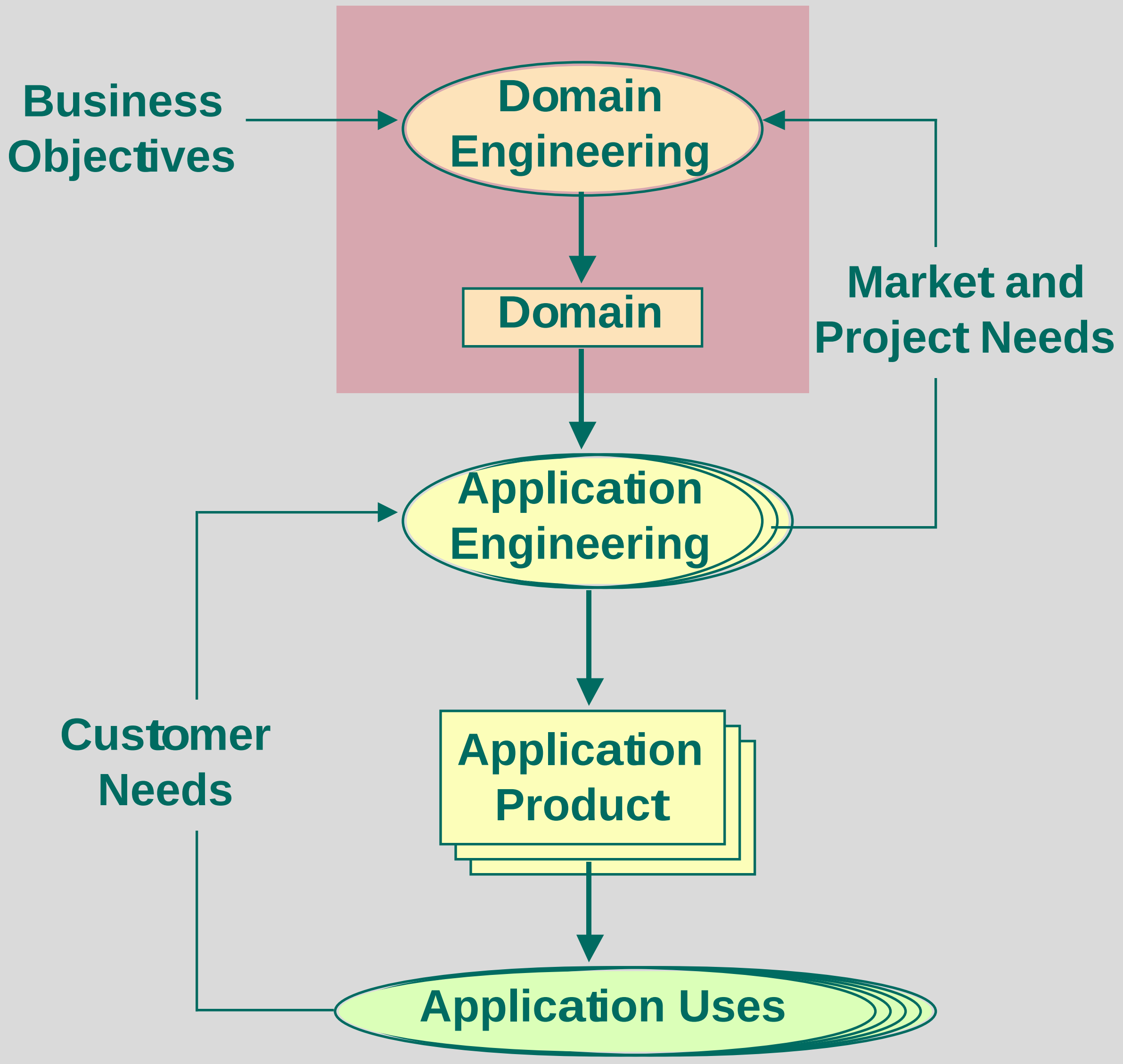
Domain-specific Engineering

Domain Engineering

Domain-specific Engineering



Understanding the Purpose of DE

What is the goal of Domain Engineering?

To institutionalize and improve an organization’s ability to address the needs of a product line market

What is a product line market?

A set of customers having similar needs

How does Domain Engineering accomplish its goal?

