

**Prosperity
Heights
Software**

**Variability 2026
Tutorial**

29 September 2026

***The Engineering and
Manufacture of
Software-based Products***

Grady H. Campbell, Jr.

gradycampbell@domain-specific.com

Copyright © 2026, Prosperity Heights Software LLC. All Rights Reserved.

Objective

Promote an Evolved Methodology for Developing Software-based Products

- **Process Innovation for Producibility**
- **A Broader Enterprise Perspective**
- **Standardization based on Process and Product Similarity**
- **Diversity and Flexibility through Constrained Customization**

Draft book (2026): domain-specific.com/EMSP

{based on Spectrum (1984), Synthesis (1990), DsE (1998)}

Tutorial Topics

- 1. Overview: A Software Methodology for Producibility**
- 2. Software Engineering for a Singular Evolving Product**
- 3. Domain-specific Engineering (DsE) for an Evolving Product Family**
- 4. Automation and Future Technologies**

Topic 1

Overview: A Software Methodology for Producibility

The Concept of Producibility

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

=> Predictably build high-quality products

- **A product is conceived to provide capabilities that enhance a customer's ability to achieve their objectives**
- **Every product has many potential realizations, both initially and as circumstances change over time.**
- **Factors enabling producibility goals**
 - **Developer Productivity** (*efficiency & effectiveness*)
 - **Product Value** (*utility & quality*)
 - **Customer Acuity** (*insight & foresight*)

Basic Terminology

- **Abstraction** – A concept that denotes criteria (e.g., similarity) for membership in a set
- **Family** – (1) A subset (of a population) whose instances satisfy an associated abstraction
(2) In mathematics, a set of functions that can be generated by varying the parameters of a general form
- **Coherent Market** – Customers having similar needs
- **Product Family** – An envisioned set of similar products
Dijkstra, E.W. Notes on Structured Programming: On Program Families, [EWD249](#), 50–52, Apr 1970.
- **Product Line** – (1) Realized instances of a product family
(2) A set of related products
- **Domain** – The knowledge (product family) and expertise (process) needed to build a customized product
- **Model** – A representation of a [product] that is sufficient to provide approximate answers to a designated set of questions about that [product]

Methodological Themes

- **Uncertainty and change are fundamental concerns**
- **Two formulations**
 - Singular evolving products
 - An evolving product family and its instances
- **Shared aspects for engineering discipline**
 - A product model *{facets and constituent elements}*
 - ▶ Separation of concerns based on scope of authority (competence & information needed)
 - ▶ Consistency across elements
 - A flexible iterative process
 - ▶ Collaboratively Concurrent
 - ▶ Method–Agnostic

Why A Concurrent Process

Disciplined Flexibility

- **How a Sole Developer Tends to Work**
- **How Conventional Development Actually Works (When It Works)**

Task Ordering Influenced by:

- **Availability of Competence** *{Knowledge+Experience+Expertise}*
- **Problem/Solution Unknowns and Uncertainties**
- **Product Model Facet/Element Dependencies**

Other Considerations:

- **Specifications <-> Realizations**
- **Management need to track progress**

{Parnas, D., Clements, P.: A Rational Design Process: How and Why to Fake It, IEEE Transactions on Software Engineering, SE-12(2), 251–257 (1986)}

Singular Product Formulation

- **An independently-managed project**
 - Targets a single customer or “simple” market
 - Operates in terms of a Software Project Model
 - Builds a Product Model for an evolving software product that best approximates customer-needed capabilities
- **Product Versions and Variants $\approx$ a Product Family**
- **A Transition Path to a Product Family Formulation**

Product Family Formulation

- Enterprise investment for market advantage
- Defines a Domain
 - Targets a coherent market
 - An envisioned set of similar products
 - Product diversity –> deferred decisions
(why a market needs multiple products)
 - A decision-focused process (and automation) for the manufacture of customized products
- Rapid Development of Singular Products
 - Product family • decisions –> Product Model
 - Automation and common content to streamline project effort

Topic 2

Software Engineering for a Singular Evolving Product

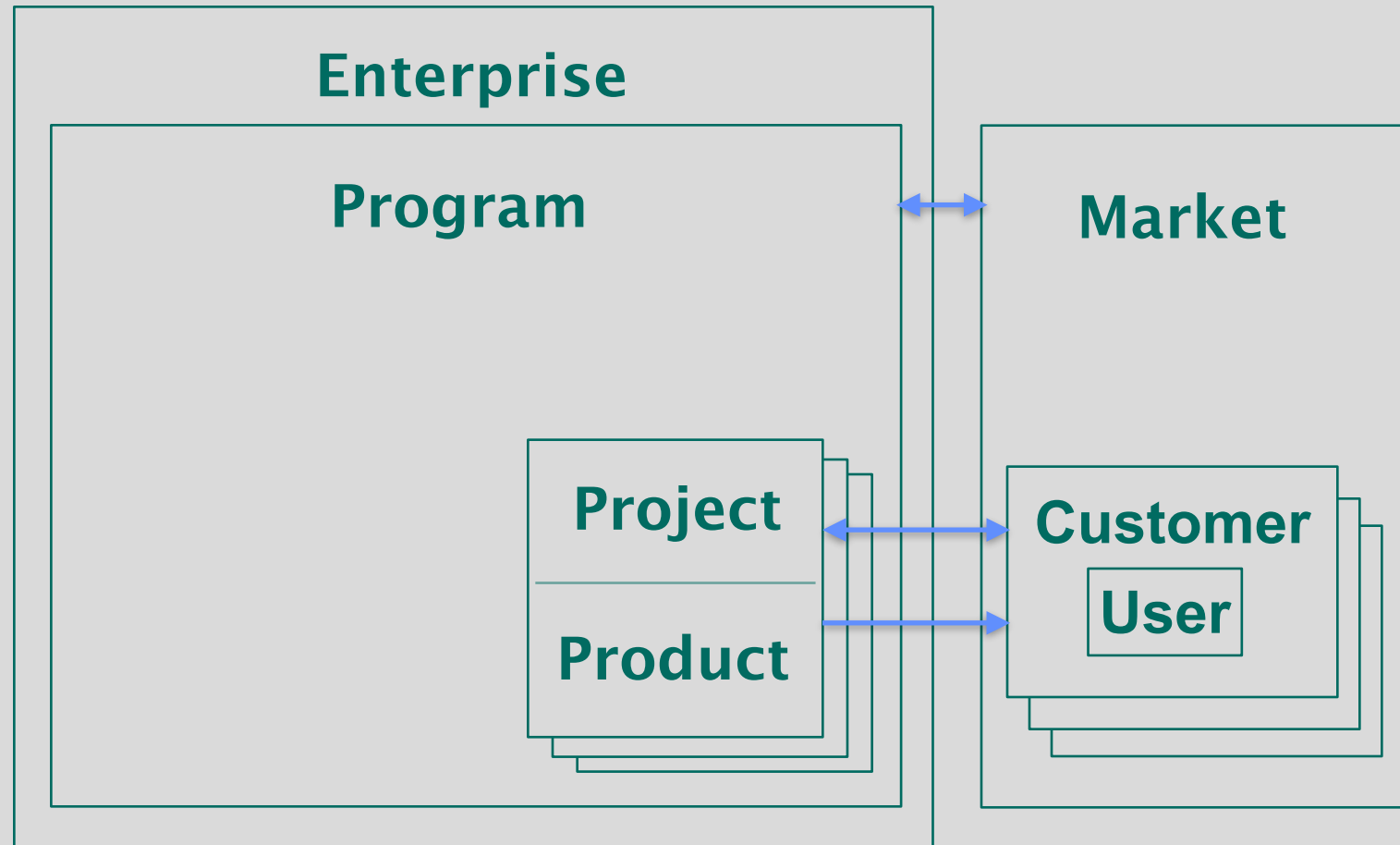
Building a Singular Evolving Product

- **A Simple Market** *{1–n customers, equivalent needs, limited customization}*
- **Disciplined engineering methods and practices**
- **Software Project Model**
 - **Project Management Model**
 - **Product Model** *{facets and elements}*
 - **Competence–scoped Activities**
 - **Concurrent, Method–agnostic Process**
- **Developmental (Process) Quality** *{Feasibility, Sustainability, Conformability, Verifiability}*

