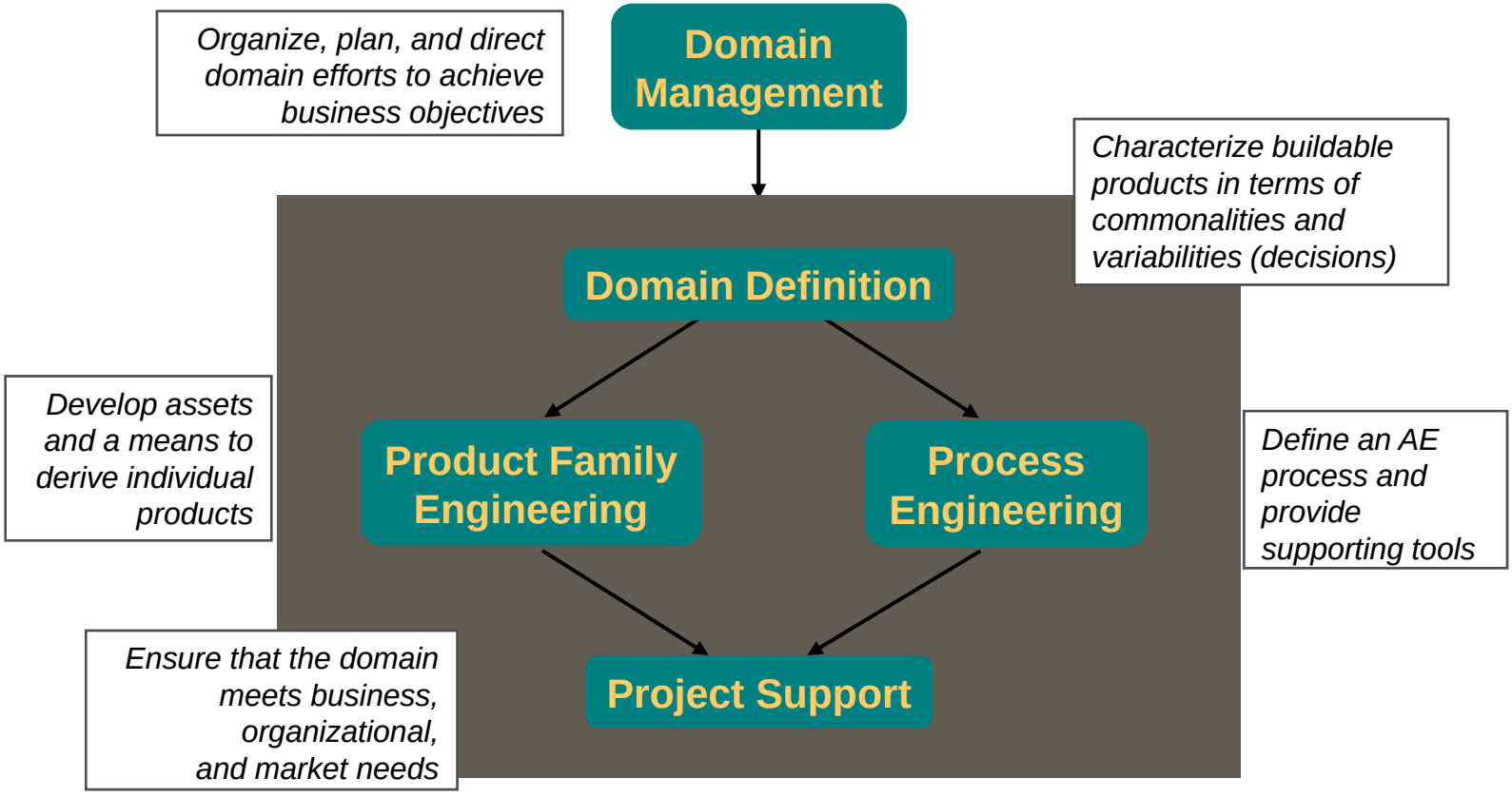
How should adaptable component developers anticipate potential domain evolution?