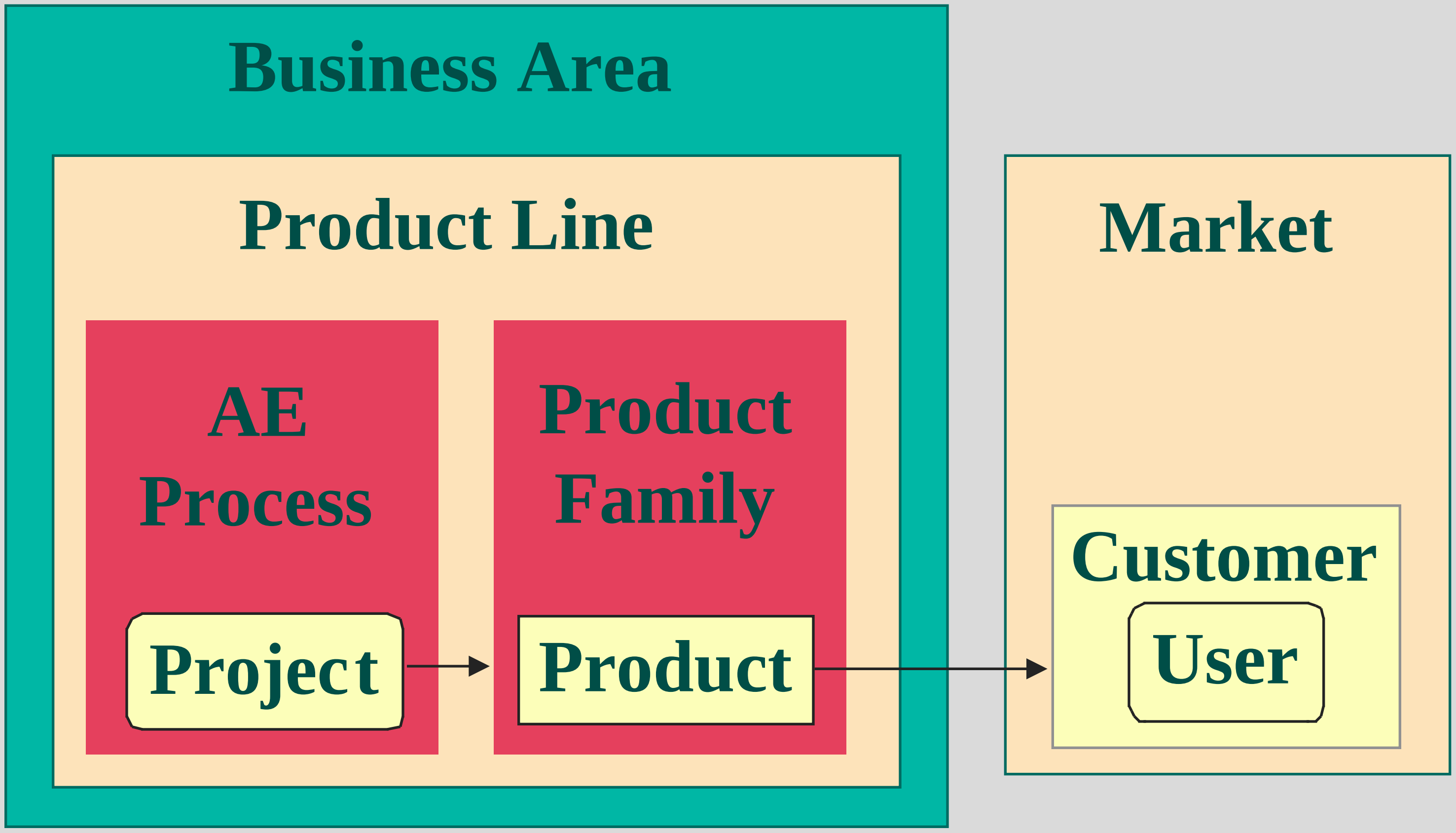
By providing Application Engineering projects with a capability for building application products more effectively (according to organization objectives)

What is the product of Domain Engineering?

A <u>domain</u> and all associated work products

What is a Domain?

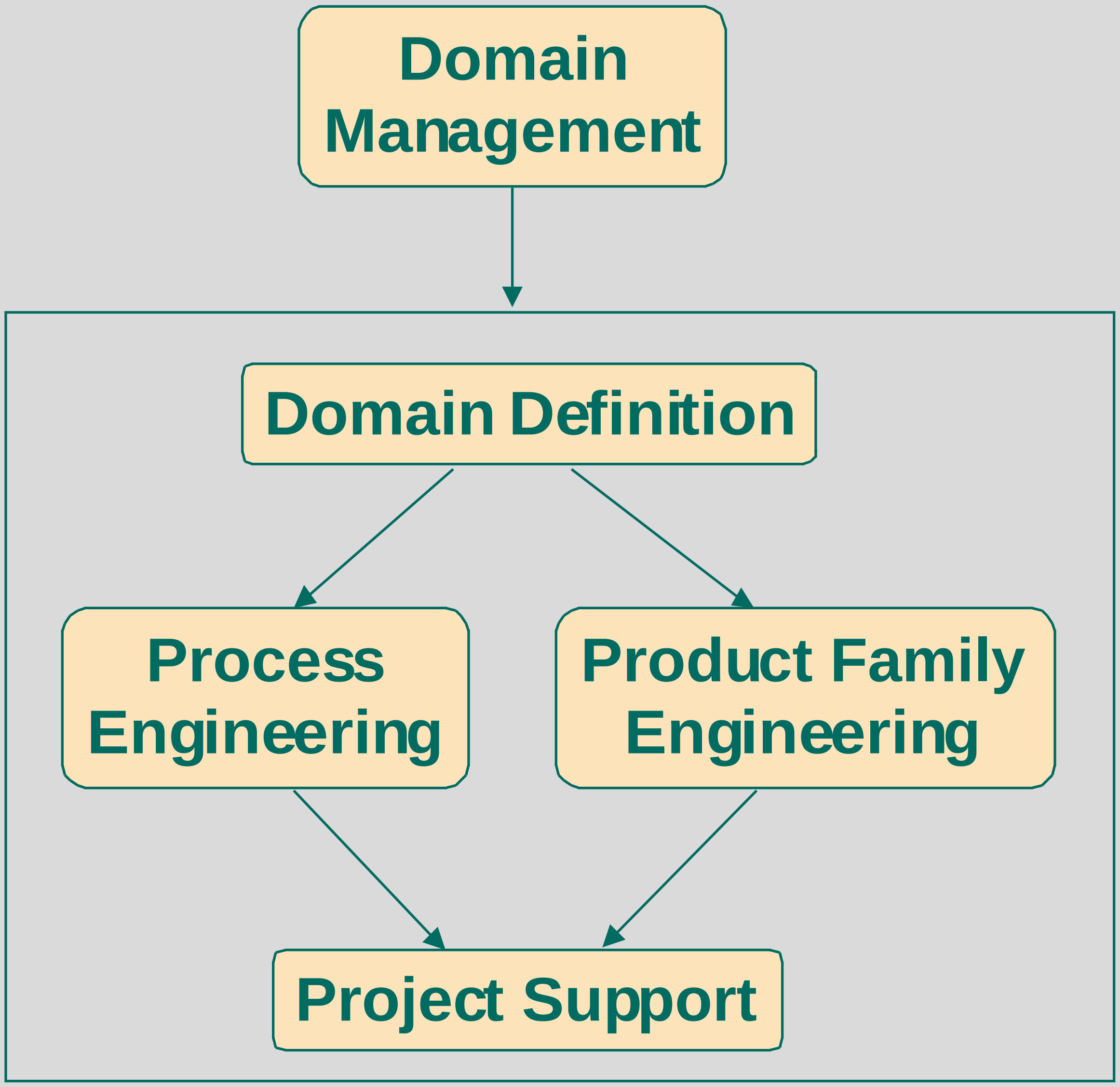
The knowledge (product family) and expertise (process) required to build a particular type of product.



