Prosperity Heights Software

SPLC 2025
Tutorial

2 September 2025

The Engineering and Manufacture of Software-based Products

Grady H. Campbell, Jr.

gradycampbell@domain-specific.com

4 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

<domain-specific.com/EMSP/Tutorial.pdf>

Book for Reference

Draft Content: <domain-specific.com/EMSP>

Preface

1. **The Foundations and Future of Software Production**
2. **Basic Software Product Engineering**
3. **Domain-specific Engineering**
4. **Automating Software Development**
5. **Anticipating Change**

Appendix: Mathematical Formulation of Product Family Concept

Objective

- An Enhanced Methodology for Developing Software-based Products
  - Process Innovation for Producibility
  - Standardization based on Product Similarity
  - Flexibility through Constrained Customization
- Topical Elements
  - Principles of a Product Family Methodology
    {First articulated in my conception of Synthesis/ Reuse-driven Software Processes 1990, then DsE 1996}
  - Information Content for a Product and Product Family
  - Potential Influences on the Software Enterprise

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

=> **Rapidly build high-quality (software-based) products**

Basic Precepts:

- **A problem must be properly understood to be properly solved.**
- **Every well-defined problem has many potential solutions.**
- **Every constructible product is similar, in varying degrees and aspects, to other products, past, present, and future.**

Perspectives on Producibility

A product is conceived to provide capabilities that enhance a customer's ability to achieve their objectives

- **The Engineering–Manufacturing Combine**
  - **Engineering: Provide resources by which product can be efficiently produced**
  - **Manufacturing: Apply engineering-provided resources to build a needed product**
- **Factors enabling producibility goals**
  - **Developer Productivity** *(efficiency & effectiveness)*
  - **Product Value** *(utility & quality)*
  - **Customer Acuity** *(insight & foresight)*

Foundations

- Basic Terminology
- Organizational Concepts
- Approximating an idealized “Design” (Product Development) Process
- Process Alternatives
- Operational Concepts
- Key Foundational Constructs
- Methodological Themes

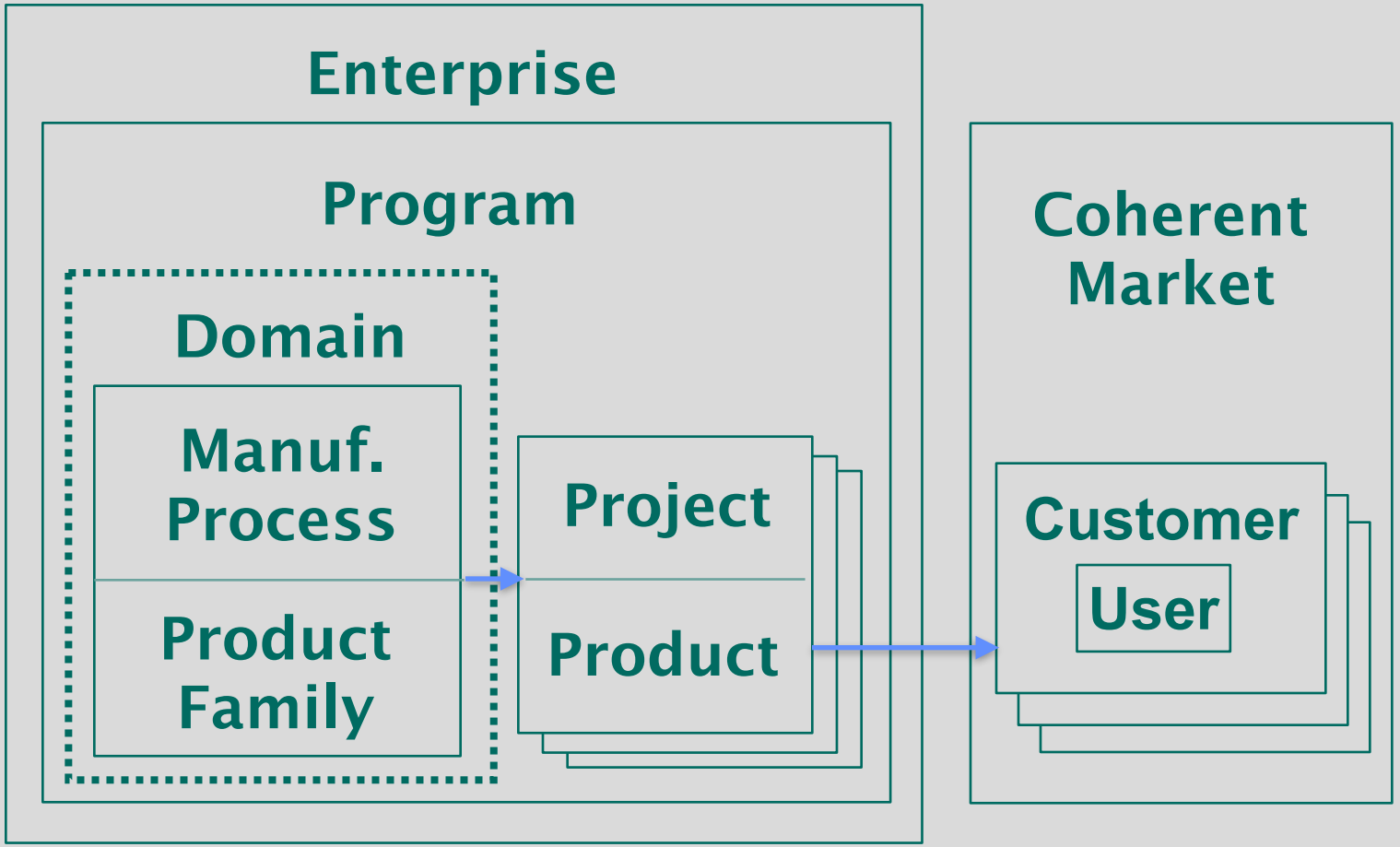
Basic Terminology

- **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**
- **Product Family – An envisioned set of similar products**
- **Product Line – Related products or realized instances of a product family**
- **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]**

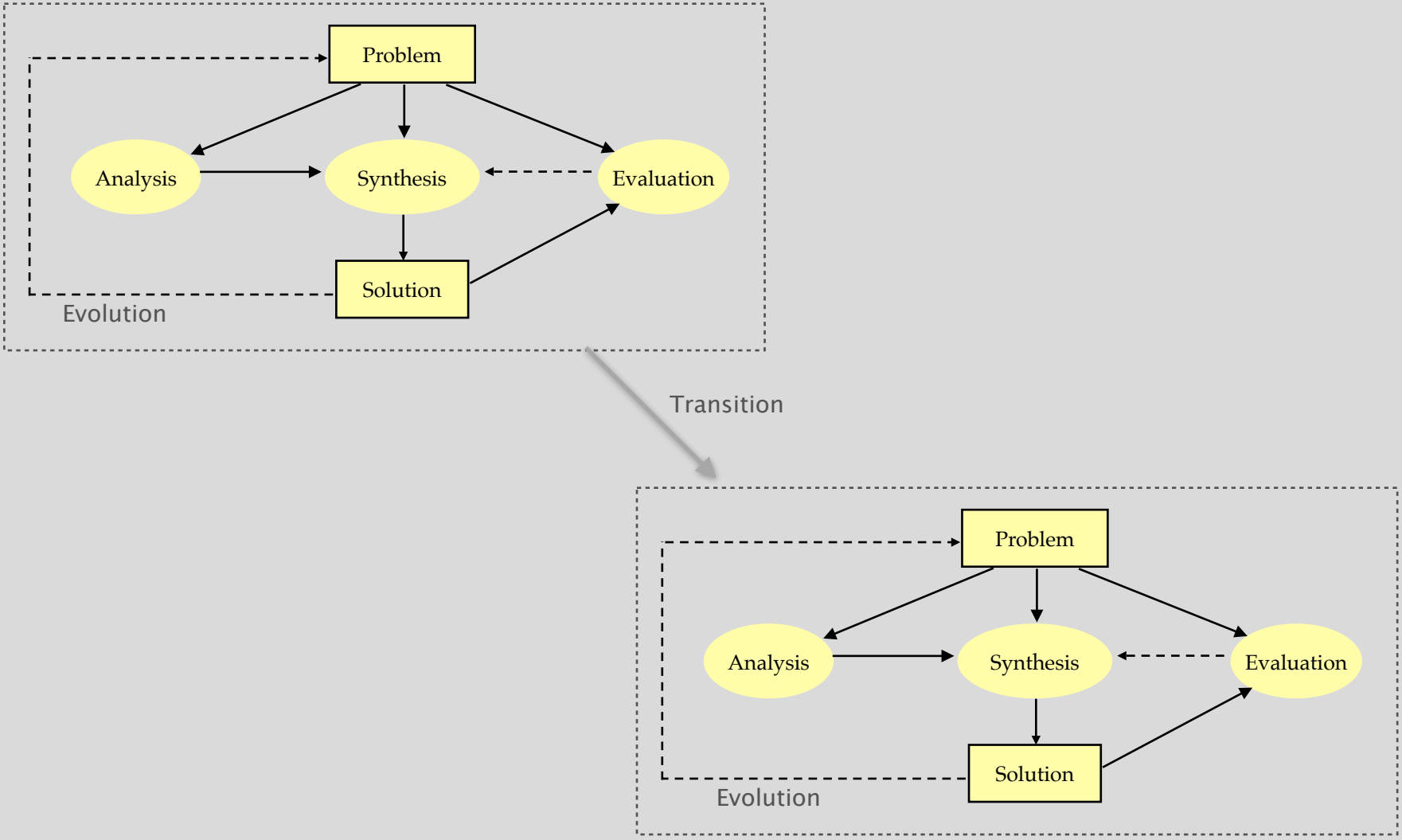
A Mathematical Formulation of a Product Family (EMSP Appendix)

- 4 Views of a Program
  - Representational: Source vs Object
  - Behavioral: Functional vs Computational
- Distinguishing Among Programs
  - Equality, Equivalence, and Similarity
- Metric Space for Representing a Program Family
  - Distance Metric based on Similarity
    {Distance implies differences—changes required to transform one instance into another—perhaps in terms of how deferred decision resolutions differ}
- Extended Metric Space for a Product Family

Organizational Concepts



An Idealized Product Realization Process



Alternative Product Development Processes

- Iterative "Waterfall"
  - Well-Understood Problem
  - Solution Competence
- Agile
  - Incompletely Understood Problem
  - Evolving Solution
- Concurrent
  - Uncertain/Evolving Problem
  - Alternative Solutions

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:

- **Product Model Element Dependencies**
- **Availability of Competence** *{Knowledge+Experience+Expertise}*
- **Maintaining Model Consistency**