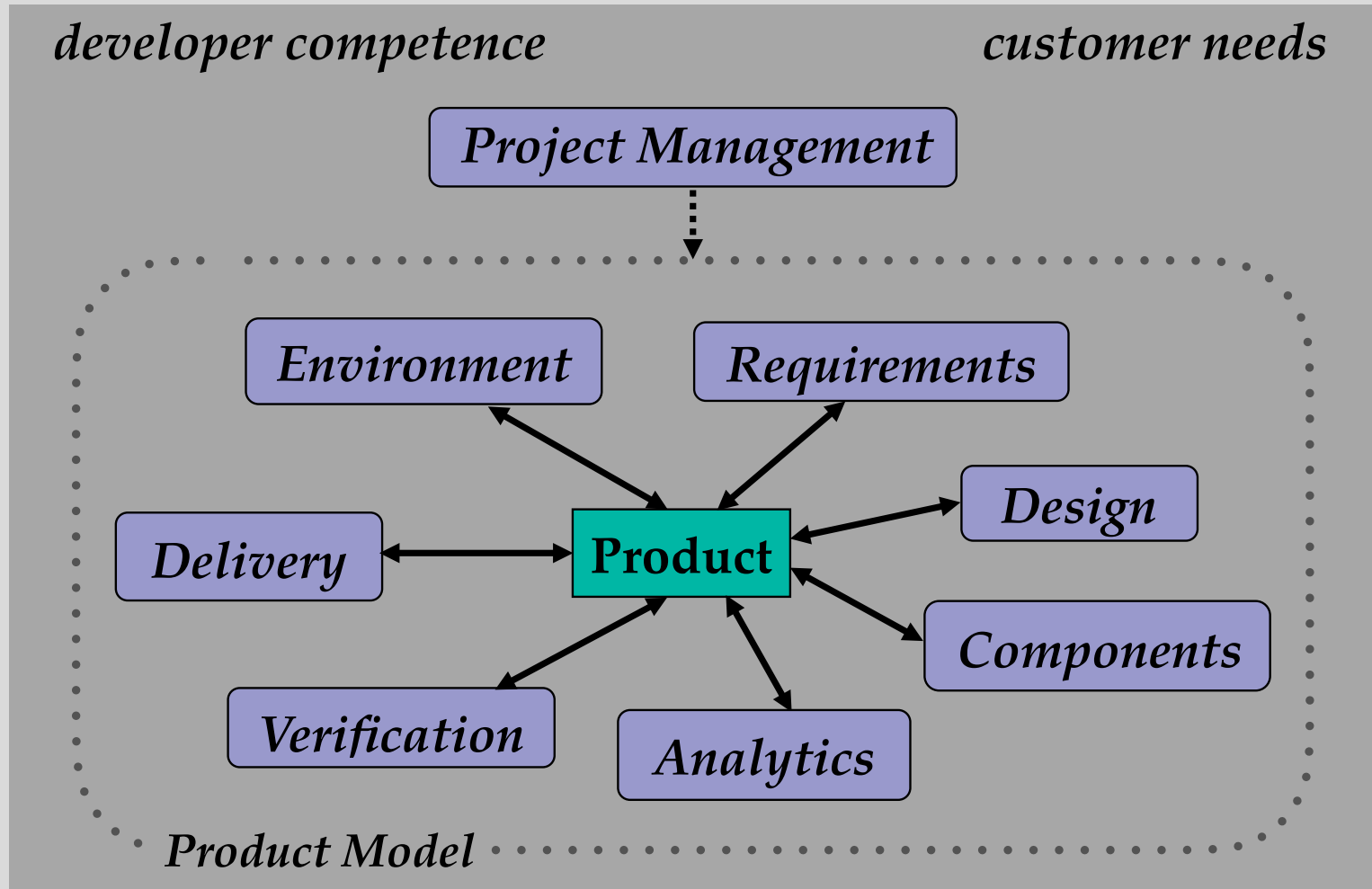
Project Organizational Context

- **Enterprise Governance and Executive Management**
- **Program Management**
- **Project -> Product**
 - **Information = (Data+Metadata) <~ Reality**
 - **Hardware (including Device) Engineering**
 - **Systems Engineering**
 - **Implications for Customers**
 - **Product Acquisition**
 - **Organizational Transition**
 - **Product Evolution**

Conventional Organizational Model



A Software Project Model



Project Management

Managing Project Performance in Coordination with Program Management and Customer

- **Project Direction**
 - Project Performance
 - Process Definition
 - Developmental Quality
- **Product Planning**
 - Customer Relationship
 - Product Master Plan
- **Increment Performance**
 - Increment Plan
 - Version Control
 - Task Performance

Product Delivery

Supporting the Customer in Obtaining and Using the Needed Product

- **Customer Needs (Capabilities & Constraints)**
- **Product Acceptance & Deployment**
 - **Product Configuration and Packaging**
 - **Installation Materials and Assistance**
 - **Support for Validation and Certification**
 - **Support for Customer Transition**
- **Support for Product Use**
 - **Documentation, Training, and Assistance**
 - **Feedback: Deficiencies and Potential Improvements**

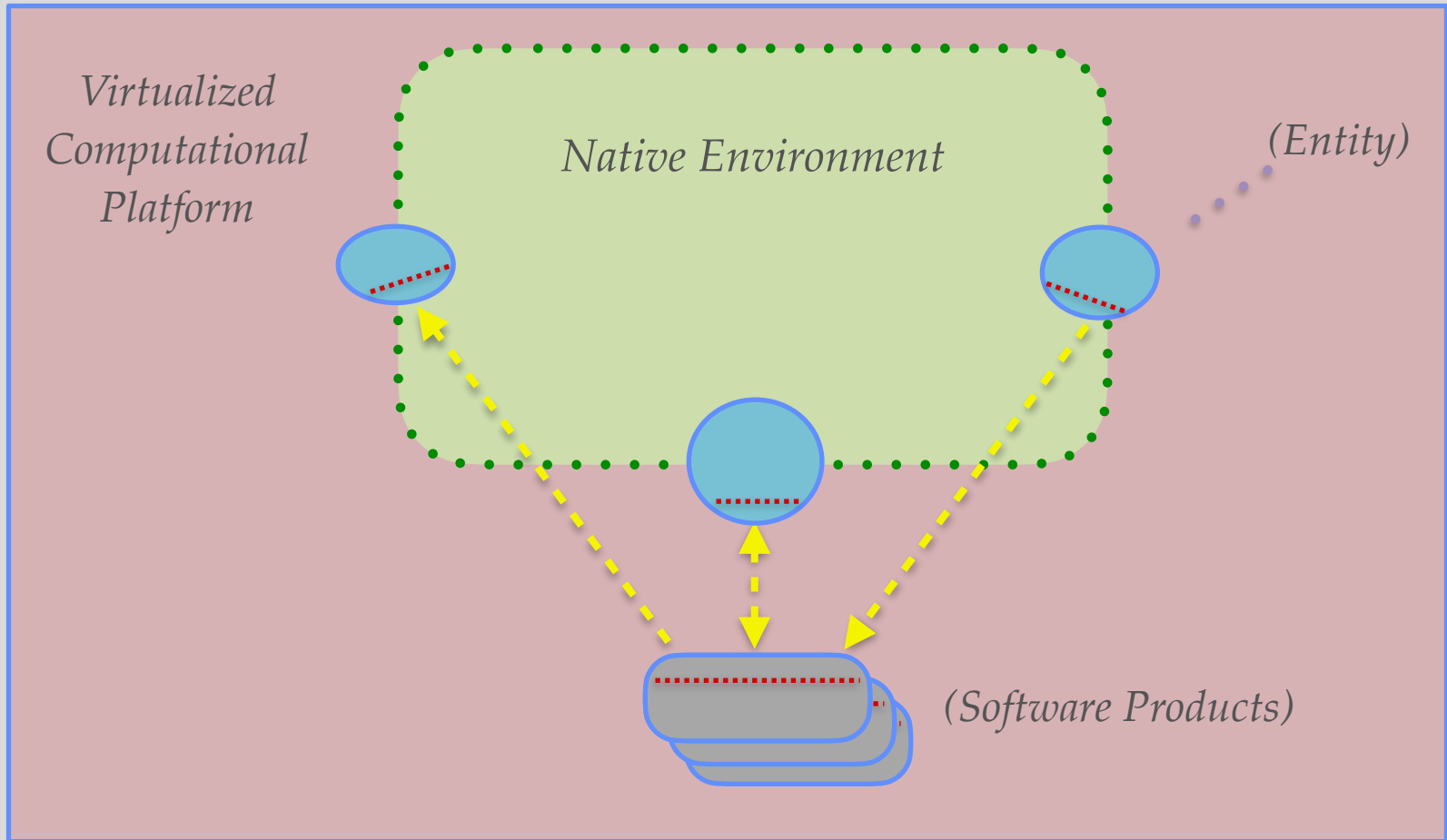
Product Environment

Specifying the Information Content and Behavior of the Product's Operational Environment