Responsibilities of DE

- Define the strategic direction and capability of the organization to address a product line market
- Institutionalize the knowledge and expertise upon which the organization depends
- Give AE projects the ability to create products at a minimum in cost, time, and divergence from need:
  - Process and procedures
  - Reusable components
  - Tools
- Evolve domain capabilities as market needs change

Domain Engineering Process



Domain Engineering Activities

Domain Management

Organize, plan, and direct domain efforts to achieve business objectives

Domain Definition

Establish the focus and scope of the domain

Product Family Engineering

Develop assets and mechanisms for deriving tailored instances of a product family

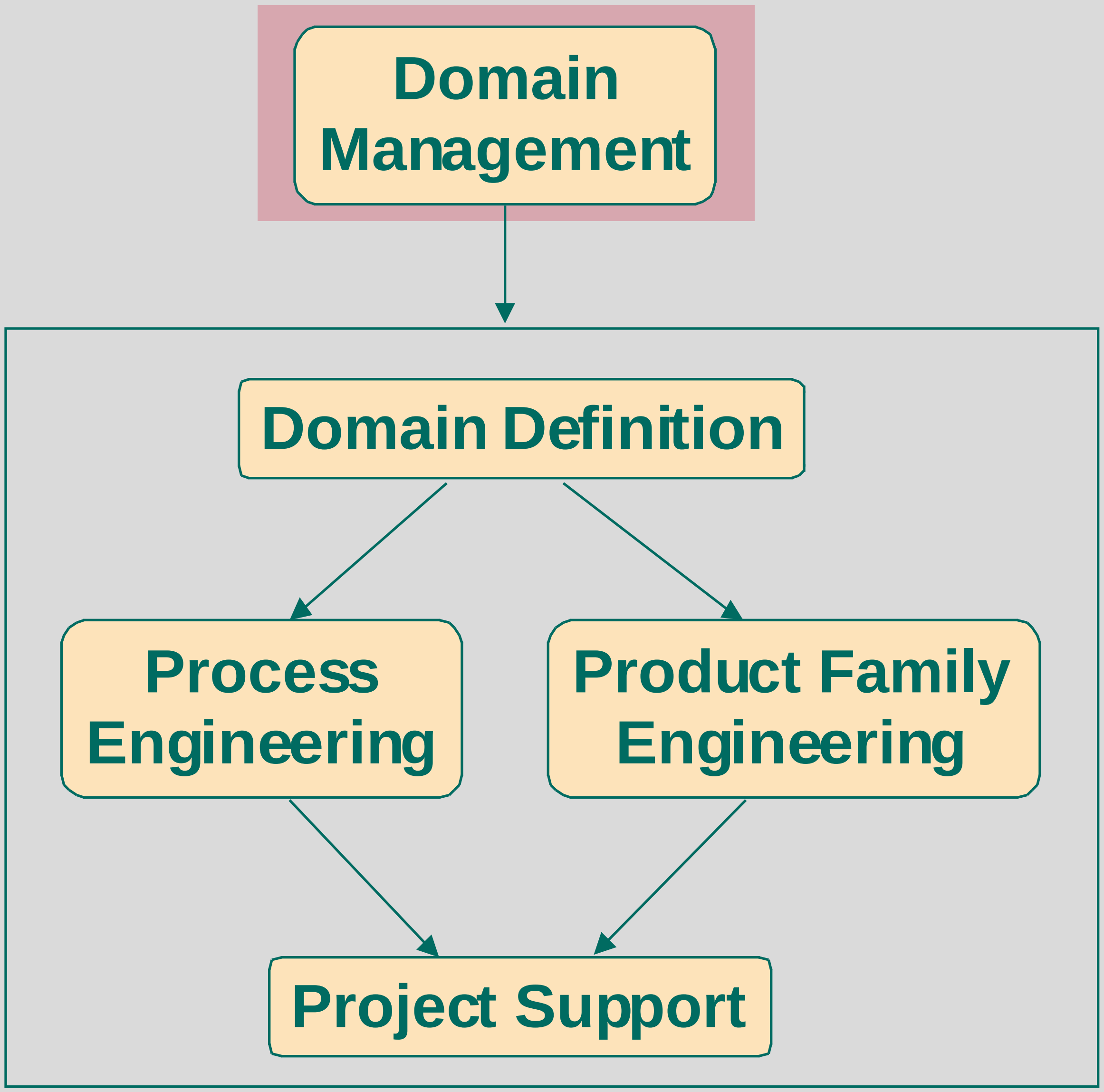
Process Engineering

Define a standardized application engineering process and develop a supporting infrastructure