An adaptable component expresses an abstraction that suggests “natural” variations (alternative implementations); how are these correlated to and guided by a decision model?

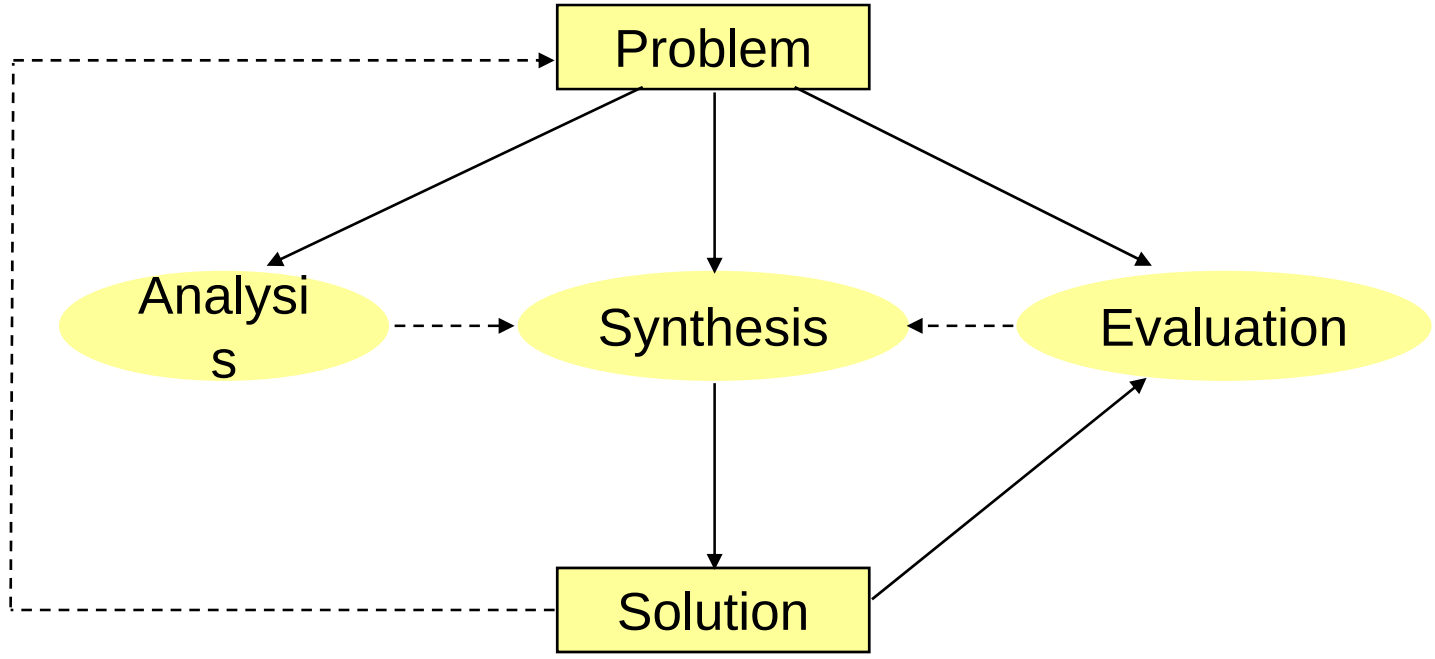
How are dependencies among adaptable components defined and managed?

How are errors in executing software diagnosed with respect to source adaptable components and instantiation decisions?