- **Ecosystem Content**
 - Native (Physical or Virtualized) Environment
 - Entities (People, Devices, Aggregates/Systems)
 - Information (Data • MetaData)
- **Entity Capabilities**
 - Supply Ecosystem Information
 - Perform Product-requested Actions
- **Intrinsic Behaviors (for Understanding or Virtualization)**
 - Environment
 - Entity

Virtualized and Generalized Product Environments

A Computational Ecosystem



Software prescribes behavior that entities convey

Product Requirements

Specifying the Product's Expected Observable Behavior

- **Concept** *{Realization-independent product capabilities}*
 - **Behavior** *{Coordinate and control entity & composite actions}*
 - **Behavioral (Product) Quality and Tradeoffs**
{functionality, performance, dependability, usability}
 - **Constraints** *{External/Customer/Program Imposed}*
-

Product Evolution *{Guidance on addressing uncertainties, alternatives, and potential changes}*

Product Design

Specifying the Composition of the Product (to-be/as-realized)

- **Product Architecture (3 Views, 6 Structures)**
- **Design Rationale**
 - Analysis of Alternatives
 - Architectural Quality
 - Interface Conventions (User, Device, System)
- **Product Realization**
 - Compositing
 - Platform Virtualization
 - Product Instrumentation

Product Architecture

- **Static View**
 - **Decomposition Hierarchy (= > Components)**
 - **Dependency Relation {Module-level}**
- **Dynamic View**
 - **Concurrency Structure (= > Processes and Threads)**
 - **Coordination-Communications Structure**
- **Physical View (= > Product Realization)**
 - **Composition Mapping** *{incl. applying any build-time customization parameters}*
 - **Deployment Platform Mapping**

Product Components

Specifying each Design-specified Component for Realizing a Product

- **Component Design**
 - Interface Specification
 - Internal Specification
- **Component Implementation**
 - Modules (1–n alternatives)
- **Component Verification**

Product Analytics

Specifying Evidence of Product Quality Using Applicable Analytic Means and Methods

- **Product Quality Analytics**
 - Metrics and Measurements
 - Tradeoff Analyses
 - Formal Methods and Models
- **Product Observability and Introspection**
- **Root Cause Analyses**
- **Comparative Evaluation of Alternative Products**

Product Verification

Specifying Results of Evaluating Product Model elements

- **Consistency and Completeness of Product Model to Customer Needs**
- **Product Behavior Conformance to Product Model**

Perspectives:

- **Directed Reviews**
- **Normative, Anomaly, and Regression Testing**
 - **Testing Platform, Scenario Specifications, Testing Events**
- **Applied Analytics**

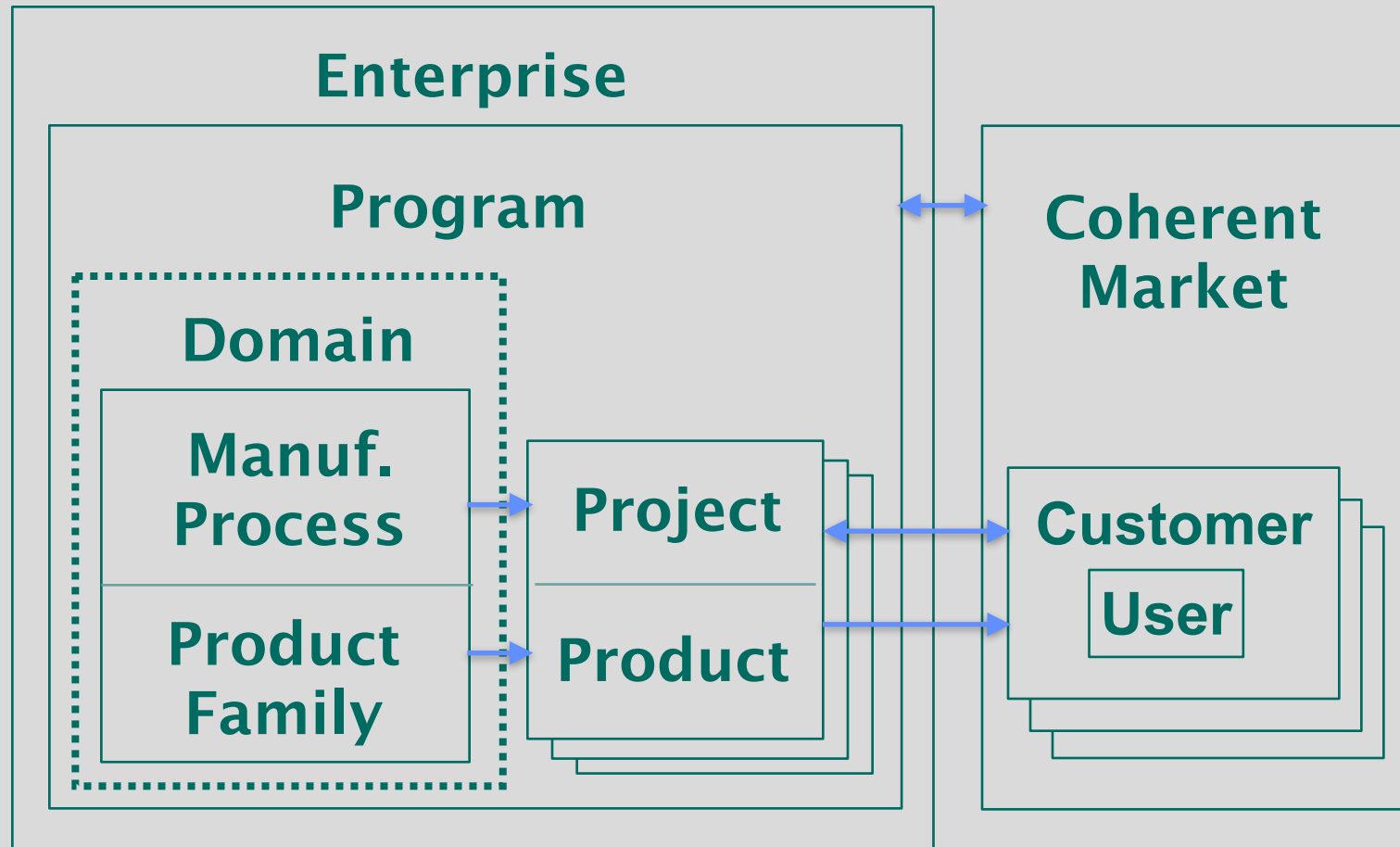
Topic 3

Domain-specific Engineering (DsE) for an Evolving Product Family

Concept of the DsE Methodology

- **Similar problems warrant similar solutions**
- **A Program–Managed Domain**
 - **Coherent Market (Similar Needs)**
 - **Product Family (Similar Products)**
 - **A Concurrent Domain Engineering Process**
 - **A Standardized Product Manufacturing Process**
- **Nature of Manufacturing (Producibility)**
 - **Product Differentiation (Deferred Decisions)**
 - **Streamlined Process (Productivity)**
 - **Comprehensive Product Model**