Project Support

Ensure that the domain meets business, project, and market needs

Domain Engineering Process



Domain Management

Context: **Business objectives**

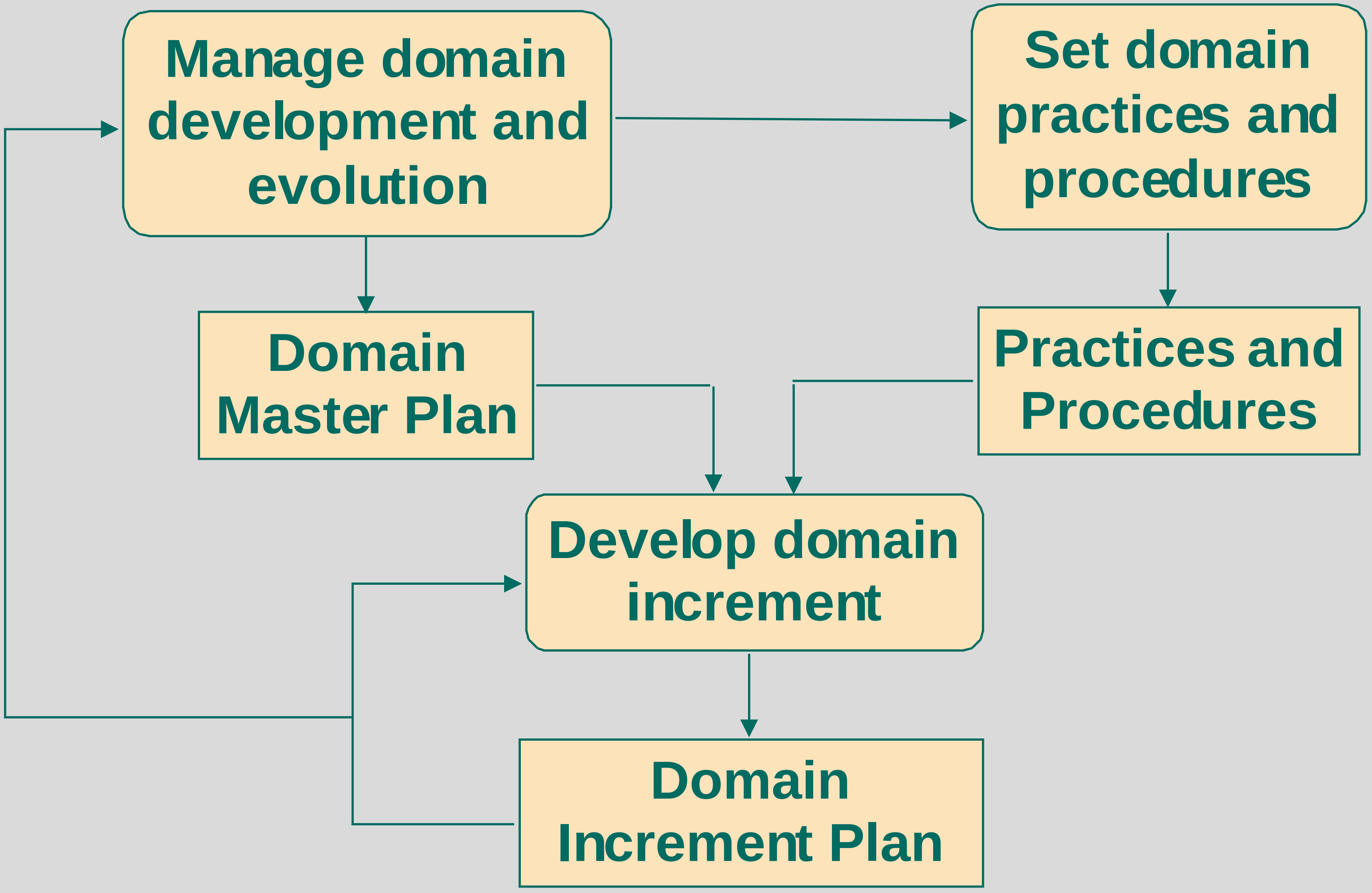
Needed Expertise:

- **Market characteristics**
- **Strategic business development, business area management**
- **Application project management**

Responsibility: **Manage business area resources to achieve business objectives**

Work Product: **Domain Plan (Domain Master Plan, Practices and Procedures, Domain Increment Plans)**

Domain Management Process



Domain Plan

Domain Master Plan

An overview of the scope and life-cycle for a domain suited to business objectives for a targeted market

- **Domain Objectives** *(strategic mission and vision)*
- **Market projections** *(current and future customer needs and opportunities)*
- **Domain life cycle strategy** *(conception, elaboration, expansion, consolidation)*
- **Resource and automation profiles** *(assets available for meeting domain objectives)*
- **Domain development profile** *(domain increments in overview)*
- **Domain status** *(assessment of technical progress and market experience versus expectations)*

Domain Plan
Practices and Procedures

Standards an organization intends to follow in developing a domain.