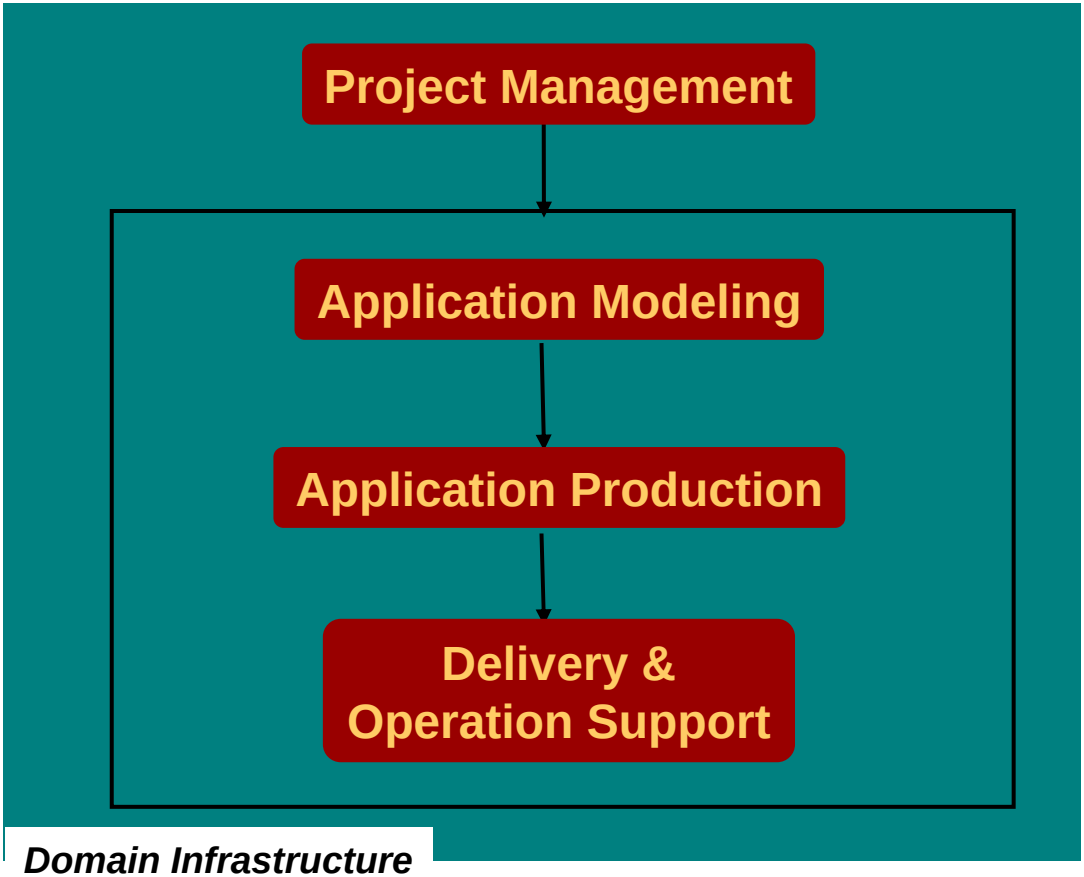
Domain Engineering for a Domain-specific Process



An Idealized Application Engineering Process



A Streamlined Application Engineering Process



Planning & Coordination

Product Specification & Validation

Product Generation & Verification

Product Distribution

Domain-specific Process Perspectives

Is a streamlined process achievable?

- Does the organization value improvements that require change?
- To what degree do external constraints impose a suboptimal process?

Is a different process worth the effort?

- What will the transition cost in time and business risk?
- How much will the process improve responsiveness to customers?

To what degree can the process be automated and tool supported?

What are the factors in defining a domain-specific process?

- Essential activities
- Required work products
- Application product iteration and feedback
- Domain (market and product family) evolution

Total Product Generation

Generators for the total product, with consistent decision-driven customizations, changing as a whole over time:

- Plans
- Requirements specifications
- Design specifications (architectural and component-level)
- Customized code
- Reviews, tests, formal specifications
- Domain-specific testing infrastructure (operational environment simulation)
- User documentation
- Installation and support materials

Total Product Generation Perspectives

What are the work products that comprise a total product?

Can work products be standardized sufficiently to avoid hand tailoring?

What are the mechanisms for transforming customer problem-solution descriptions into software?

How is the software evolved and deployed as customer descriptions of it change?

How are other (non-code) work products produced and maintained consistent with the software?