DsE Organizational Model



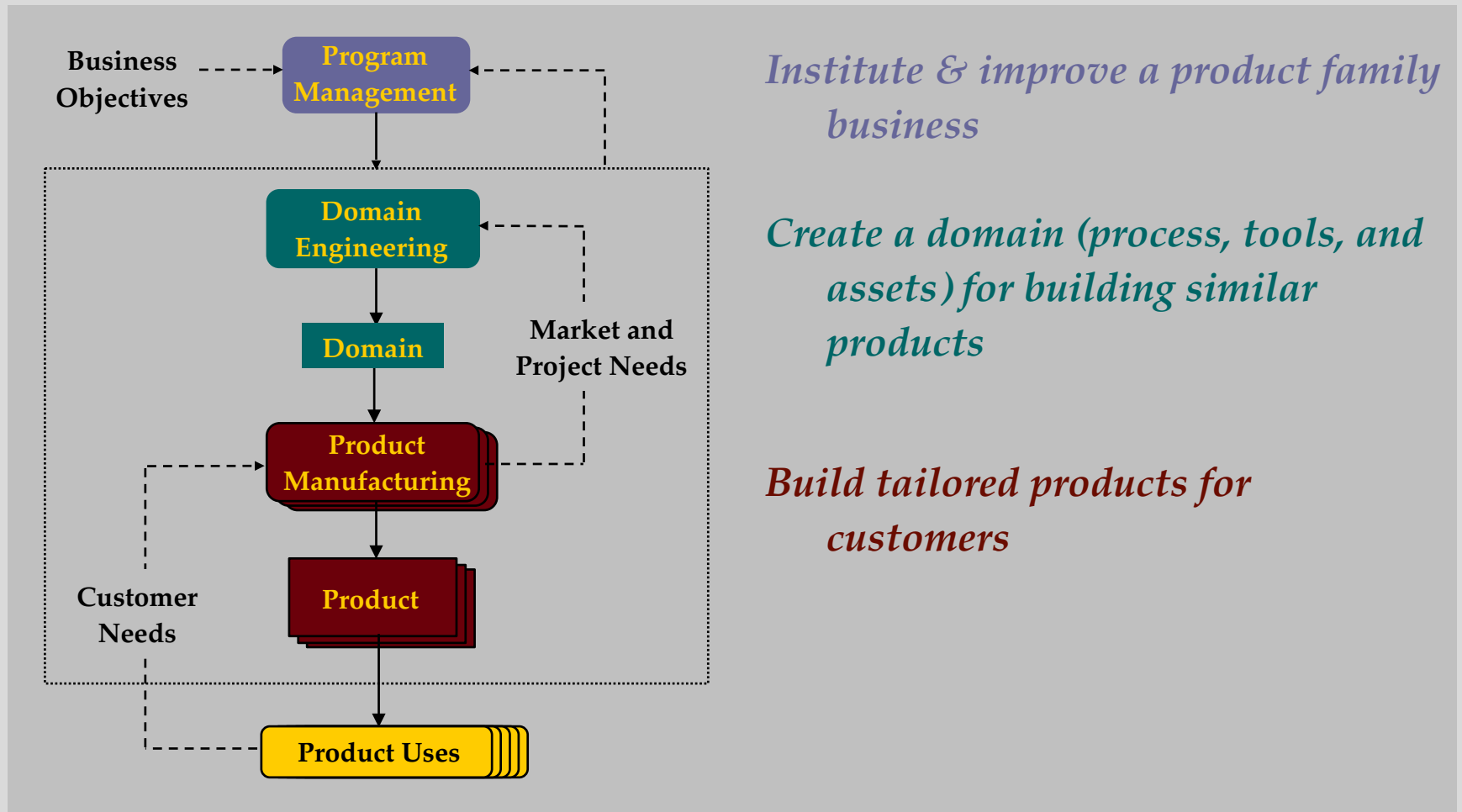
Extended DsE Context

- **Enterprise: Program Charter and Oversight**
- **Diversity of Customer Needs**
- **Added Operational Elements**
 - **Program Management**
 - **Systems Engineering**
 - **Hardware Engineering**
 - **Project–Customer Collaboration**
- **Management & Engineering Discipline**
- **Preferred Singular Product Model & Methods**
- **Developmental & Operational Platforms**

Advantages of DsE Approach

- **Cost-reducing standardization:**
 - Market-focused business objectives define explicit limits on product diversity
 - Development process reduced to its essentials for a particular product family
- **Market-responsive flexibility:**
 - Products tailored to each customer's specific needs
 - Alternate versions of a product built (or modeled) to let a customer compare
 - New product versions rapidly produced whenever needs change

The DsE Tripartite Process



3.1: DsE Program Management

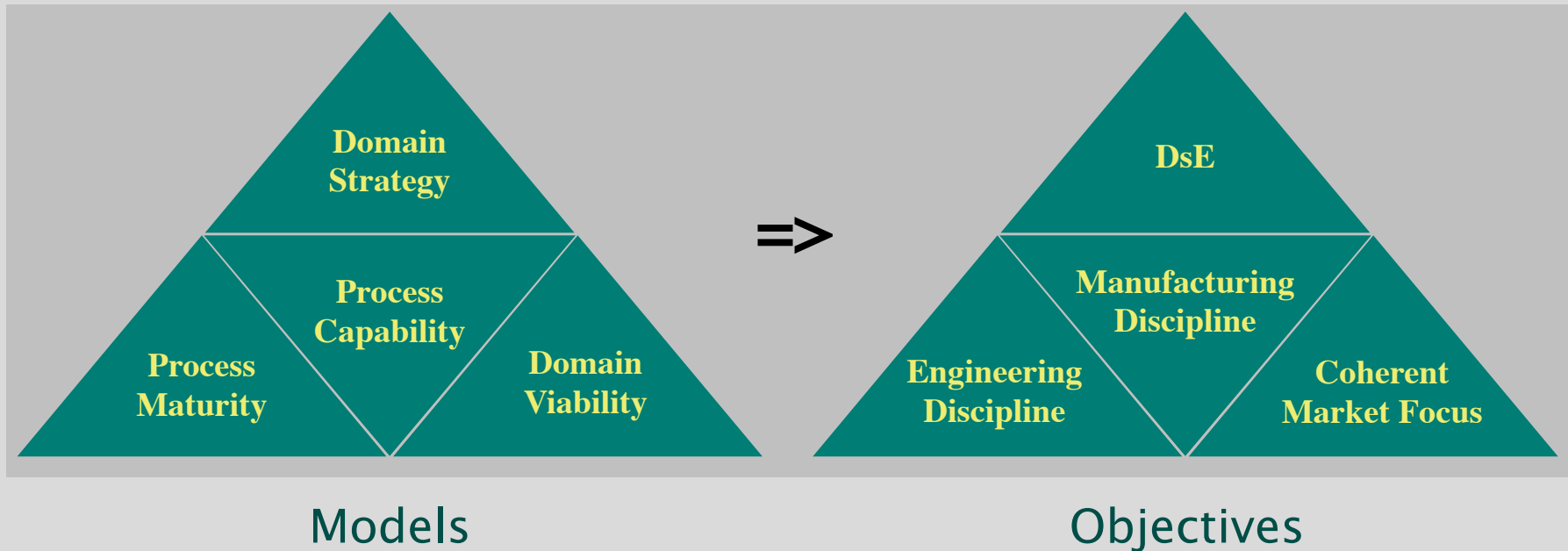
Instituting and Improving a Product Family Business

- **Program Governance**
 - **Domain Viability (Market Opportunity, Technical Competence, Business Commitment)**
 - **Process Capability [Metrics & Tradeoffs]
=> Program Organizational Structure**
- **Program Direction**
 - **Process Maturity**
 - **Domain Strategy**
- **Program Performance (Marketing, Domain Engineering, Product Manufacturing Projects)**

Program Management Objectives

4 Models for Formulating Appropriate Objectives

(Process Improvement for Software Product Lines [DsE PIr Tutorial])