- **Establish administrative practices**
- **Select preferred development methods**
- **Define documentation standards**
- **Identify guidelines for project management and control**
- **Institute quality assurance mechanisms**
- **Institute configuration management procedures**

Preferred strategy: Adapt current organizational standards

Domain Plan

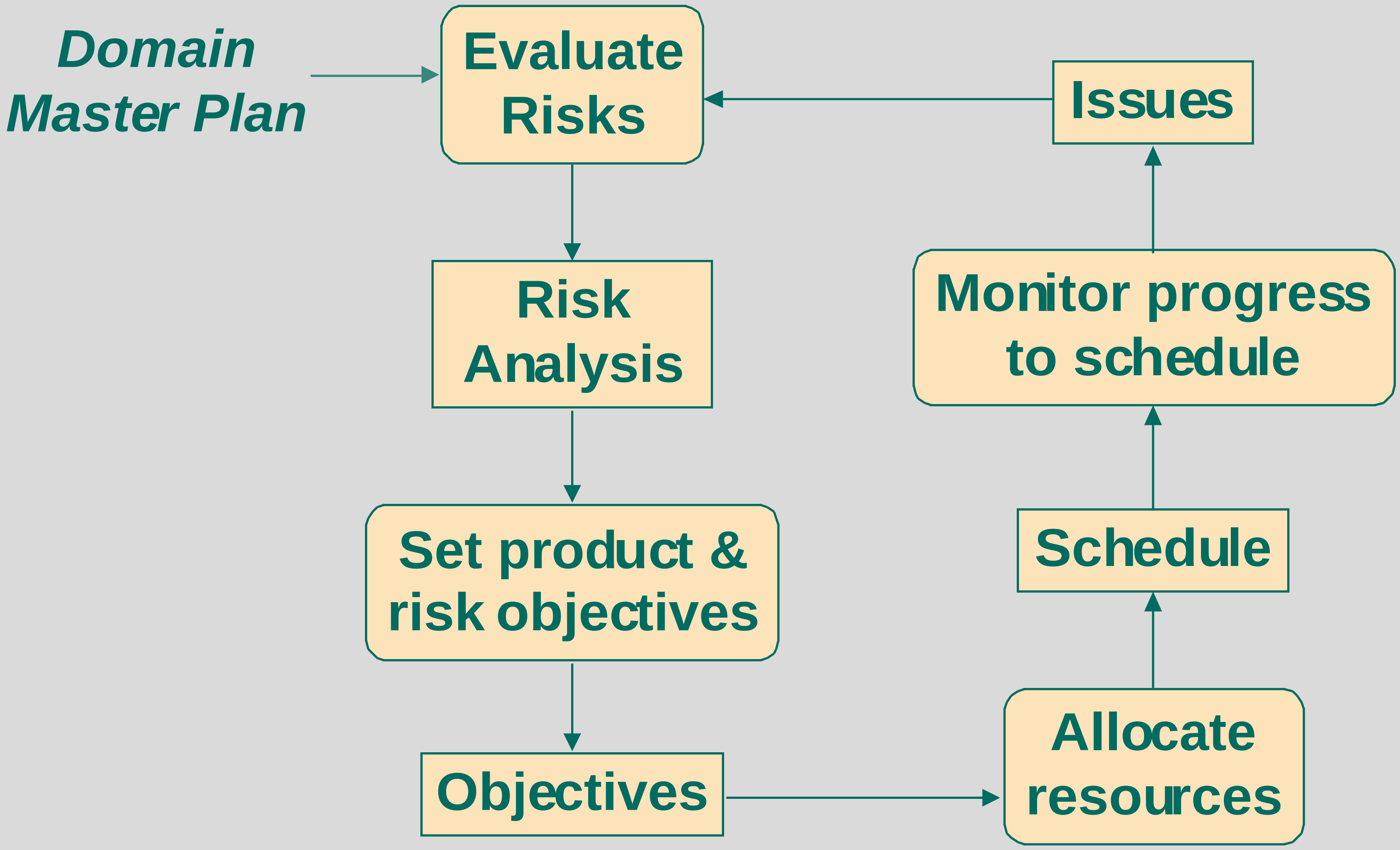
Domain Increment Plan

A plan for completing an increment of the domain master plan

- **Analyze risks (potential threats and mitigation in progressing toward allocated domain objectives)**
- **Define increment objectives consistent with domain objectives and identified risks**
- **Schedule tasks, allocating resources over time to achieve measurable goals consistent with increment objectives**
- **Monitor progress to plan and document deviations in terms of cause, implications, and disposition**