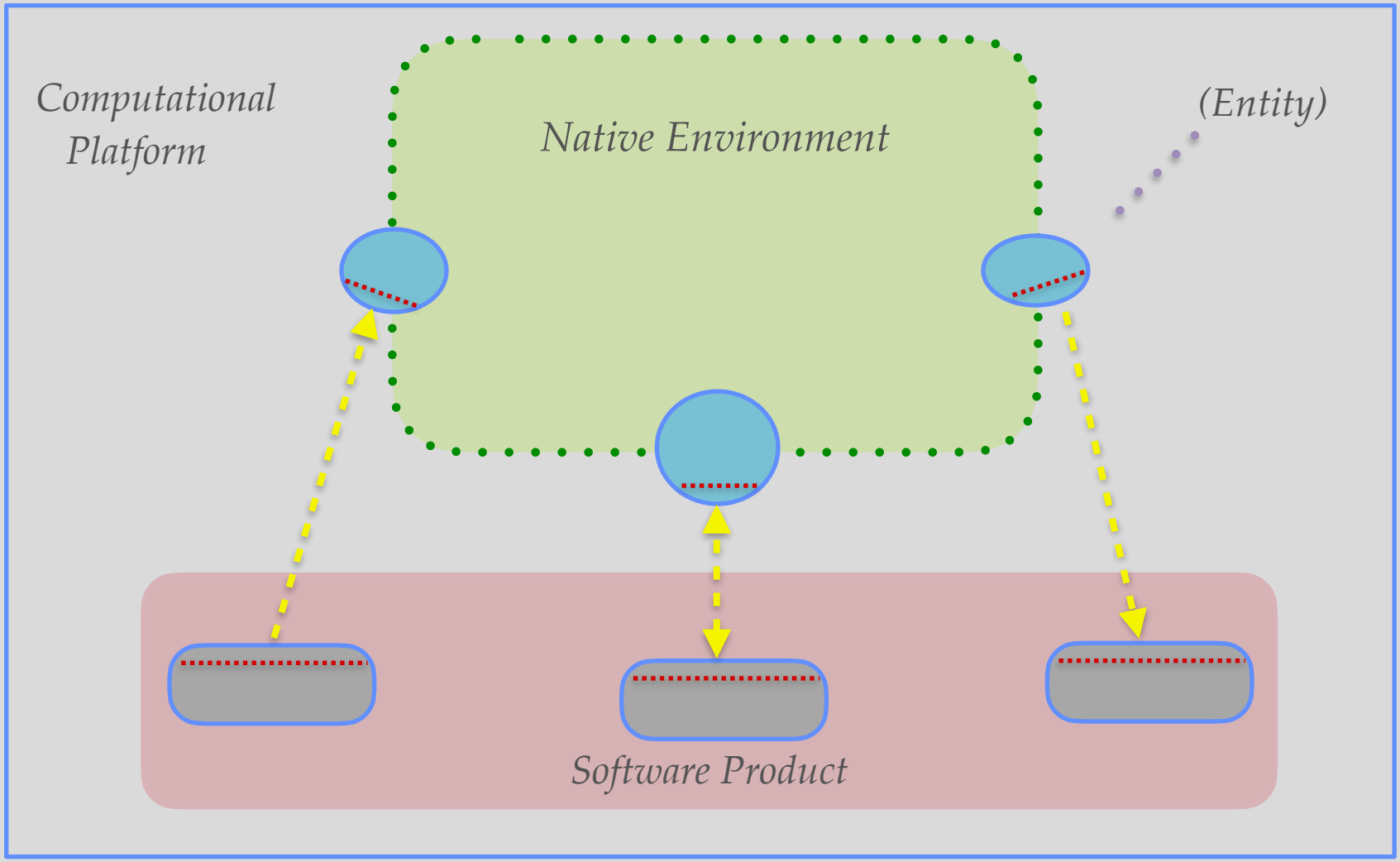
Other Considerations:

- **Specifications <-> Realizations**
- **Management need to track progress (Portray ("fake") a rational process)**

Operational Concepts

- Operational Environment (Natural/Artificial)
- Ecosystem: Environment + Entities
- Entities: People/Roles, Devices, Aggregates/Systems
- Information -> (Data, MetaData)
- Hardware, Software, Platforms
- Devices (physical or virtual)
  - Computational, Edge, Interface, Storage
  - Aggregate
- Software Platform: Virtualized Computational Device

A Computational Ecosystem



Key Foundational Constructs

- Software Development as an Engineering Discipline
  - Separation of Concerns Principle
  - Competence-based Tasking
- Process and Product Quality Criteria
  - Quality Perspectives on Process Variation (Capability, Maturity, Performance)
- Directed Reviews (Author-Initiated, Issue-Focused, Selected Reviewers, Relevant Competence)
- Anticipating Product Changes
  - Problem-Solution Uncertainty
  - Solution Deficiencies
  - Problem-Solution Changes

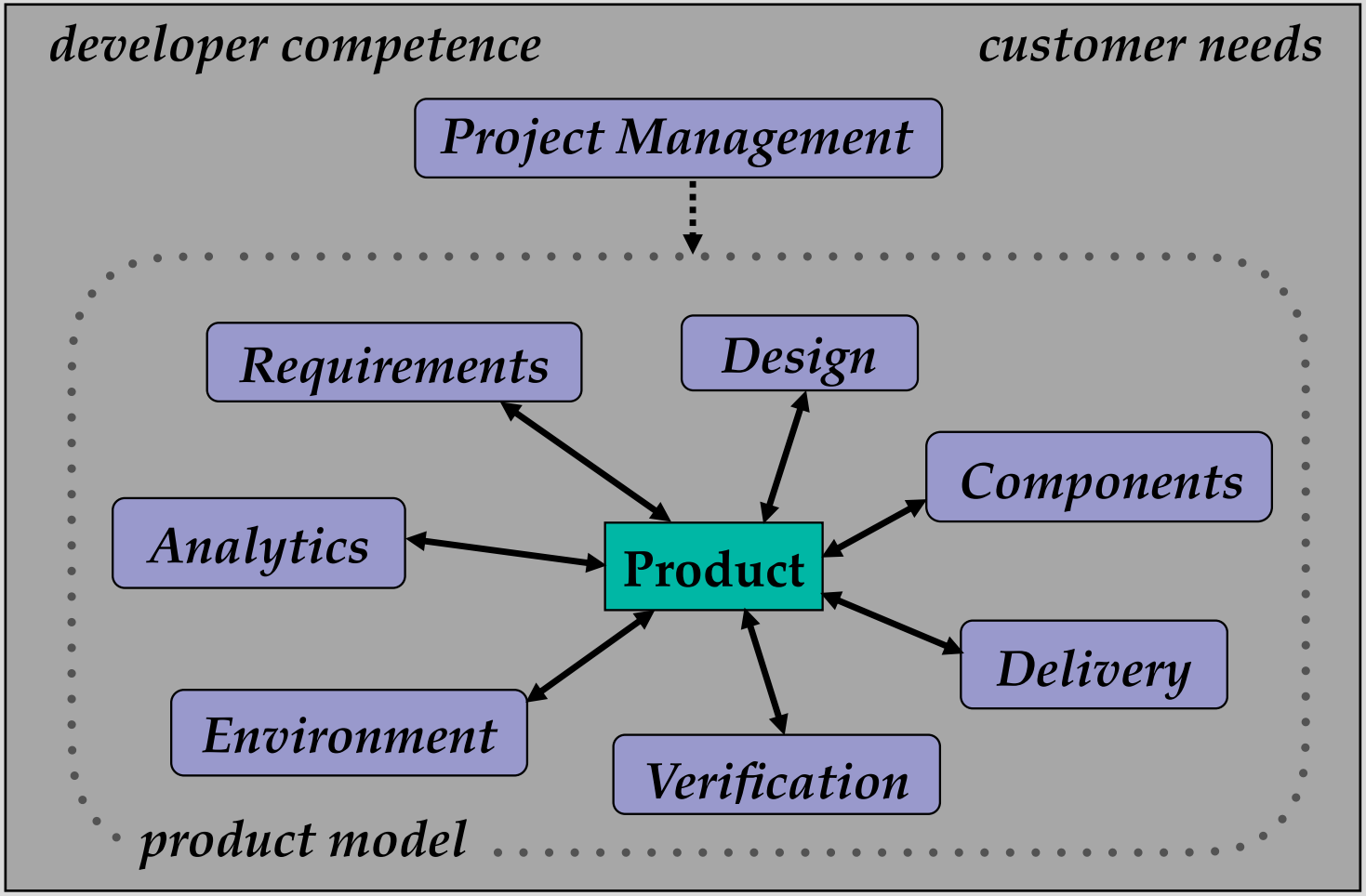
Methodological Themes

- Software prescribes behavior that entities convey
- Uncertainty and change are fundamental concerns
- Two formulations
  - Singular evolving products
  - A product family and its instances
- Shared aspects
  - A product model
    - Separation of concerns based on information content/related competence
    - Consistency across elements
  - A flexible process
    - Concurrent
    - Method-Agnostic

Singular Product Formulation

- An independently-managed project builds and sustains an evolving software product
  - Targets a single customer or “simple” market
  - A product that best approximates current & changing customer needed capabilities
- Project operates in accordance with a Project Model
  => Project Management Model + Product Model
  - Disciplined engineering methods and practices, managed to developmental quality criteria
- Flexible Concurrent, Method-Agnostic Process
  - Tasking to produce a product model
  - Constrained as needed
- Product Versions and Variants ≈ Product Family
- Serves as a Transition Path to a Product Family Formulation

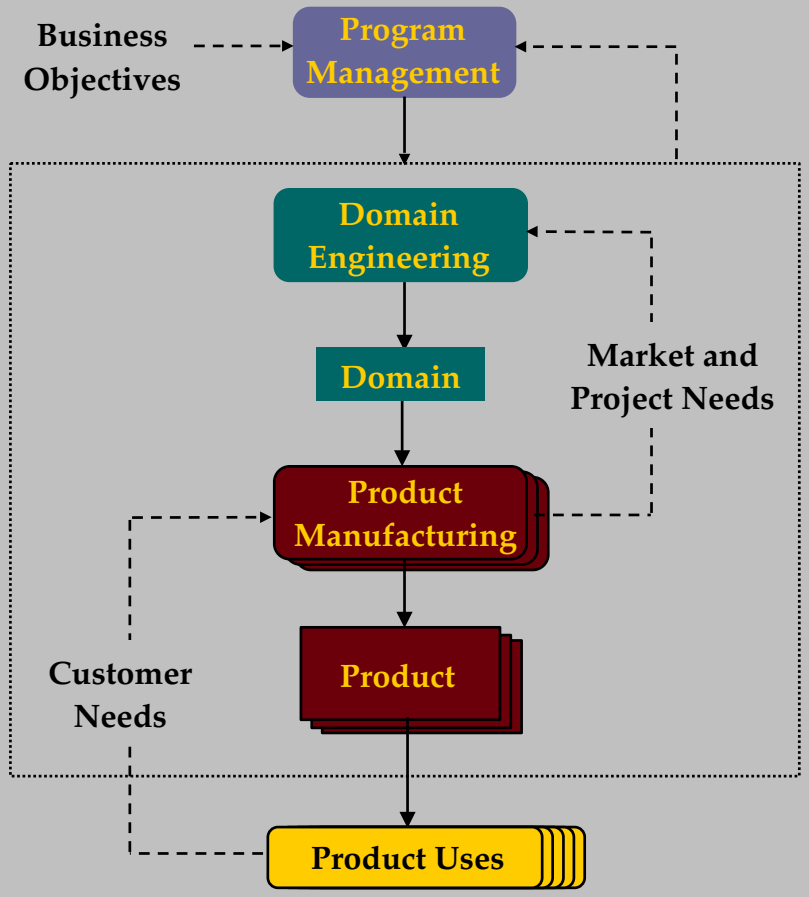
A Software Project Model



Product Family Formulation

- Enterprise investment for Market competence
- Conceives a “Domain”
  - Targets a coherent Market
    - An added cause of product “change”: Market Diversity
  - Represents an envisioned set of similar products
  - Deferred decisions express why a market needs multiple products
  - A decision-focused process (and automation) for streamlined building of customized products
- Rapid Development of Instance Products
  - Leverages common content to eliminate effort
  - Product family • decisions -> customized product