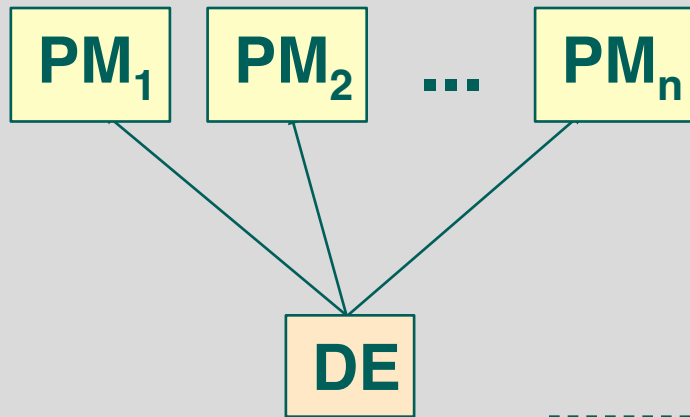


Economic Profile for a DsE Program

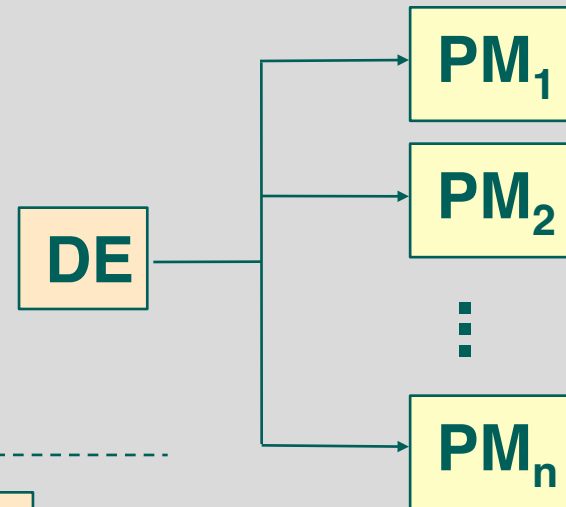
- **Predictability with Reduced Effort, Uniformity for Increased Product Quality**
 - **Capital Asset: Problem–Solution Competence**
 - **Domain Engineering**
 - ▶ **Eliminates Non–Essential Product Differences**
 - ▶ **Streamlines/Subsumes Recurring Product Manufacturing Effort**
- **Tradeoffs:**
 - **Domain Focus on Market vs Manufacturing Project Focus on Customer**
 - **Product Family Breadth vs Domain Complexity**
 - **Domain Investment vs Technical Debt**

Program Governance: Alternative Organizational Structures

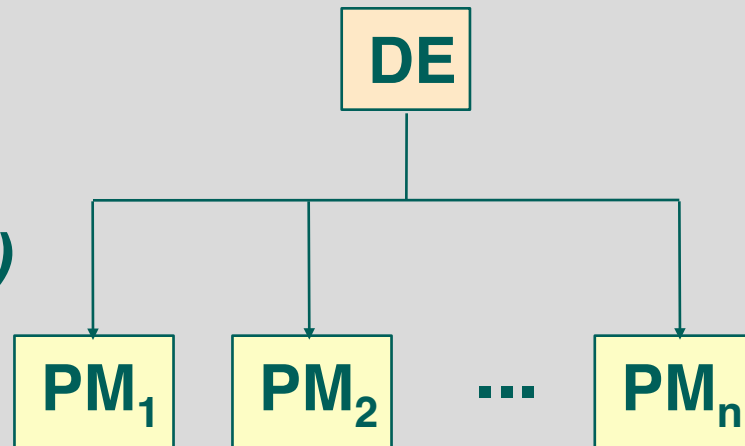
Independent



Collaborative



Coevolving (Domain-Market)



BREAK
(10:30–11:00)

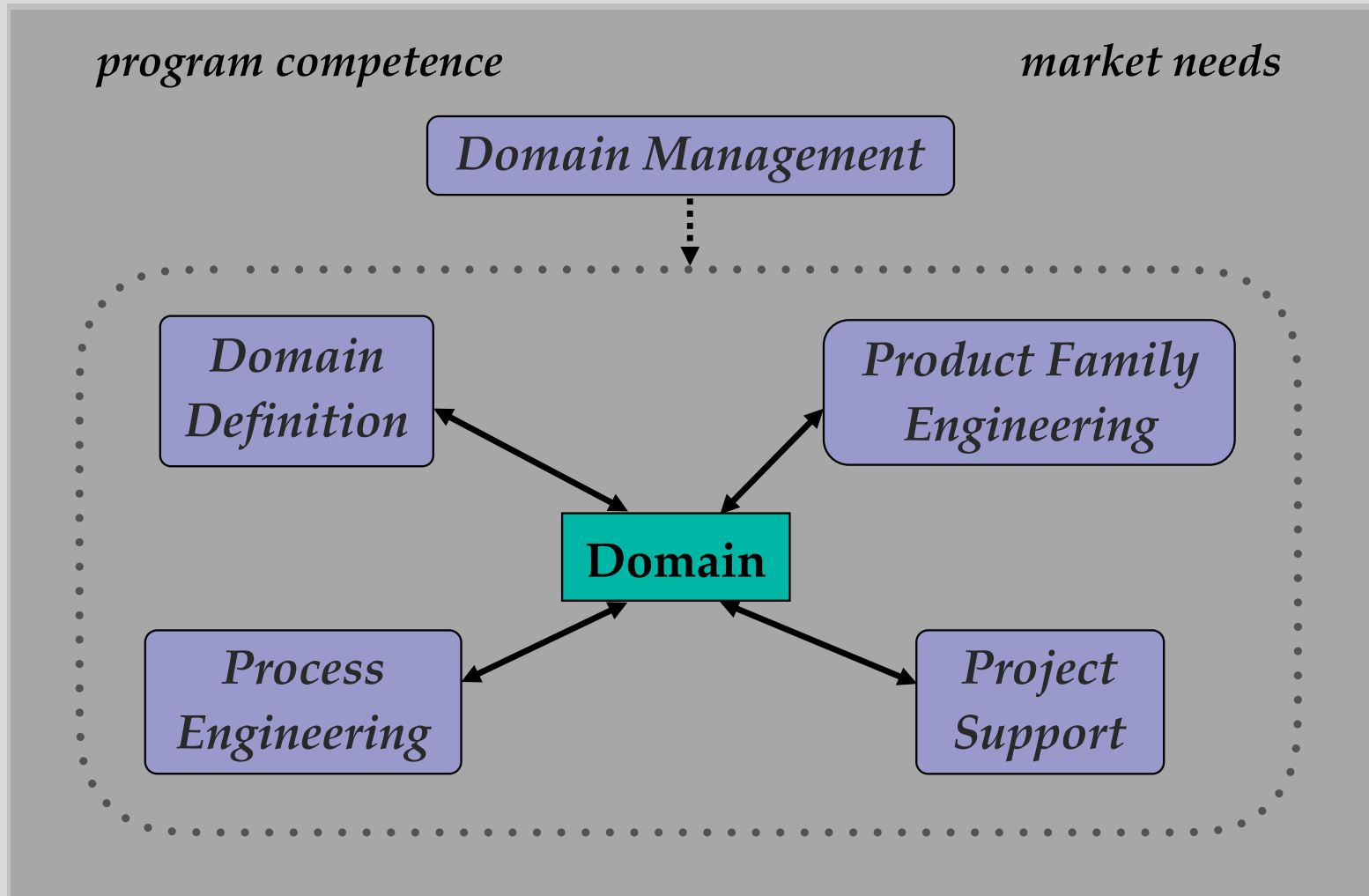
3.2: DsE Domain Engineering

{Technical Agent for Program Management}

Creating a Domain for Building Similar Products

- **Abstraction -> Assumptions -> Product Family**
- **Variability Assumptions -> Deferred Decisions
=> Product Specification**
- **Product Manufacturing Process**
{for affiliated Manufacturing Projects}
[product family • product specification]
=> product model
-> product

DsE Domain Engineering Model



Domain Management

Planning and Conducting the Domain Engineering Process

- **Domain Direction**

- Domain Strategy {per Program Management}
- Domain Process + Methods + Process Quality Criteria