Domain Management

Managing a Domain Increment



A Notional Example Domain

Product category: List management

Examples

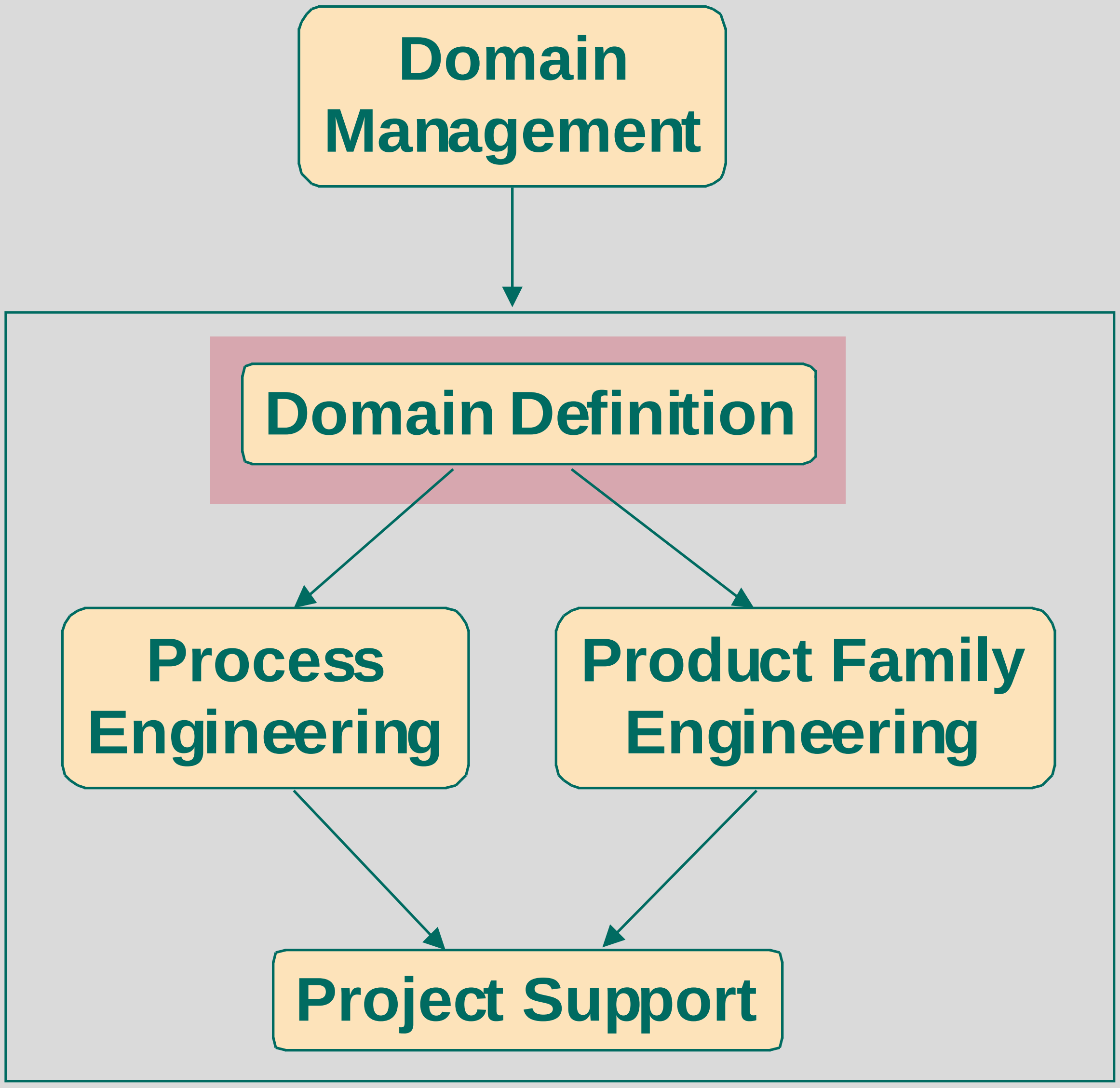
- Small business or personal expense records
- Publication references
- Task assignments

Business objective: Rapid, low-cost provision of a diverse product line of simple list management products

Goals

- Ability to customize product features over a range of options suited to likely uses by an individual
- Less than two hours of effort to produce a defect-free customized product

Domain Engineering Process



Domain Definition

Context: **Domain Plan**

Participants:

- **2-3 management and marketing/sales representatives**
- **3-5 technical managers and lead engineers**

Needed Expertise:

- **Business strategy and market trends**
- **Customers' needs**
- **Current and potential solutions**

Responsibilities:

- **Refine product line scope and focus**
- **Establish terminology**
- **Characterize defining product similarities**

Domain Definition

Work Product: **Domain Definition**
(Domain synopsis, Legacy products, Domain glossary, Domain assumptions, Decision model)

Domain Definition

Domain Synopsis

An informal statement of domain scope

- Describe any product in the domain
- Start with 1 sentence, expand up to 2 pages
- Use and refine the Domain Glossary
- Reflect a concensus view of domain experts
- Establish minimum criteria for whether a proposed product fits in the domain

Example

Domain Synopsis

The List Management (LM) domain is a family of products for maintaining a list of a designated type of *asset* or *activity* description. Each product is customized to represent information as *attributes* specific to a particular *asset* or *activity* type.

Each LM product is intended for use by an individual in support of the need to track and manage the status of corresponding *assets* or *activities*. The LM domain provides the capability to construct an LM product customized to the *attributes* associated with any single *asset* or *activity* type. Managed *assets/activities* may be further classified based on the values of *key attributes* that enable/disable the inclusion of other *informational attributes*.

Each LM product will provide mechanisms for adding and removing *asset/activity* instances and for modifying associated *attributes*. An LM product will present *asset/activity* lists in an order appropriate to the type of *asset* or *activity*.