The DsE (Concurrent) Tripartite Process



Institute & improve a product family business

Create a domain (process, tools, and assets) for building similar products

Build tailored products for customers

BREAK
(10:30-11:00)

Topic 2

Software Engineering for a Singular Evolving Product

Software Engineering for a Singular Evolving Product

- **A Simple Market** *{1-n customers, equivalent needs, limited customization}*
- **Organizational Context**
- **A Framework for Developmental Quality**
- **A Software Project Model**
  - **Project Management Model**
  - **A Product Model**
    - **Competence-defined Activities**
    - **Concurrent, Method-agnostic Process**

Organizational Context

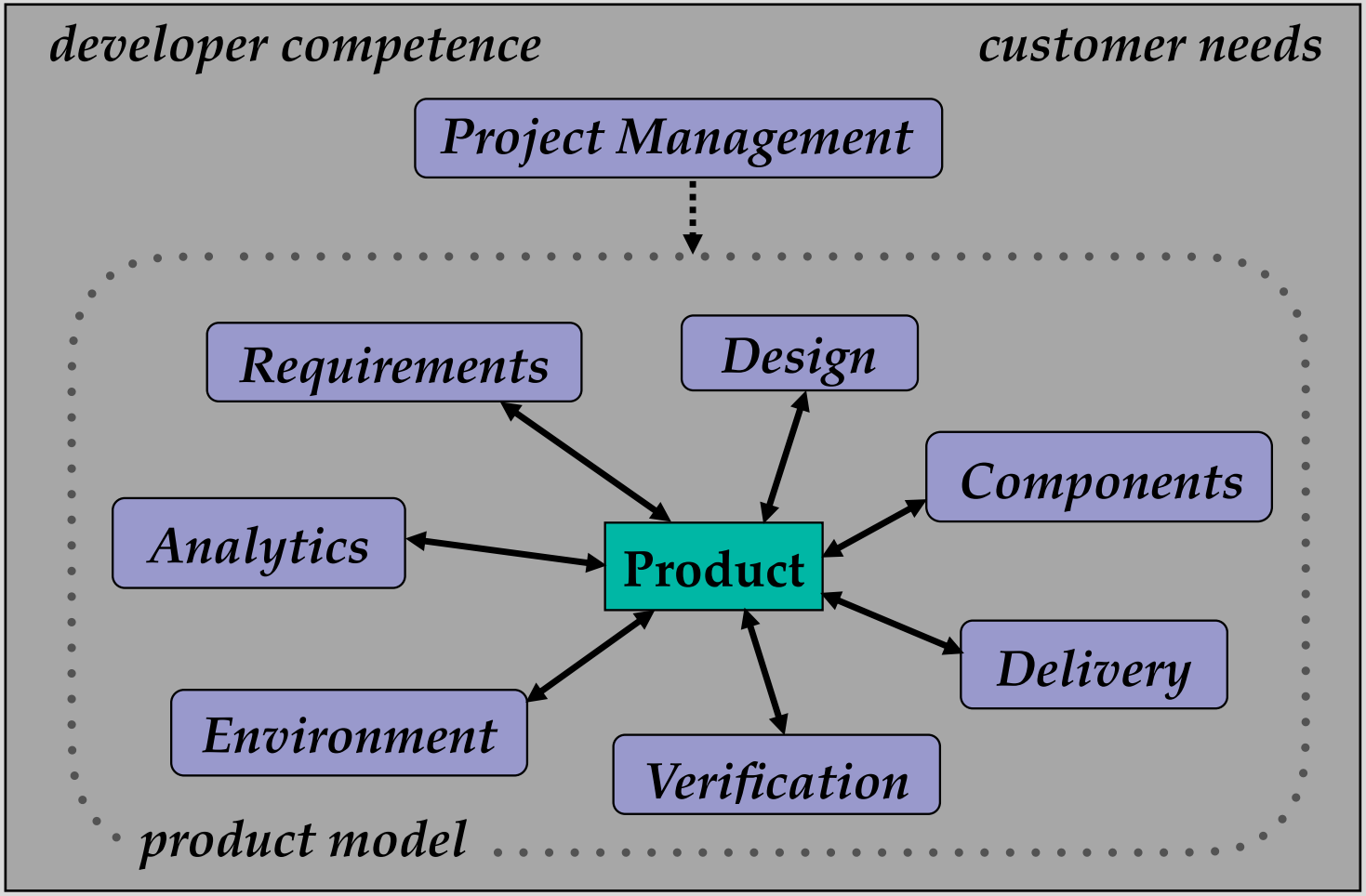
- Enterprise Governance and Executive Management
- Program Management
- Data and Reality (Information)
- Hardware (including Device) Engineering
- Systems Engineering
- Product Implications for Customers
  - Product Acquisition
  - Organizational Transition
  - Product Evolution

Process (Developmental) Quality

Productivity in Building an Effective Product

- **Feasibility** *(Buildability, Deployability, Integrability, Supportability)*
- **Sustainability** *(Maintainability, Configurability, Portability, Evolvability)*
- **Conformability** *(Irreducibility, Modularity, Adaptability)*
- **Verifiability** *(Readability, Observability, Testability, Certifiability)*

A Software Project Model



A Simplified Concurrent WorkFlow

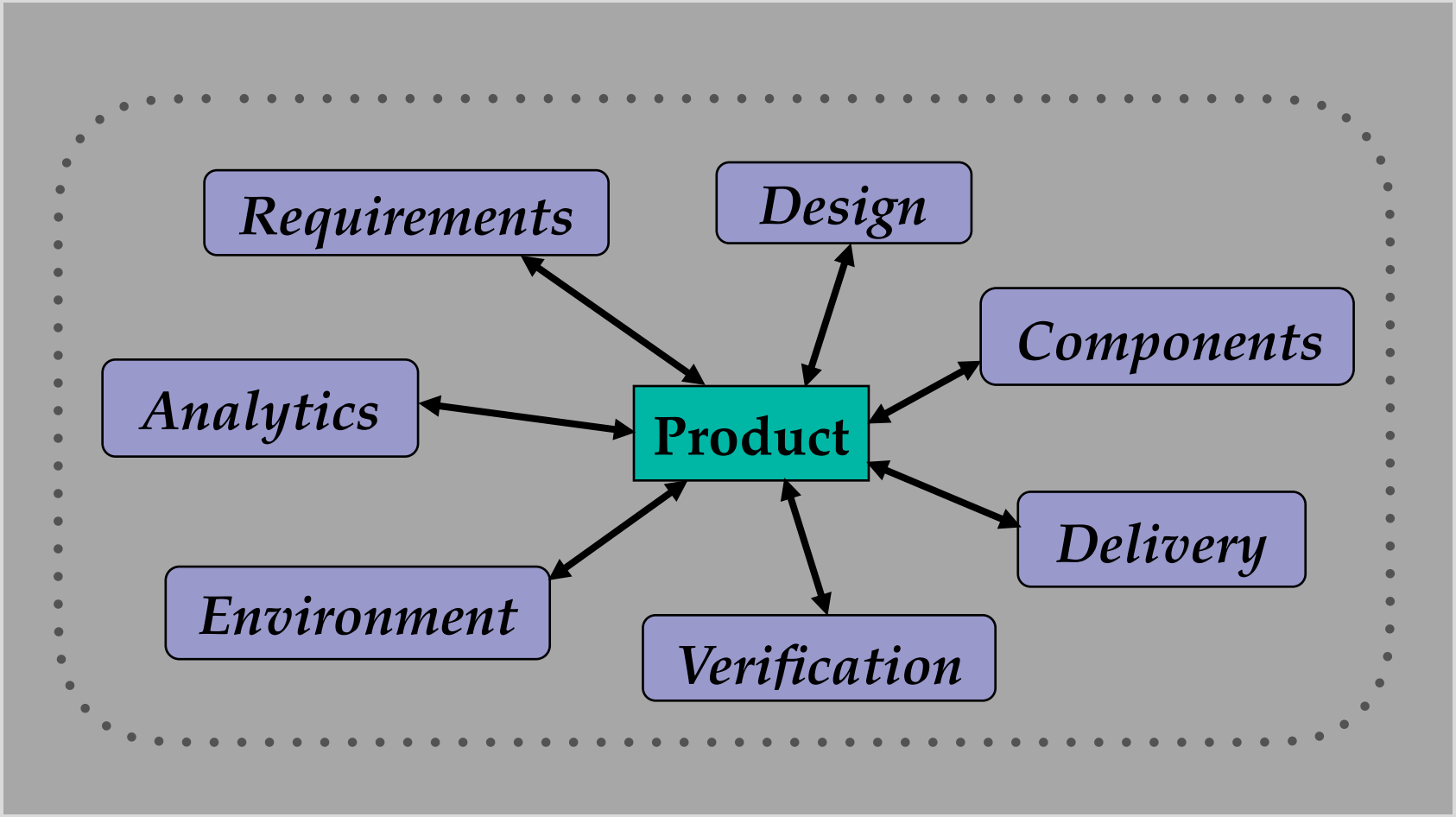
1. **[Project management] a project is instituted to define a product master plan**
2. **[Project management] a product increment is specified with tasks to work on the various product model elements assigned to developers**
3. **[Product delivery] customer needs and product usage scenarios are specified**
4. **[Product environment] the environment information space and relevant entities are specified**
5. **[Product requirements] product observable behavior, with quality criteria, is specified**
6. **[Product delivery] product documentation and training materials for the customer are developed**
7. **[Product verification] a platform for product testing is developed**
8. **[Product design] the product's static and physical structures are specified**
9. **[Product design] the product build platform is developed**
10. **[Product components] software service and platform components are specified (designed, implemented, and verified)**
11. **[Product design] the product's dynamic structure is specified**
12. **[Product components] behavioral components are specified (designed, implemented, and verified**
13. **[Product design] a (partial or complete) product version is built**
14. **[Product verification] the product version is evaluated against scenarios**
15. **[Product analytics] a completed product version is evaluated against behavioral quality criteria**
16. **[Product delivery] a verified product is packaged, validated, deployed to the customer, and supported**

Project Management

Manage 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**

A (Concurrent) Software Product Model



Product Delivery

Coordinate with Customer on Product Expectations, Preparation for and Transition into Effective Use

- **Customer Needs**
- **Product Documentation**
- **Product Configuration and Packaging**
- **Product Acceptance and Deployment**
- **User Support and Feedback**

Product Environment

Specify 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

Product Requirements

Specify the Product's Expected Observable Behavior

Product Perspectives

- **Concept** *{Describes any potential solution}*
- **Product Behavior**
- **Behavioral Quality and Tradeoffs**
- **Constraints**