- **Domain Planning**

- Program–Market <– Project–Customer Needs
- Domain Master Plan {over Program Life}

- **Increment Performance**

- Process Iteration to Baselines/Delivery
- Coordination with Dependent Projects

Domain Definition

Establishing a Unified Domain Vision

- **Domain Synopsis**
- **Domain Terminology** *{and Technical References}*
- **Legacy Assets** *{e.g., existing/prior singular products}*
- **Assumptions of Commonality & Variability**
{hierarchical text; formalize in a Feature Model?}
- **Decision Model** *{Assumptions-based Deferred Decisions}*

The Role of Deferred Decisions

- Development is a decision-making process
- Decisions represent:
 - Customer-specific needs and constraints
 - Engineering tradeoffs to achieve product quality
- A focus on similar problems {a family} => commonalities-based standardization *{reduces number, variety, and complexity of deferred decisions}*
- A product family shows how different ways of resolving a set of decisions lead to different products *{every decision partitions a family into subfamilies; fully resolved decisions -> an equivalence/unary subfamily}*
- “Decision model” condenses customer-developer dialog (product specification) to its essentials

Product Family Engineering

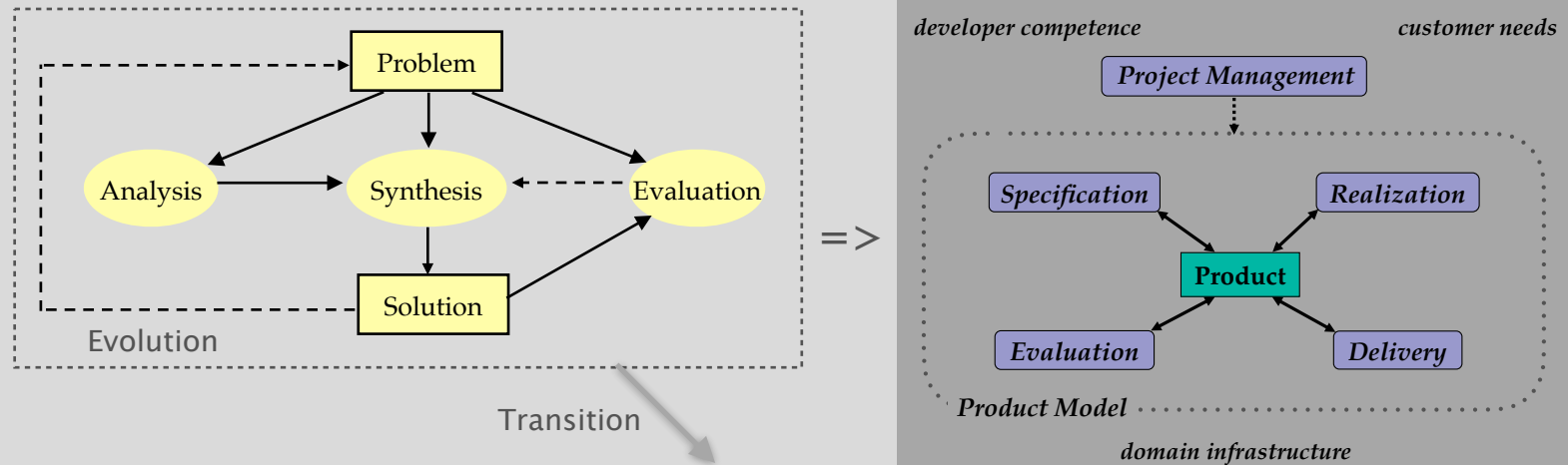
Formulating an Envisioned Set of Similar Products

- **Representing a Product Family**
 - (product model • decision model)
 - > generalized product model
- **Instantiating a Product Family**
 - (product family • product specification)
 - > instance product model
- **Verifying a Product Family**
 - Directed Reviews, Selective Instance Testing, Family-level Analytics
- **Incrementally Evolving a Product Family**
 - Evolving Market, Unmet & Evolving Customer Needs

Process Engineering

Prescribing How Manufacturing Projects Will Work

- **Product Manufacturing Process**
 - **Decision Model –> Product Specification dialog**



- **Product Manufacturing Capability**
 - **Developer Guidance**
 - **Automated Process Support**

Project Support

Coordinated Assistance for Associated Manufacturing Projects

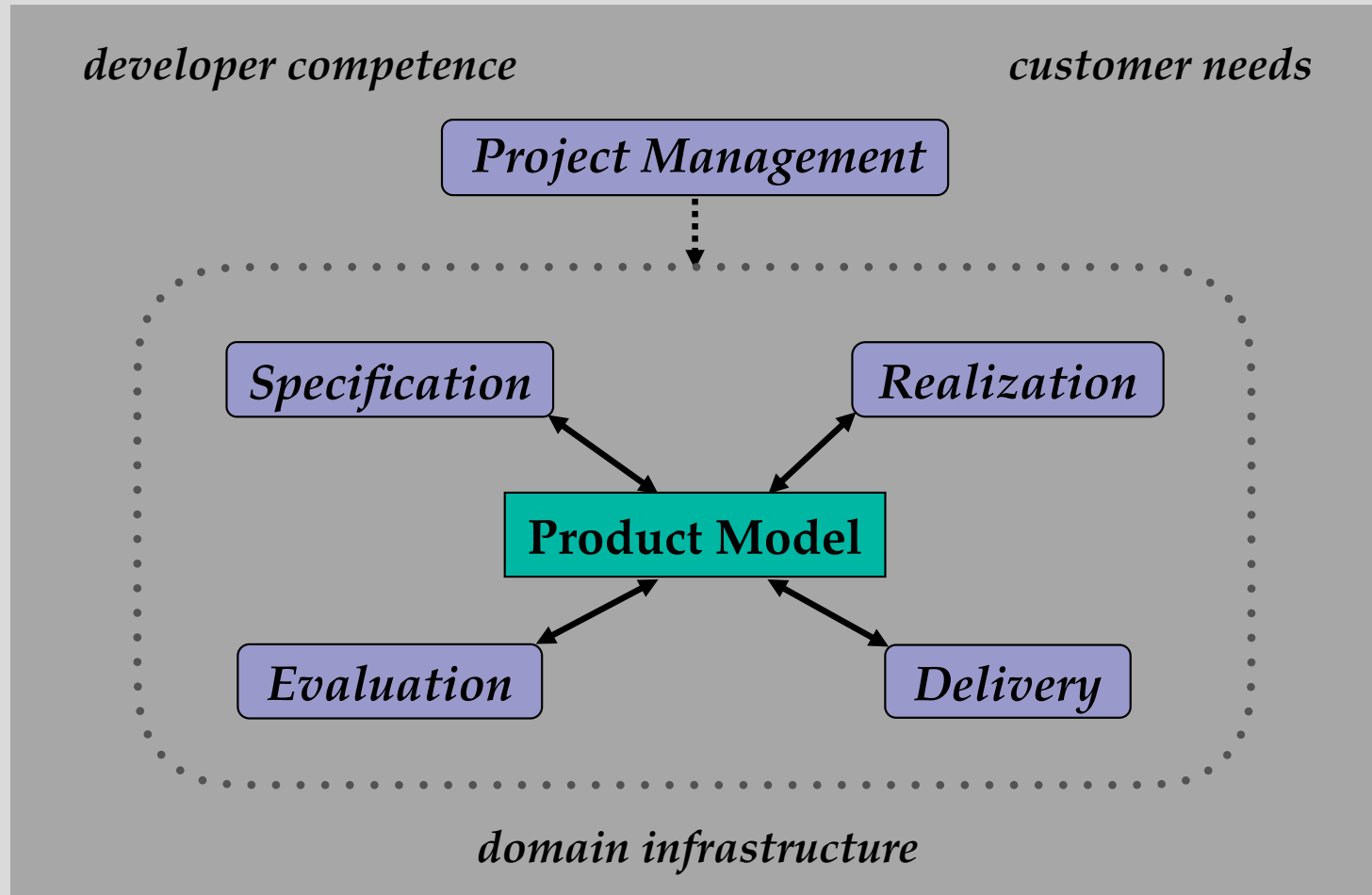
- **Project Needs**
 - Needed Manufacturing Capabilities
 - Mitigating Domain Limitations
- **Domain Evaluation**
 - Domain Capabilities Fit to Project Needs
 - Achievable Project Productivity and Quality
- **Domain Delivery**
 - Assisting Projects in Effective Domain Use
 - Identifying Needed Improvements