Domain Definition
Legacy Products

Existing products that exemplify important features of the domain as a product family

- **Analyze to understand the concepts, content, and form and structure of products within the domain**
- **Use work products as a source of raw material from which to construct domain work products**
- **Derive or verify relevant information through the focused application of reverse engineering techniques**

Domain Definition

Domain Glossary

Definitions and references for concepts and terminology having significance for domain experts

- Identify significant terms and phrases
- Note and clarify any redundant or ambiguous terms
- Establish a consistent set of terms as preferred standard terminology
- Correlate alternatives (of other organizations) to preferred terminology
- Characterize aggregate concepts and connections among terms
- Provide references to primary and secondary source materials

Example

Domain Glossary

Asset - A physical object whose instances are tracked with an LM product.

Activity - A time-spanning effort whose instances are tracked with an LM product.

Attribute – A data value associated with an *asset* or *activity*.

Description - A set of *key* and *informational attributes* that are sufficient to identify and determine the status of an asset/activity.

Informational attribute – An *attribute* that provides additional information about an *asset* or *activity*.

Key attribute – An *attribute* that can be used in unambiguously determining the category or identity of an *asset* or *activity*.

Domain Definition

Domain Assumptions

Defining assumptions that describe and justify how products are similar

- **Consider how products – past, current, and future – are alike (*commonality assumptions*), emphasizing observable features**
- **Analyse each commonality assumption to discover differences among products (*variability assumptions*)**
- **Analyse product subsets defined by each variability assumption to discover additional commonality assumptions**
- **Identify deferred, excluded, and unsupported features as commonalities**
- **Document each identified assumption with an informal description and justification**

Types of Domain Assumptions

Commonality: A way in which a set of products are alike

Commonality Assumption:

A feature common to a set of similar products (possibly a subset of a larger family)

Variability: A way in which a set of similar products differ

Variability Assumption:

A feature that differs within a set of similar products (characterizing a subfamily of products that are similar according to this assumption)

Exclusion: A feature that characterizes products which would otherwise be in the domain but are outside due to this (a commonality associated with a particular variability option)

Sample Commonality Assumptions

For cellular telephones,

- operation is manual via a keypad. (justif: low cost, responsive)
- status information is displayed in a text panel. (justif: flexible, adequate for low bandwidth)
- if sold in the U.S., 220 volt chargers are not offered. (justif: use outside U.S. is precluded by comm. system incompatibilities)

For automobiles,

- there is onboard fuel storage. (justif: must be indep. mobile)
- there is an engine which converts fuel to mechanical and electrical energy. (justif: different forms better for each)
- with gasoline and diesel powered engines, a battery supplies electrical power when the engine is off. (justif: need alternative to running engine for low power)
- if sold in Finland, air conditioners are not offered. (justif: no demand to justify logistics)

Sample Variability Assumptions

For cellular telephones

- **displayed text is in one of several different languages (justif: the targeted market is international and multilingual)**
- **if sold in the U.S., text can be preset to display in either English or Spanish (justif: only these languages offer a sufficient market opportunity in the U.S.)**

For automobiles

- **engines are powered by either gasoline, diesel, or electricity (justif: these are the only fuels today that are sufficiently efficient, distributable, transportable, and safe)**
- **for electicity-powered engines, electricity is stored in batteries or fuel cells (justif: use-on-demand requires generation when convenient with storage in one of these media for later use)**

Example

Domain Assumptions

C: An LM product concerns a single specific type of asset or activity.

C: An LM product identifies specific assets/activities with a set of key attributes.

C: An LM product provides a mechanism for the addition of asset/activity instances.

C: An LM product displays asset/activity instances in a predictable order.

C: An LM product provides a mechanism for setting the value of each attribute of an asset/activity instance.

C: LM products do not support concurrent access by multiple users to a particular set of asset/activity instances.

C: LM products do not support multi-valued attributes.

V: Attributes differ according to the type of asset/activity managed.

V: A date attribute can be restricted to past or future dates.

Domain Definition
Decision Model

A set of decisions that are sufficient to distinguish among the members of a product family and identify a particular product in the domain

– For each variability assumption, specify one or more decisions by name and value type

– Aggregate and name sets of related decisions

– Identify dependencies and constraints on decisions that arise when other decisions are resolved

– Evaluate completeness by adequately describing different products uniquely with decisions

– Add other questions which provide any information that a knowledgeable engineer would need to build a complete product

Example

Decision Model

Title: String

Usage: (planning, recordkeeping)?