Model-based Validation and Verification

Essential properties of a product may be complex, interdependent, and non-localized.

Expected properties express an underlying model of how a product is expected to work in its operational environment.

Similar products will have similar expected and actual properties. (?)

A product family is an explicit basis for leveraging verification and validation methods:

- Properties should vary systematically across the instances of a product family.
- Property differences may be a function of the family's decision model.
- Product quality (adherence to expected properties) may improve (across instances) based on reuse.
- A family as a whole may be verifiable with respect to some properties.
- Given a product family and a validated application model, the resulting derived product should be valid if it can be verified as satisfying the application model.

Validation-Verification Perspectives

What properties are required of a product? How does this vary across the product family?

What are the interdependencies among relevant properties?

How is satisfaction of properties affected by customer decisions?

Can properties be validated with respect to a product family as a whole, and by implication with respect to an application model?

If a product family and application model can be asserted to satisfy certain properties, can the resulting product be more easily verified for these properties?

Adoption/Improvement Process

A product line expresses the market focus of a business organization.

Organizational objectives and capabilities inform product line management choices.

A product line approach generally implies substantial changes in the behavior of an organization.

These implications warrant a systematic evaluation of alternatives and careful transition when changes are required:

- Domain viability -> Product line market focus
- Process maturity -> Engineering discipline
- Process capability -> Manufacturing discipline
- Product line strategy -> Domain-specific engineering

Iterate as organizational objectives and capabilities and the market evolve.

Adoption/Improvement Process Challenges

Is a product line approach justified for a business area? (opportunity; expertise; commitment)

What constitutes readiness to take a product line approach? (relevant experience; process maturity)

What sort of product line approach suits an organization's needs and capabilities? (market/business objectives; organization; process capability; tools and methods)

What must an organization do to transition how it works? (policies and procedures; planning; education; technology infrastructure)

An Extended Vision: Producibility

The ability to deliver needed capability in a timely, cost-effective, and predictable manner

Developer productivity (efficiency and effectiveness)

- Domain knowledge and expertise, effective methods, multi-discipline integration
- Engineering discipline, process capability, systems-software engineering synergy
- Technology base (applicability, effort reduction)
- Leveragable resources (legacy, COTS, open source, domain-specific)
- Addressing uncertainty, diversity, and change

Product value (utility and quality)

- Functionality cost-effectively responsive to business/mission needs
- Quality attributes as determinants of system properties
- Compatibility with system and operational environment

Acquirer acuity (insight and foresight)

- Producibility-enabling acquisition policies and practices
- Effective technical direction, oversight, and feedback
- Mechanisms for capability-cost-schedule predictability and tradeoffs
- Infrastructure for technology development, evaluation, transition-into-use, and evolution

A Reference Vision for Producibility

CAD/CAM for Software-intensive Systems

Model-centric – All problem/solution information is represented in a comprehensive multi-faceted product model

Virtualized – A system is defined by building, pre-deploying, and validating it in software within a hardware/software virtual environment

Predictable – Software and dependent system properties of interest are able to be accurately predicted and mutually optimized

Decision-focused – Multiple alternative solutions can be modeled, produced, and empirically evaluated based on identified customer and engineering choices

Evolvable – The problem/solution model can be continuously evolved to create product variants that meet anticipated differing or changing needs

Producibility Research and Transition Themes

Model-based development

Bridging the conceptual gap between customers and product developers to rapidly formulate, build, and evaluate alternative solutions to evolving needs

Predictable software attributes

Measuring, predicting, and controlling SiS software properties and tradeoffs

System virtualization

Creating virtualized environments for realistically evaluating solutions

Disciplined methods

Applying effective methods for engineering discipline in the development of software within systems

Infrastructure and emerging technology

Exploiting changing infrastructure and computing technology capabilities

Validation

Demonstrate the applicability and practical value of research results for building software-intensive systems

Integration and Productization

Engineer research results into integrated engineering tools and methods suitable for production use

Adoption

Facilitate the adoption of productized producibility technologies by acquisition programs