3.3: DsE Product Manufacturing

Building Singular Products for Customers

- **A Single Customer or Simple Market**
- **Supported by Domain Infrastructure**
 - **Domain-specified Manufacturing Process**
 - **Focused on resolving deferred decisions**
 - **Enabled by domain-provided automation**
 - **Product Family**
 - **Instantiated with resolved decisions**

A Normative Manufacturing Process



product family (product specification) -> product model

Project Management

Planning and Directing Effort to Build a Product

- **Project Direction**
 - Organization, resources, and development guidance for the project
- **Product Planning**
 - Customer relationship and master plan for product development
- **Increment Performance**
 - Results of increment tasking to produce product candidate baselines for evaluation and delivery

Product Specification

Specifying a Candidate Product as the Resolution of Deferred Decisions based on customer needed capabilities

Deferred Decisions –> Product Specification
{a domain-specific language}

- **Resolved decisions: perceived customer needs**
- **Unresolved decisions: uncertainties**
- **Tradeoffs: alternative decision resolutions**

Product Realization

Deriving a Total Product for Evaluation & Use

- **(Product Family • Product Specification)**
=> **Product Model**
 - **Singular Product**
 - **Project internal documentation**
 - **Profiled Quality Criteria**
 - **Virtualized Operational Environment (Native Environment + Entities) for Evaluation**

Product Evaluation

Evaluating Product Specification and Realization to Needs

- **Validation of Product Specification**
 - Consistency
 - Completeness
 - Domain Limitations (*unsupported capabilities*)
- **Verification of Realized Product Behavior**
 - Behavior correlates to Product Specification
 - Reviews & Scenario-based Testing of Behavior
 - Static & Dynamic Analyses of Product Behavior
- **Product Family & Process Discrepancies**
- **Comparative Analyses of Alternative Product Specifications**

Product Delivery

Supporting Customer in Obtaining and Using the Product

- **Customer Needs (Capabilities & Constraints)**
- **Product Acceptance & Deployment**
 - **Installation Support**
 - **Validation and Certification Support**
 - **Organizational Transition Support**
- **Support for Product Use**
 - **Documentation, Training, and Assistance**
 - **Feedback: Deficiencies, Potential Improvements**

Topic 4

Automation and Future Technologies

Reuse Capabilities

Anticipating Technologic Change

4.1: Reuse Capabilities

Software Component Reuse

- **Opportunistic: Existing/Repository-based Parts, Used As-Is or Modified**
- **Systematic: Parts Intentionally Built for Envisioned Multiple Uses and Change**

Abstraction-based Environments

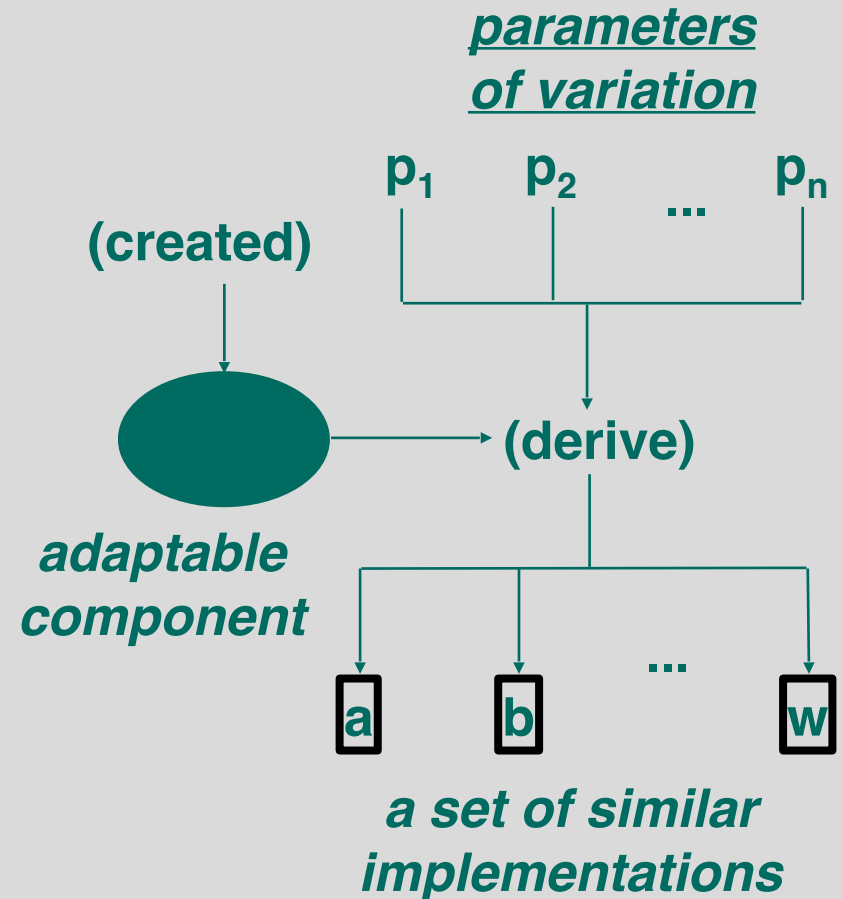
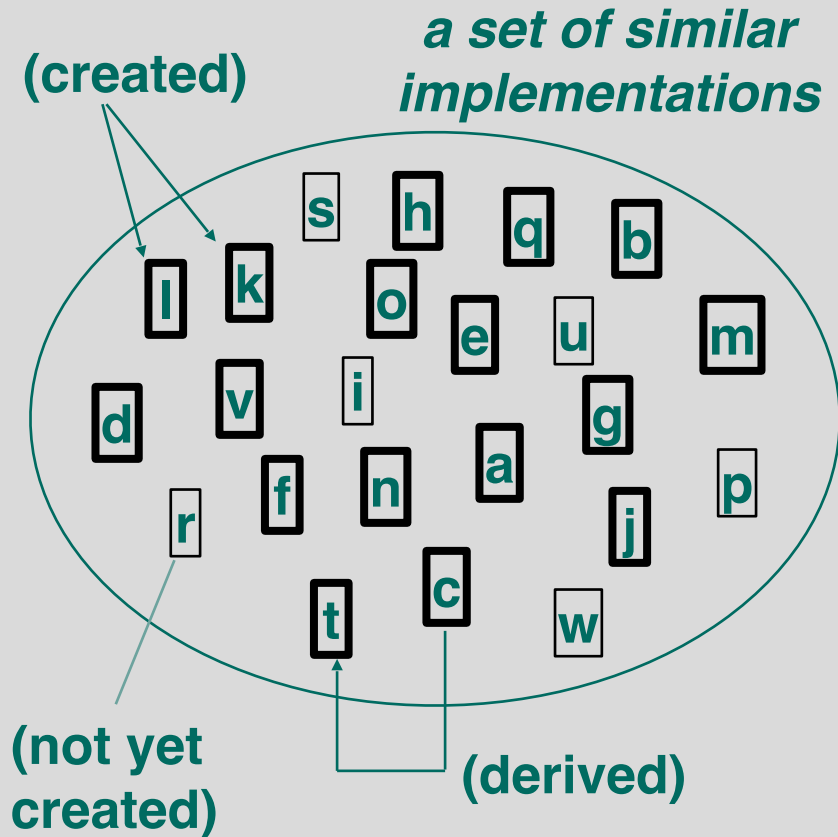
Domain-specific Environments

Basic Tenets of Systematic Reuse

- The only sound basis for reuse is an envisioned set of similar products (or parts)
- Similarity implies commonality and variability
 - Commonality is the basis for standardization of products and process (to form a domain).
 - Variability is the flexibility for customization to differences in customers' needs.
- Adaptability is an explicit representation of similarity, characterized by a set of deferred/changeable decisions sufficient to designate a particular instance of a set.