Product Evolution: Guidance to developers on addressing uncertainties, alternatives, and likely changes

Product Behavior

Manage Coordinated Entity Actions to Achieve Behavior

Conditional Activity Activation, Continuation, and Termination

- **Entity Activity (1 for each Entity "service")**
  - **Conditional Action Selection**
  - **Action Formulation, by Entity Type**
- **Composite Activity (Coordination and Control of Designated Activities)**
  - **Conditionally Sequenced, Selective, or Concurrent Initiation**

Product (Behavioral) Quality

Specify Quality Criteria for Product to be Fit for its Intended Use

- **Functionality** *(Utility, Interoperability, Adaptivity)*
- **Performance** *(Responsiveness, Efficiency, Throughput, Scalability)*
- **Dependability** *(Reliability, Availability, Integrity, Safety)*
- **Usability** *(Conformability, Learnability, Explainability, Aesthetics)*

Product Constraints

Specify extrinsic factors that limit options for the product's realization

- **External**
- **Customer-Imposed**
- **Enterprise-, Program-, or Project-Imposed**

Otherwise: Best Engineering Judgement

Product Design

Specify the Internal 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
  - Deployment Platform Mapping

A Decomposition Hierarchy
(top 2 levels)

Behavior *{observable behavior as specified in the product requirements model}*

- External Interface *{how the product creates behavior via the entities in its ecosystem}*
- Information Model *{the product's model of the ecosystem's composite past/present/future information content as context for its behavior}*
- Introspection Model *{the product's model of its past/present/future behavior for explanation and planning}*
- Enablement *{software capabilities for common elements of behavior such as conventions for entity-tailored presentation of data and media}*

Services *{software capabilities needed to implement other components}*

- Data *{abstract types, structures, object management, analysis, transformation}*
- Reasoning *{math/physical, knowledge/expertise, analytic/statistical/heuristic, and composite/fusion models}*
- Media *{static and dynamic forms, composition, accessibility/dynamics/interaction-control}*

Environment *{the elements of the product's operational environment}*

- Entities *{capabilities for interacting with users, devices, and systems operating in the product's ecosystem}*
- Platform *{the product's foundational capabilities for computation, communication, and data storage}*

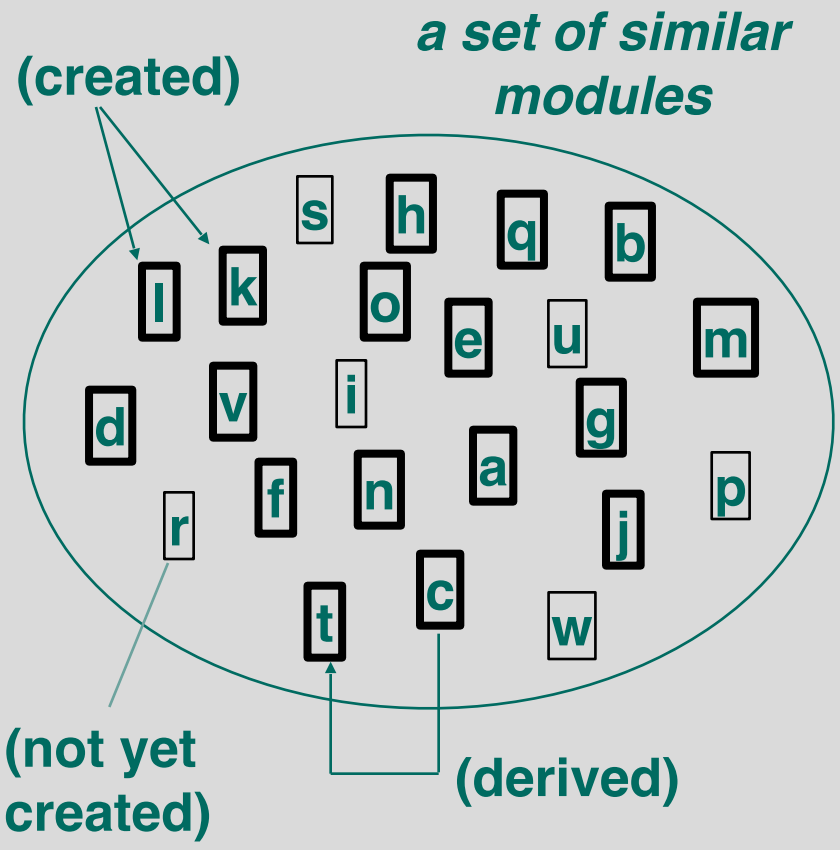
Product Components

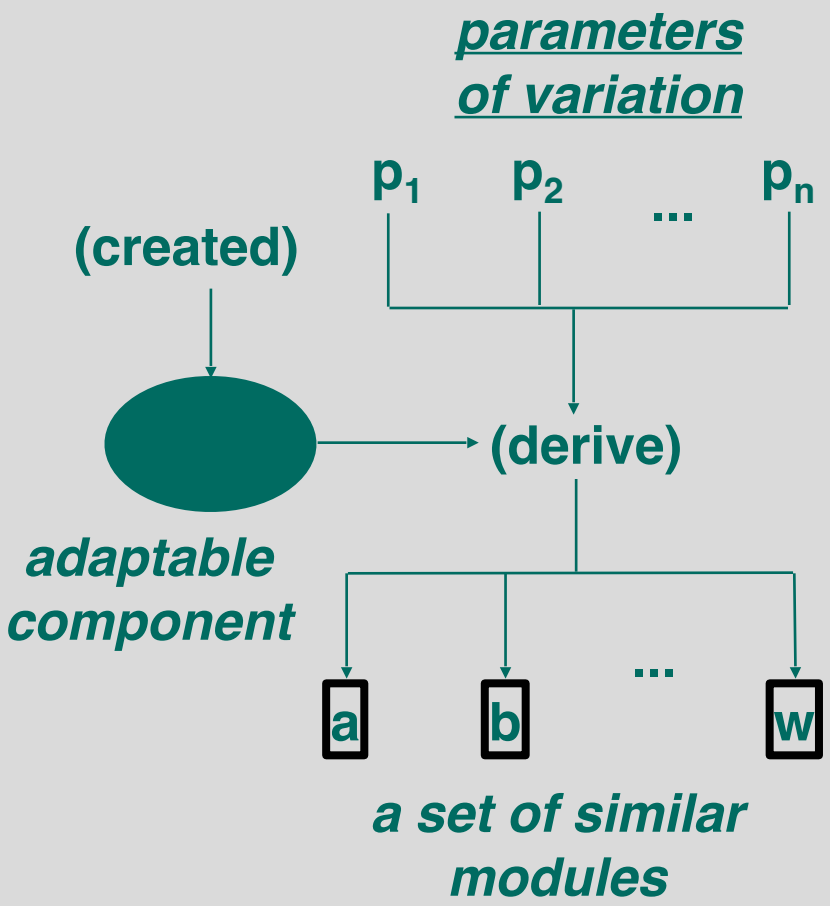
Specify the Components to be Used in Realizing a Product

(For each design-specified component:)

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

Two Views of a Component





Product Analytics

Specify Product Quality Using Applicable Analytic Means and Methods

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

Product Verification

Evaluate Product Model elements in composition

- Consistency and Completeness of Product Model Content
- Product Behavior Conformance to Product Model

Perspectives:

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

LUNCH
12:30-14:00

4 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

<domain-specific.com/EMSP/Tutorial.pdf>

Topic 3

Domain-specific Engineering 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 (Decision-based Product Specification -> Deployable Product)
- Nature of Manufacturing (Producibility)
  - Product Differentiation (Deferred Decisions)
  - Streamlined Process (Productivity)
  - Comprehensive Product Model

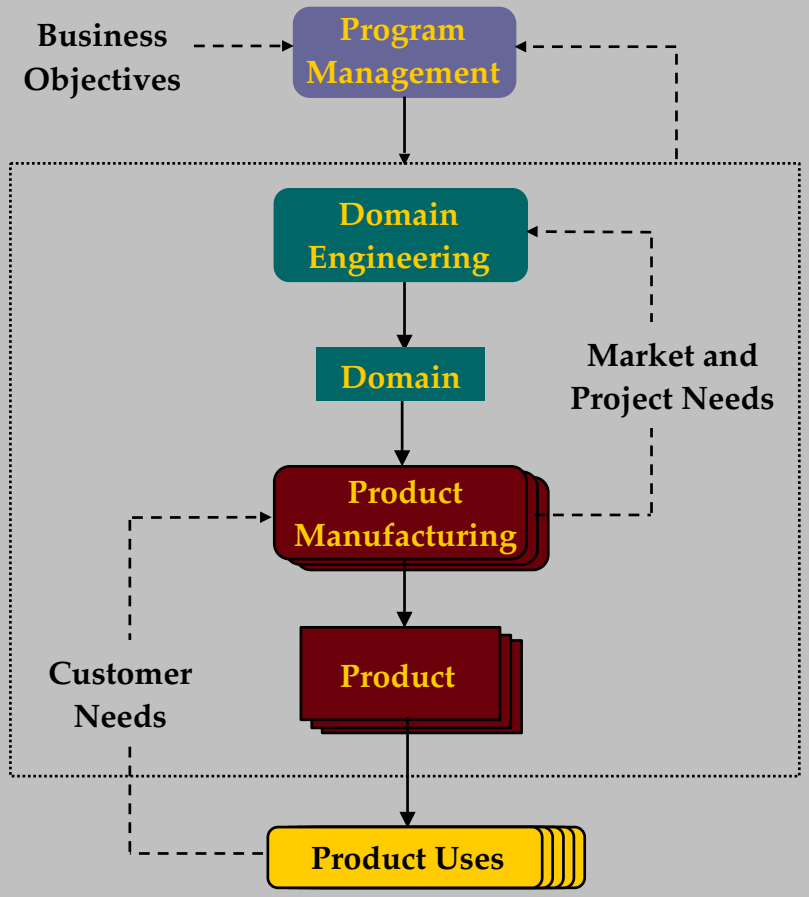
Advantages of DsE Approach

- Cost-reducing standardization:
  - Market-focused business objectives lead to explicit limits on product diversity.
  - The development process is reduced to its essentials for a particular product family.
- Market-responsive flexibility:
  - A product can be tailored to each customer's specific needs.
  - A new product version can be rapidly produced whenever needs change.
  - Alternate versions of a product can be built (or modeled) to let a customer compare.