Two Views of a Component

Component => 1-n Alternative Implementations



(DsE Adaptable Components Tutorial)

Generative AI-Enabled Reuse

LLM-based statistical analysis of similarity in legacy software content

- **for Opportunistic reuse:**
 - Enter a free-form prompt to generate reusable content (as-is or modify?)
 - Alternative versions: modify the prompt
- **for Systematic reuse:**
 - Enter a free-form prompt to generate and verify adaptable (parameterized) content
 - Apply parameters to adaptable content to derive uniformly customized content
 - Alternative versions: modify parameter values

Abstraction-based Environments

- **Characteristics**

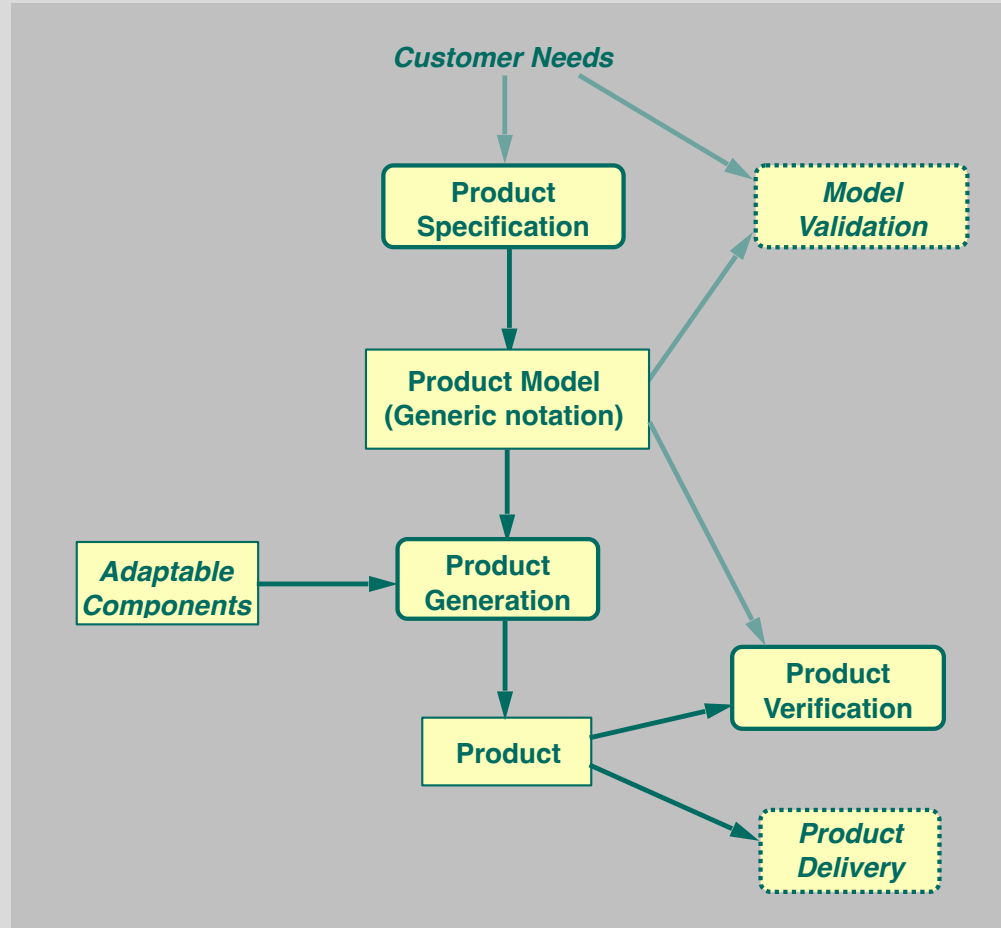
- **Product specification using a semi-formal requirements notation to define behavior**
- **Encapsulated component-based software product design**
- **Associated set of computational abstractions ($\Rightarrow$ adaptable components)**
- **Fully automated product generation with instantiated components**

- **Spectrum Environment 1986 (Formative/Proof of Concept)**

- **Design and computational abstractions shared with generated applications**

Spectrum (1984–8, Precursor to Synthesis/DsE)

- **Information Processing Applications**
- **Specify World Model**
 - Data Model
 - Reasoning
- **Specify Interfaces**
 - Users
 - Relational Database
 - Platform Devices



Domain-specific Environments

Comprehensive DsE Capabilities

- Program Management
 - Domain Engineering
 - Product Manufacturing
-

Potential for a Family of DsE Environments

- Ability to Derive a Customized DsE Environment
{A Software-based Product}
- Generic base of Management & Technical
Methods, Practices, and Representations
- Specialization based on Domain Diversity *{e.g.,
Terminology, Product Family Content}*

4.2: Anticipating Technologic Change

- **Developmental/Operational Practices**
- **Predictive Analytics**
- **Computational Technologies**
- **Data Science, Artificial Intelligence, and Machine Learning (Generative AI)**

Developmental/Operational Practices

- **Developmental Capabilities**
 - **Methods (Engineering Discipline)**
 - **Techniques and Tools (Problem–Solution Analyses)**
 - **Enhanced Pre–Deployment Evaluation**
 - **Supporting Alternative Versions and Variants**
- **Operational Capabilities**
 - **Hardware Faults and Failures Mitigation**
 - **Temporal Behavior Flexibilities**
 - **Information/Communications Overload**
 - **Data Efficiency, Integrity, Flexibility**

Predictive Analytics, Motivations

- Improvements in Analytic Methods based on Predicted vs Observed Quality
- Selective Focus on Components that Determine Key Quality Factors
- Sensitivity among Interrelated Quality Factors
- Predictive Change-Directed Root Cause Analyses (*Prognostics*)
- Comparative Quality Analyses of Alternative Products
- Ecosystem and Platform Behaviors that Limit Quality Factors
- Product Family-level Analytics *{generalize formal methods, reviews, and test scenarios to obtain a quality profile for a set of similar products}*

Computational Technologies

- **Specialized Processors** *{e.g., sensors/actuators, specialized/optimized computations}*
- **Distributed, Mobile, & Autonomous Computing**
 - Cyber-Physical Computing
 - Augmented/Spatial Computing
- **Manufacturing Technology**
 - Hardware Fabrication and Assembly
 - Additive and Robotic Techniques
 - Mass Customization *{=> Product Family}*
 - Factory Virtualization
- **Quantum Computing**
- **Biologic Computing** *{e.g., DNA data storage, organoids}*

Generative AI: Relevance for DsE

{Examples of options in 4 categories}

Profiling Product Behavior

- **Generate a user view of a product's behavior**

Generating Product Content

- **Derive a component implementation (with a standardized decision-based prompt)**

Diagnosing Product Quality

- **Compare quality criteria satisfaction for alternative component implementations**

Leveraging Similarity

- **Profile how changing a prompt will affect product behavior**

Generative AI: Potential Concerns

- Enormous expense in resources for building and operating remote LLM data centers and for responding to prompts
 - Reliance on mixed-quality source material {*defective, missing, restricted, non-relevant, outdated*} => *Authoritative?*
 - Insufficient deep or specialized competence for consistent, reliable results (for complete product model content)
 - Inability to recognize insufficient contextual or tacit information to produce a valid response
 - Superficial “reasoning” to answer, explain, or justify results {*“The Illusion of Thinking...”* <https://arxiv.org/abs/2506.06941>}
 - Uncertain effects of change: probabilistic derivation of differing (even if equivalent) or overly uniform results
 - Progressive shallowing of developer competence
-
- How to determine utility and proper use of generative AI in products

Distill Domain-specific LLMs?

Objective: Derive product content alternatives based on substantially domain-related source material

- **Isolation of domain-aligned product content for generating customized instance products**
- **A simpler standardized prompt, based on identified customer-relevant deferred decisions**
- **Comparative evaluations of degree to which generated instances fit specified behavioral quality criteria**

Rationale: Faster derivation of better quality customized results

{see <https://developers.google.com/machine-learning/crash-course/llm/tuning>; <https://doi.org/10.1049/sfw2/5691460> section 4.1; and slide on “Generative AI-Enabled Reuse”}