Transitioning From Singular Products

- Enveloping Organizational Context
- Engineering (and Management) Discipline
- Coherent Market matching Developer Competence
- Product Model specialized to Preferred Methods & Practices
- Process & Product Quality Criteria & Metrics
- Concurrent Process for Product (=> Product Family Engineering)
- Component {Alternative Modules} => Adaptable Component

The DsE Tripartite Process



Institute & improve a product family business

Create a domain (process, tools, and assets) for building similar products

Build tailored products for customers

Enhanced Context

- Duality of Diversity and Change
- Elements of Singular Product Organizational Context
  - Program Management
  - Systems Engineering
  - Hardware Engineering
  - Project-Customer Collaboration
- Developmental & Operational Platforms
- Program-Market CoEvolution
- Domain Investment and Technical Debt
- A DsE Metrics Strategy

DsE: Program Management

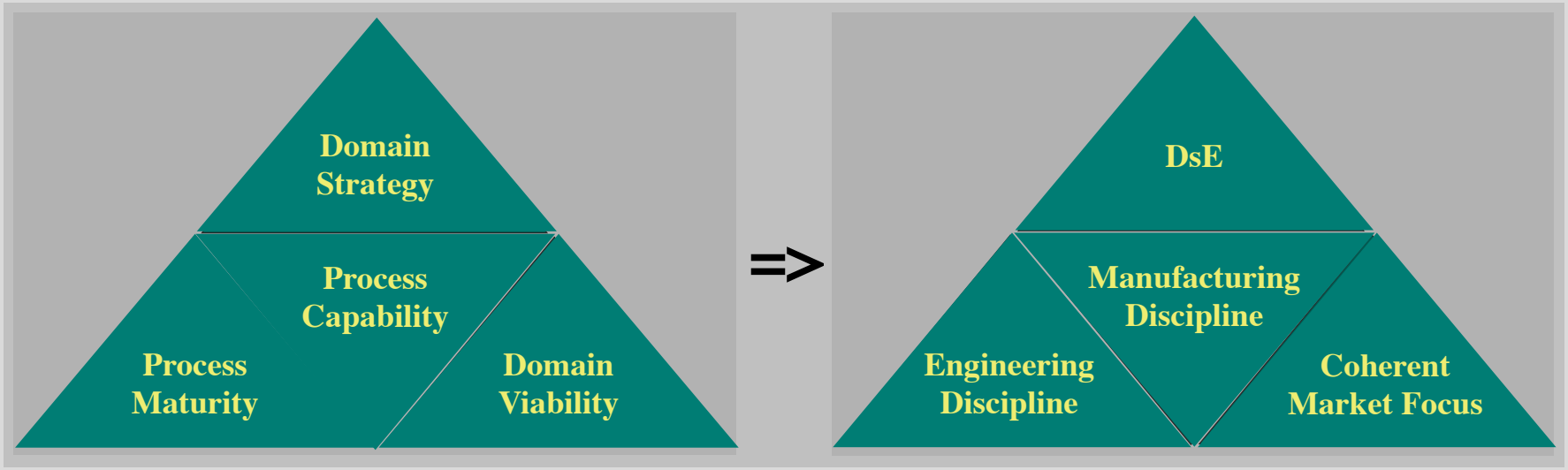
Institute and Improve a Product-Family Business

- **Program Governance**
  - **Domain Viability (Market Opportunity, Technical Competence, Business Commitment)**
  - **Process Capability [Metrics & Tradeoffs] => 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

(Reuse-driven Process Improvement [PI_r] Method)



Program Management Models

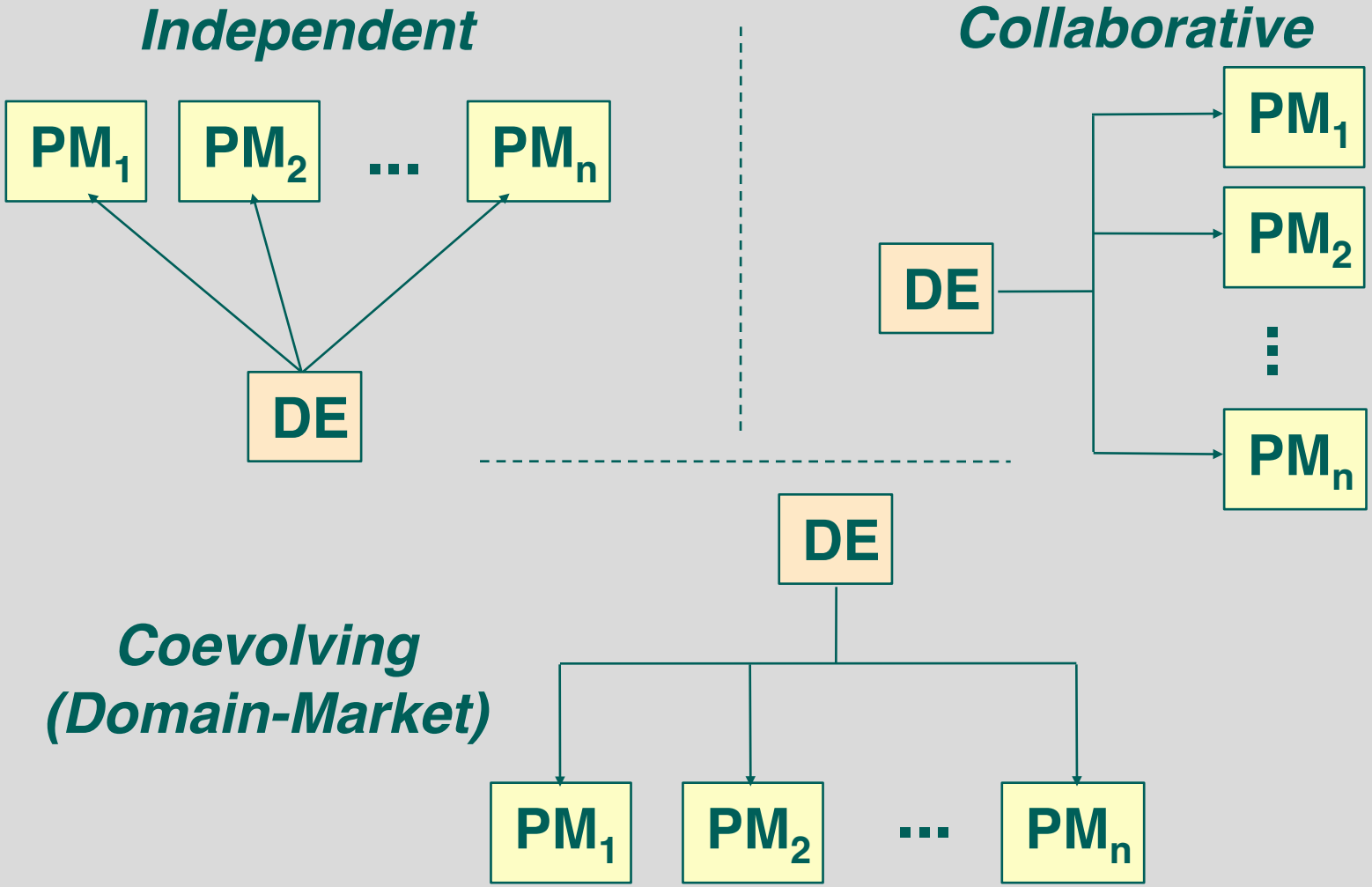
Corresponding Objectives

Economic Profile for a DsE Program

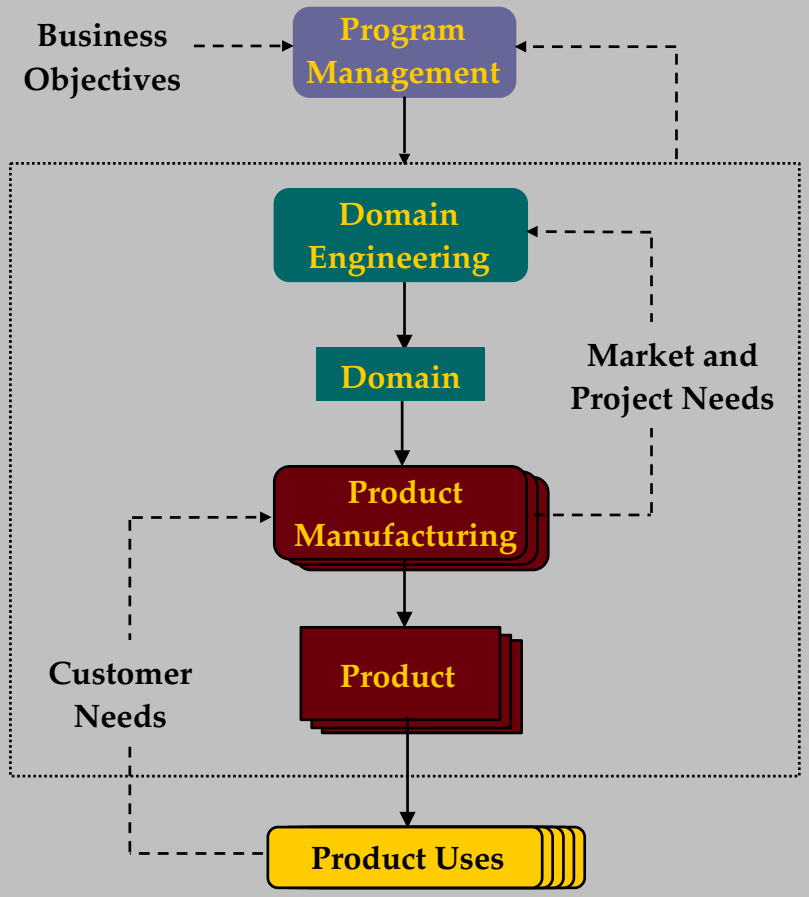
- Predictability with Reduced Effort and Uniformity for Increased Product Quality
  - Capital Asset: Problem-Solution Competence
  - Domain Engineering
    - Eliminates Non-Essential Product Differences
    - Streamlines/Subsumes Recurring Product Manufacturing Effort
- Tradeoffs:
  - Domain Focus on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

Program Governance:

Alternative Organizational Structures



The DsE Tripartite Process



Institute & improve a product family business

Create a domain (process, tools, and assets) for building similar products

Build tailored products for customers

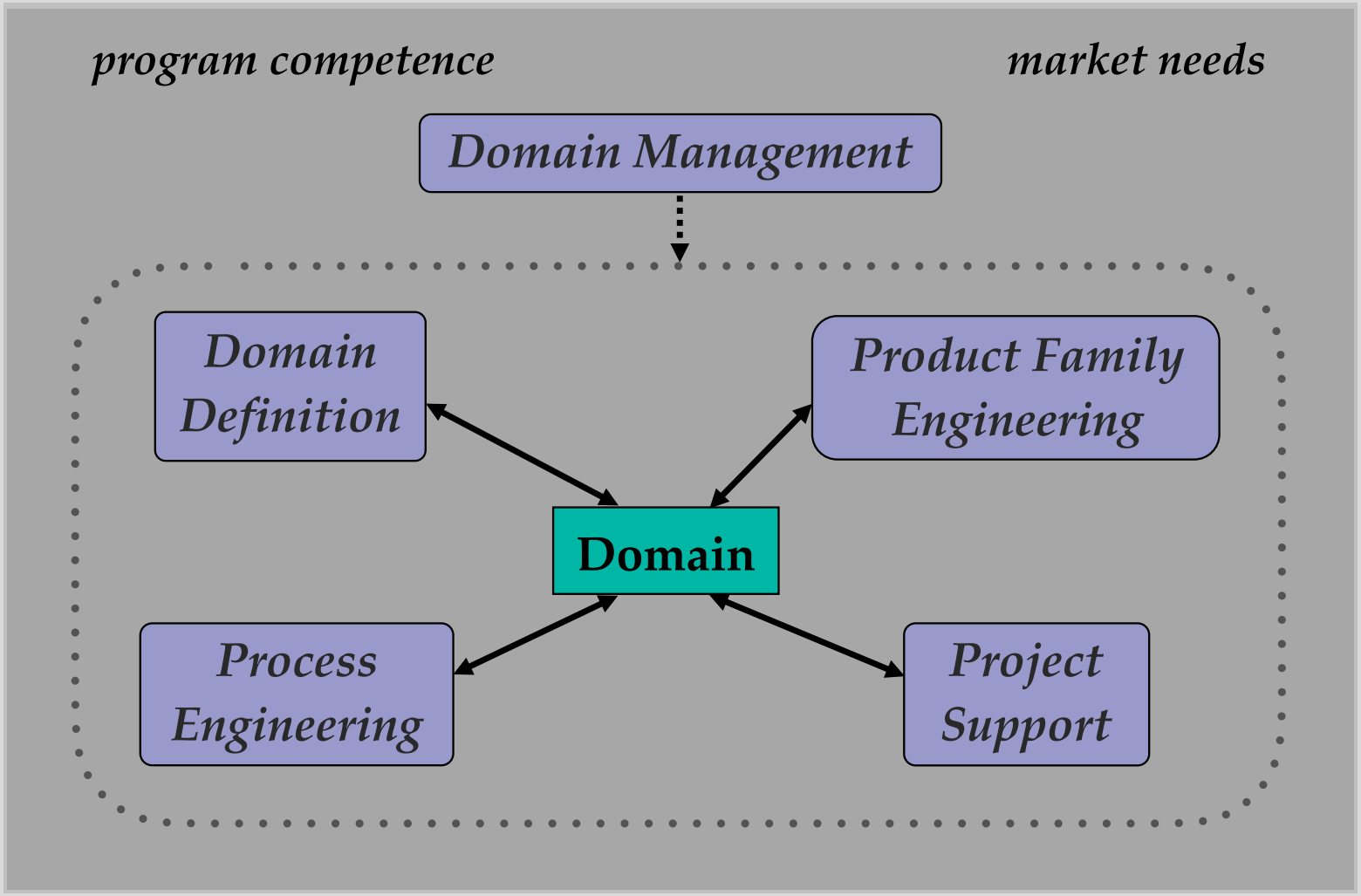
DsE: Domain Engineering

Create a Domain for Building Similar Products

- **Domain Assumptions -> Deferred Decisions => Product Specification**
- **Product Family**
- **Product Manufacturing Process (for affiliated Manufacturing Projects)**

[product family • product specification
=> product model]
-> product

DsE Domain Engineering Process



Domain Management

Technical Agent for Program Management

- Domain Direction
  - Domain Strategy {per Program Management}
  - Domain Process + Methods & Quality Criteria
- Domain Planning
  - Program-Market <= Project-Customer Needs
  - Domain Master Plan {over Program Life}
- Increment Performance
  - Process Iteration to Baseline/Delivery
  - Coordination with Dependent Projects

Domain Definition

Establishing a Unified Domain Vision

- Domain Synopsis
- Domain Terminology
- Legacy Assets (e.g., existing/prior products}
- Assumptions (Alternating Hierarchy of Commonalities and associated Variabilities)
- Decision Model {Assumptions-based Deferred Decisions}

The Role of Deferred Decisions

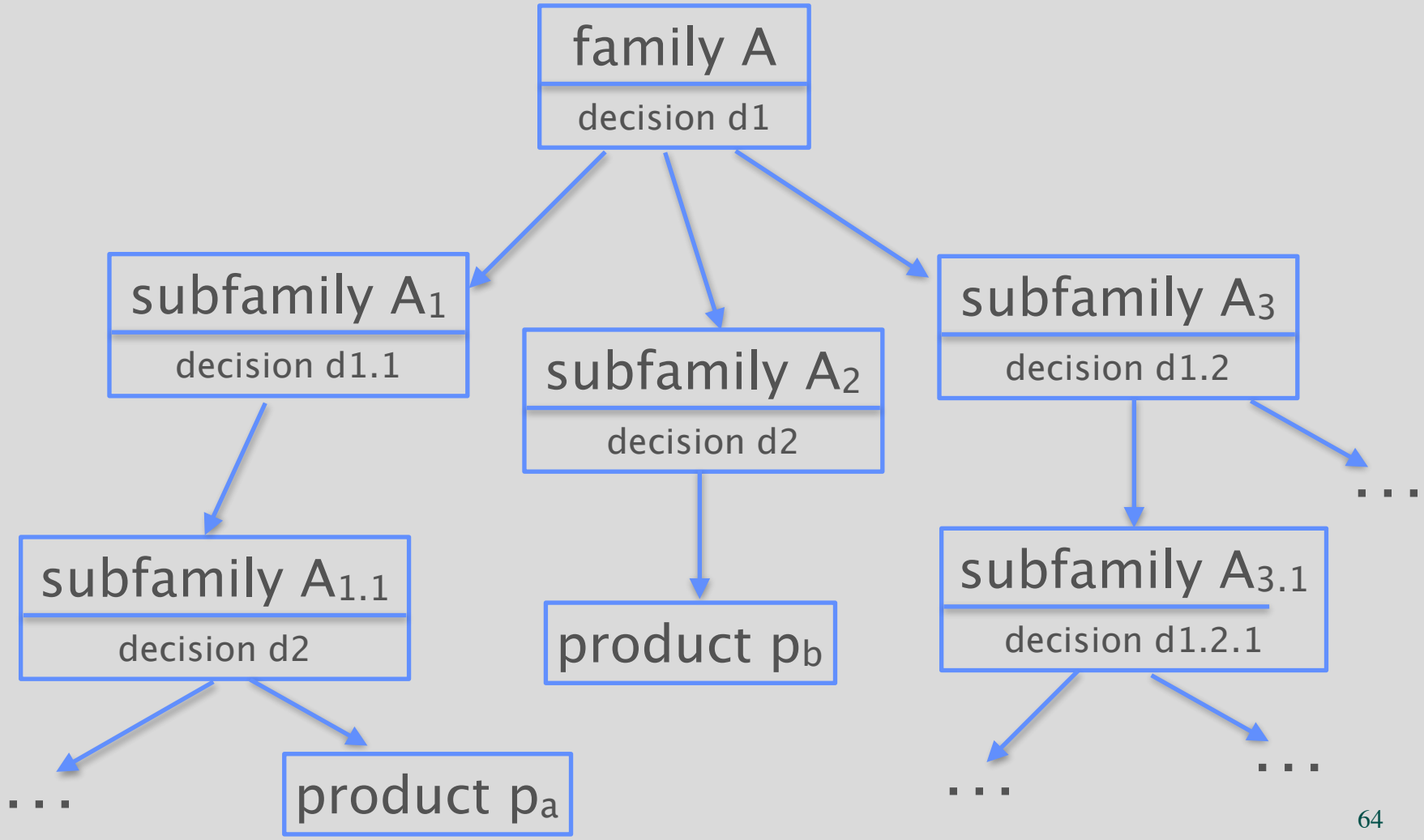
- Development is a decision-making process
- Decisions represent:
  - Customer-specific needs and constraints
  - Engineering tradeoffs to achieve quality criteria
- A focus on similar problems {a family} => commonalities-based standardization {reduces number, variety, and complexity of deferred decisions}
- 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 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 specification** *{resolved decisions}*
  - **product family (product specification) -> product model**
- **Verifying a Product Family**
  - **Directed Reviews, Selective Instance Testing, Family-level Analytics**
- **Incrementally Evolving a Product Family**
  - **Evolving Market, Evolving Customer Needs**

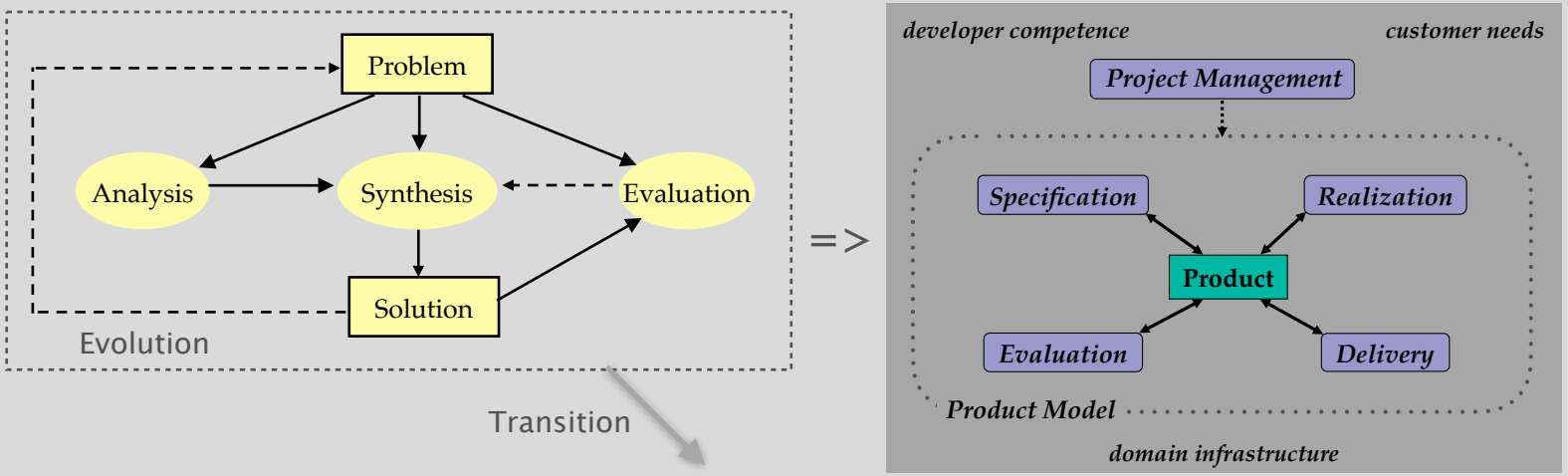
Instantiating a Product Family



Process Engineering

Prescribing How Manufacturing Projects Will Work

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



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

Project Support

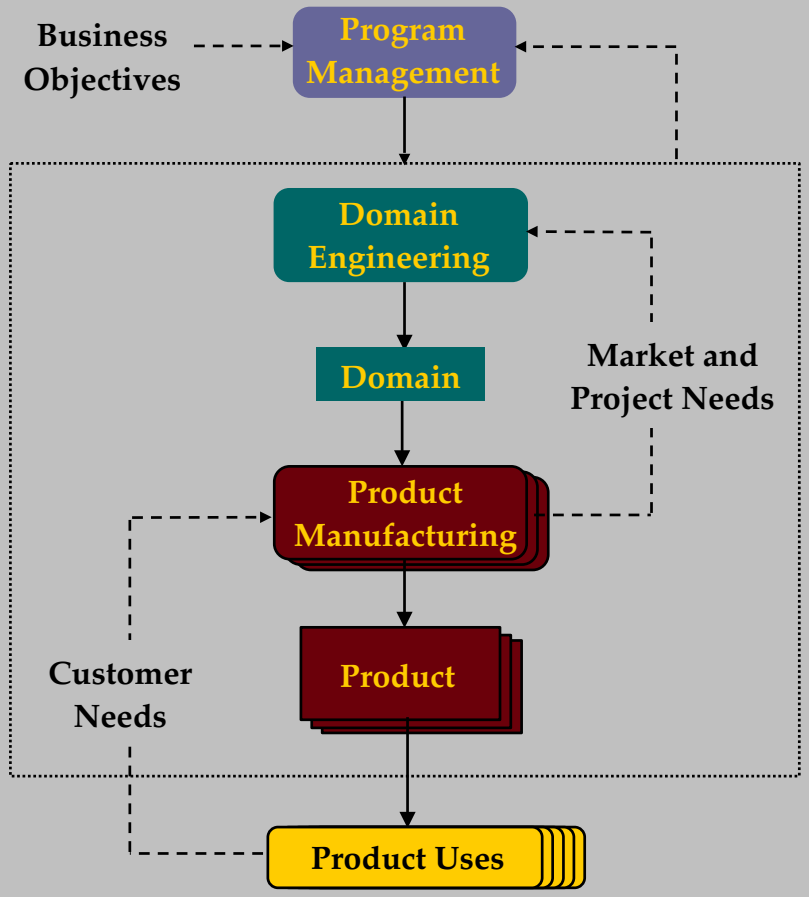
Coordinated Assistance for Associated Manufacturing Projects

- **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**

BREAK

15:30-16:00

The DsE Tripartite Process



Institute & improve a product family business

Create a domain (process, tools, and assets) for building similar products

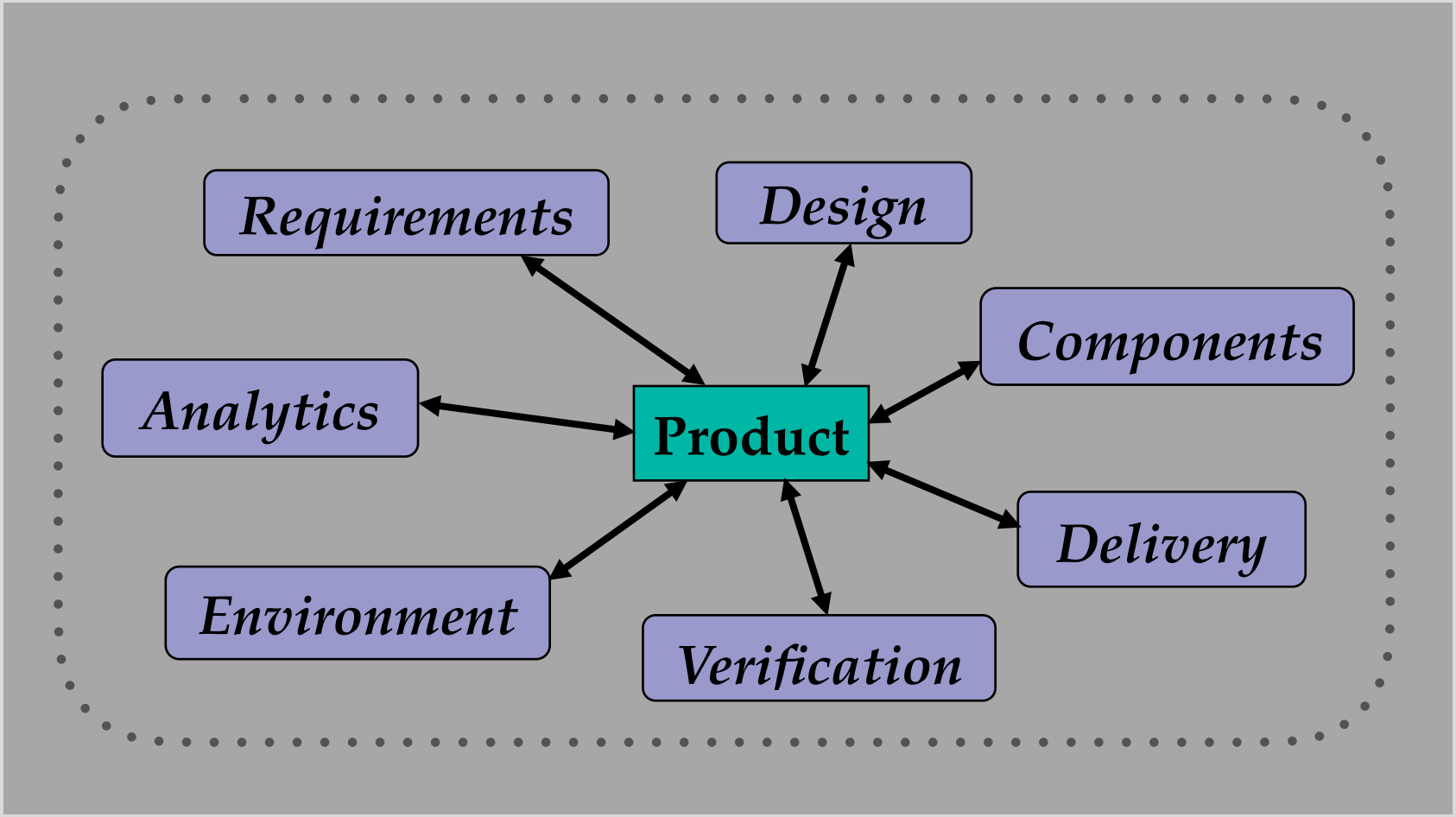
Build tailored products for customers

DsE: Product Manufacturing

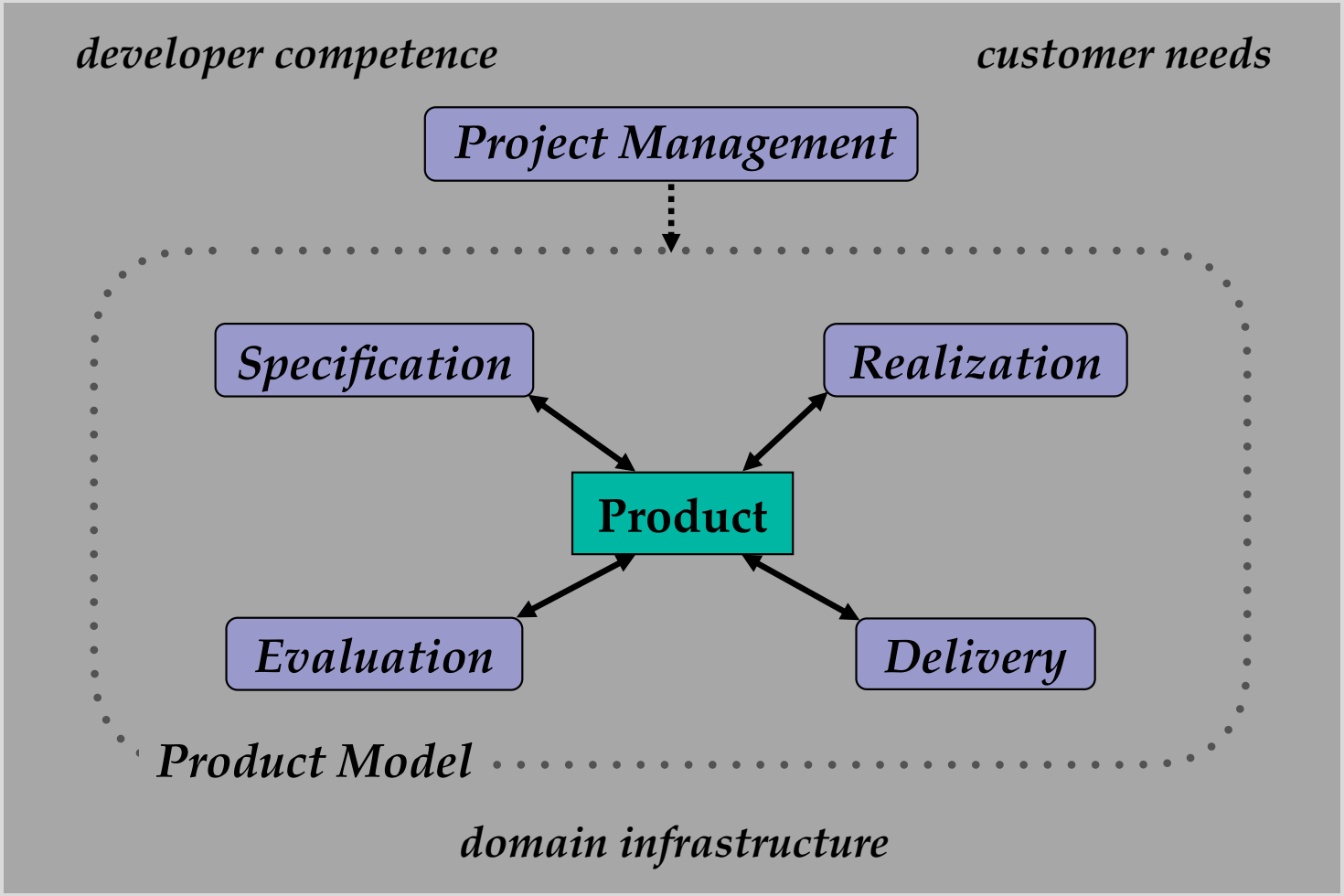
Build and Sustain a Customized Product

- **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 Singular Software Product Model



A Normative Manufacturing Process



product family (product specification) -> product model

Project Management

Coordinate Resources to Derive a Product

- **Project Direction**
  - **Determine Fit of Customer Needs to Domain**
- **Product Planning**
  - **Identify Product Increments for Progressive Completion**
- **Increment Performance**

Product Specification

Specifying a Product in terms of Deferred Decisions

- **Specification presents decisions in the form of a domain-specific language**
- **Deferred decisions are resolved based on analysis of customer needs** *{each decision reduces the set of buildable products to a subfamily}*
- **Unresolved decisions => uncertainties**
- **Tradeoffs => alternative decision resolutions**
- **Domain limitations => excluded options**

Product Realization

Deriving a Total Product for Evaluation & Use

- **Product Family • Resolved Decisions**
  => Product Model -> Product
- **Inferred Quality Characteristics**
- **Virtualized Operational Environment/Entities for Evaluation**
- **Limitations versus Needed Capabilities**

Product Evaluation

Evaluating Product Model to Specification and to Needs

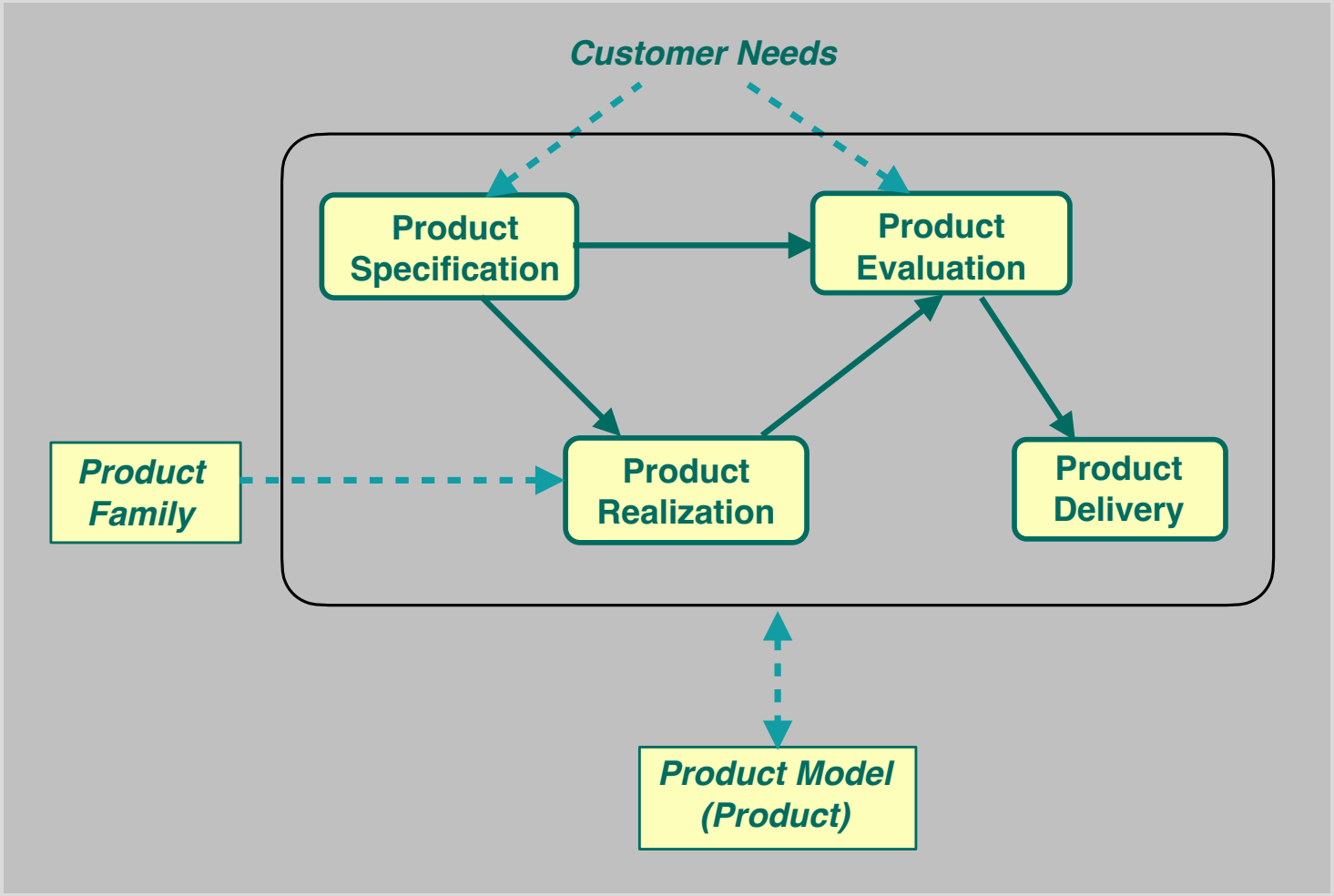
- Validation (Review/Evaluate Product Specification)
  - Consistency
  - Completeness
  - Domain Limitations
- Verification (Product Capabilities & Quality versus Customer Expectations)
  - Capabilities correlate to Product Specification
  - Scenario-based Testing
  - Static & Dynamic Analyses of Product Behavior
  - Product Family Discrepancies
- Comparative Analyses of Alternative Products

Product Delivery

Supporting Customer in Obtaining and Using the Product

- Customer Expectations/Needed Capabilities
- Support for Product Acceptance
  - Installation Materials and Assistance
  - Support for Validation and Certification
- Customer Transition Preparations
- Support for Product Use
  - Materials and Assistance
  - Feedback (Deficiencies, Potential Improvements)

A Simplified Workflow



Topic 4

Automation and Future Technologies

Reuse-enabling Capabilities

Anticipating Technologic Change

Reuse-enabling Capabilities

- Opportunistic and Systematic Reuse
  - As-Is, Modified, and Library-based Reuse
  - Building Parts for Multiple Uses and Change
    - Source Transformation *{Prescriptive}*
    - Adaptable Components *{Descriptive}*
    - (Pending: Generative AI)
- Abstraction-based Environments
- Domain-specific Environments

Basic Tenets of 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.

Systematic Reuse: Adaptable Components

- **Component => 1-n Alternative Design-conformant Modules**
- **Module => Customized (Textual) Code, Test, or Document Content**
- **Adaptable Component => Aggregated Set of Similar Alternative Modules**
  - **Parameterized for deriving instances** *{in DsE, parameters map from decisions}*
  - **Descriptive => Predictable Content**

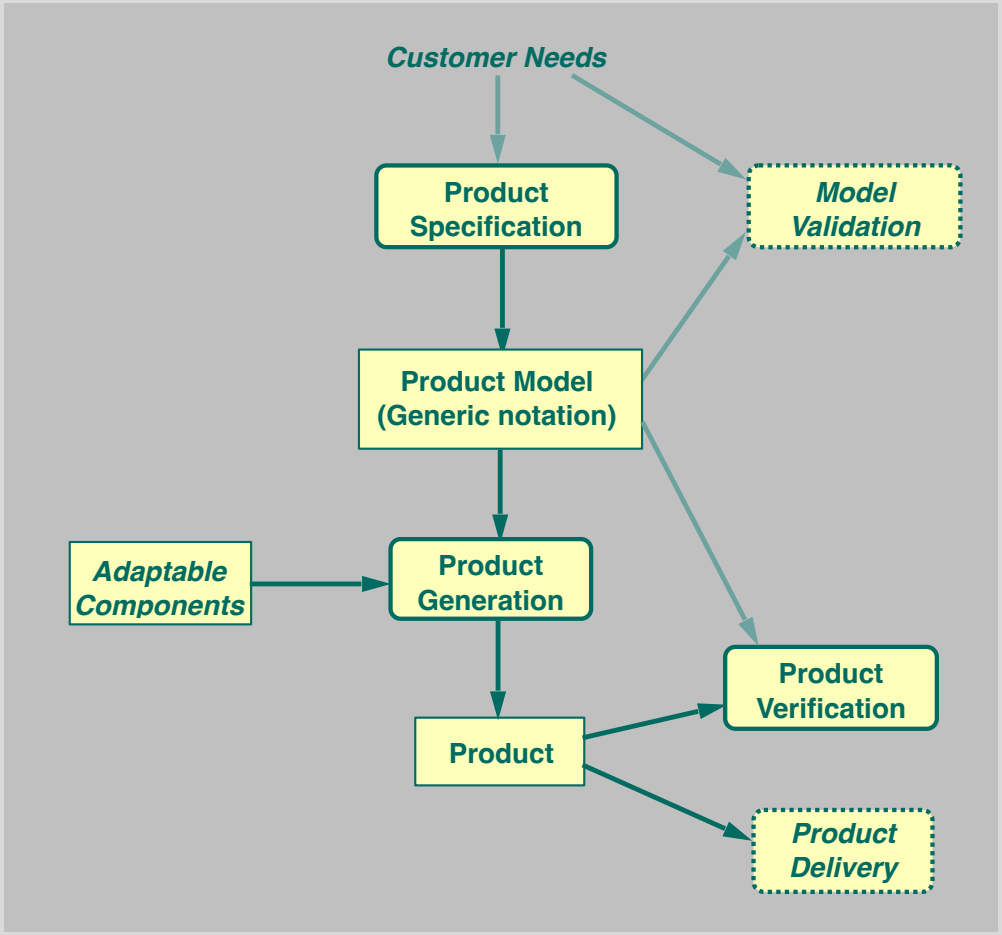
{side note: a generative AI could be conceived to be equivalent to an adaptable component}

Abstraction-based Environments

- Characteristics
  - Product specification using a semi-formal requirements notation to define behavior
  - Associated set of computational abstractions (=> components)
  - Encapsulated software product design with adaptable component per abstraction
  - Fully automated product generation with instantiated abstractions
- Spectrum Environment (Formative/Proof of Concept)
  - Most 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}*

Anticipating Technologic Change

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

Developmental/Operational Practices

- Developmental Capabilities
  - Methods (Engineering Discipline)
  - Techniques and Tools (Problem-Solution Analyses)
  - Enhanced Pre-Deployment Evaluation
  - Alternative Versions and Variants Support
- 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 Modules 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

Predictive Analytics for DsE

- **Envisioned Capabilities**
  - **How does Quality differ among a Component's Modules?**
  - **How does Quality Vary over Instances of a Family?**
  - **Do Subfamily instances have Equivalent Quality?**
  - **How do Decisions affect Quality?**

- **Product Family Analytics** *{Similar products will have similar expected and actual properties (?)}*
  - **Formal Methods Generalized to Intensional Sets {Decisions as Universal Quantifiers?}**
  - **Reviews and Test Scenarios that Address a Set of Semi-Equivalent Products**

Computational Technologies

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

Data Science, Artificial Intelligence, and Machine Learning

- **Historical Context / Foundations**
  - **"Automatic Programming"** *{IJCAI 1977}*
  - **Knowledge-based ("Expert") Systems**
  - **Neural Networks**
  - **Data Analysis** *{for Statistical Similarity}*
  - **Machine Learning**

=> Generative AI

  - **Relation to Reuse**
  - **Potential Concerns** *{specific to software dev.}*
  - **Relevance for DsE**

Generative AI: Relation to Reuse

- A Similar Basis
  - Source Materials: Legacy Software Assets (e.g., Reuse Repositories)
  - Generalizing from Similarity (Proximity-based Patterns) in Source Content
- Distinctions (vs Adaptable Components)
  - Statistically-Derived vs Developer-Envisioned Alternatives
  - Similarity in Form vs Deferred Decisions
  - Probabilistic Categorization vs Instantiating Criteria {Parameters}

Concerns, related to Expert Systems

from Bobrow, Mittal, Stefik, "Expert Systems: Perils and Promises", CACM 28(9), 1986, p 892.

- **Expert knowledge**
  - **is expensive**
  - **is (usually) not in textbooks**
  - **has to be acquired *[learned]* incrementally and tested**
  - **is sometimes distributed (multiple experts)**
- **What are the limits of an expert system's capabilities/competence?**

How do concerns regarding generative AI differ?

Does it scale with problem complexity?

Generative AI: Potential Concerns

- **Reliance on mixed-quality source material** *{defective, missing, privileged, non-compliant, outdated} => Authoritative?*
- **Unpredictability of (probabilistic) resolution among similar|equivalent alternative results**
- **Insufficient deep or specialized competence**
- **Superficial "reasoning" (to answer, explain, and justify)**

"LLMs are not principled reasoners but rather sophisticated simulators of reasoning-like text." (from a UofArizona preprint)

- **Lacking contextual and tacit information of prompts**
- **Excessive cost or time to (re)derive and access content**
- **Progressive shallowing of developer competence**

- **How to determine proper uses of generative AI in products**

Generative AI: Relevance for DsE

{Example nice-to-haves in 4 areas}

Profiling Product Behavior

- **Generate user views of a product's behavior**
- **Detect actions that have not been specified**

Diagnosing Product Quality

- **Analyze & rate component relevance to key quality factors**

Generating Product Content

- **Generate adaptable components** *(Unify 2-n derived similar instances, subsuming equivalences)*

Leveraging Similarity

- **Derive a distance metric for behavioral proximity over a product family**

End

17:30