Keys: (name:String, type:(Text, Date)?)* {V1}

Attributes: (name:String, type:(Text, Date:(past?, future?), Num, Money)?)* {V1}

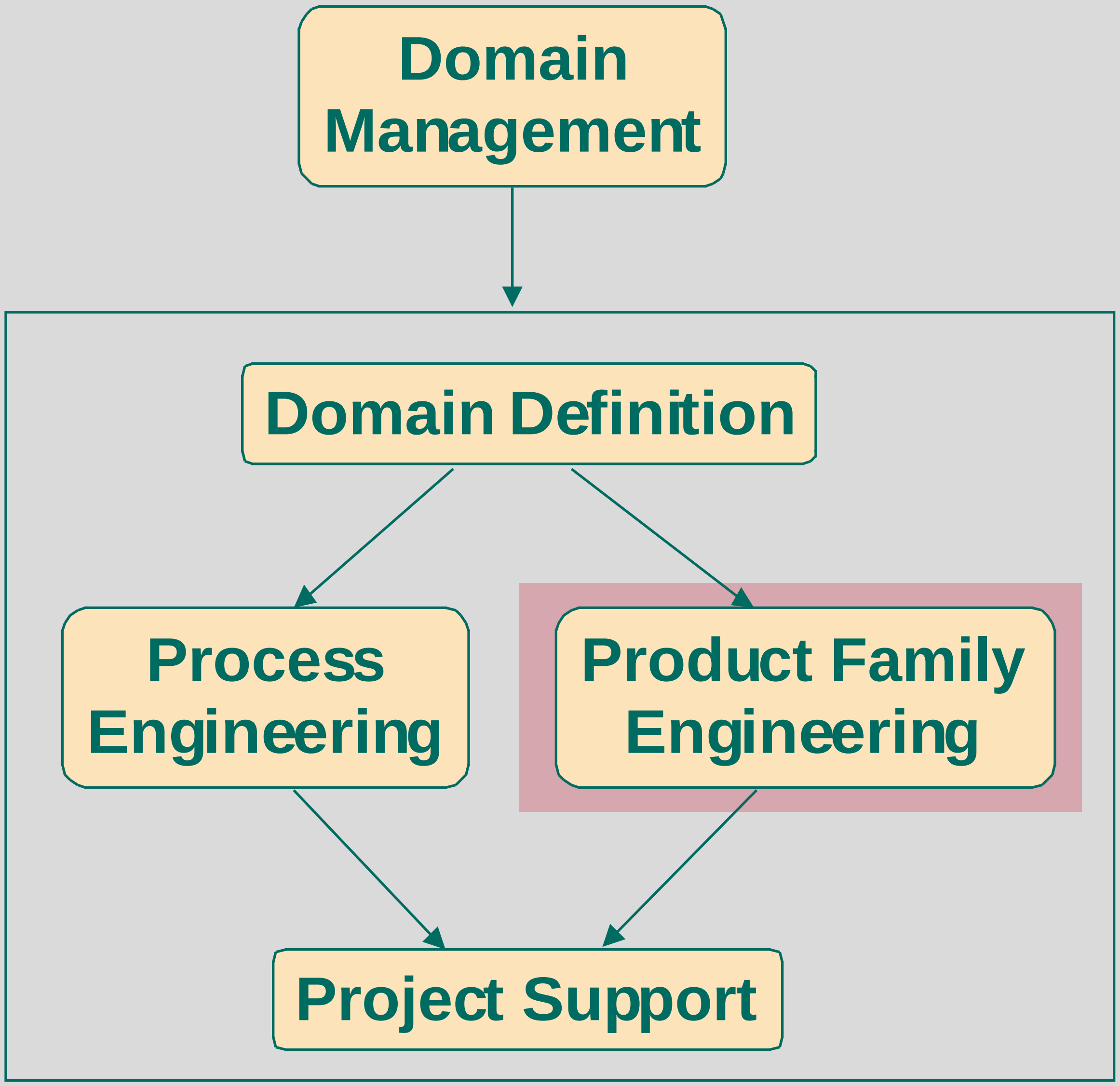
Order: String* {names of Key attributes} { not impl. }

ItemsRecur? {some items are variants of other items}

Notation

*	*repetition (multiple values)*	String	*a simple text value item*
?	*optional or a choice*	(. . .)	*a set of value items*
{ . . . }	*a comment*		

Domain Engineering Process



Product Family Engineering

Context: Domain Definition

Participants: Technical managers and engineers

Needed Expertise:

- Customer problems
- Solution techniques

Responsibilities:

- Specify the problems that Application Products solve
- Define the structure and composition of Products
- Construct components and mechanisms to be used in creating Products

Work Product: Product Family
(Requirements, Design, Implementation)

Definitions

- Method - Guidance and criteria that prescribe a systematic, repeatable technique for performing an activity
- Method suite - A set of methods that provide a consistent approach for performing a set of related activities
- Model - A representation of specific aspects of a system that permit answering particular questions
- Work product - Any tangible artifact resulting from an activity

Product Family

A representation of a set of similar products from which an individual product can be extracted by resolving associated decisions

- Use domain assumptions as a guide to family content
- Analyze legacy products:
  - to improve domain understanding or answer questions
  - to acquire raw materials as a basis for domain work products
- Create a product family by applying methods for creating a product:
  - Requirements: What does the product do?
  - Design: How is the product structured?
  - Implementation: How does the product work?
  - Verification: Does the product work correctly?
- At any point of uncertainty, changeability, or diversity, annotate for decision-controlled tailoring

Examples of Method Suites

Methods for creating a product

- Requirements: what does the product do?
- Design: how is the product structured?
- Implementation: how does the product work?
- Verification: does the product work correctly?

Examples of method suites

- Real-Time Structured Analysis and Design (RTSA/D)
- Software Cost Reduction (SCR)
- Concurrent Design Approach for Real-Time Systems (CODARTS)
- Cleanroom
- Objectory

Product Family Requirements

The capabilities and properties that products must exhibit to be accepted as solutions by customers.

- **Develop usage scenarios to characterize customers' problems.**
- **Describe the application:**
  - » **Concept (purpose and objectives)**
  - » **Context (environment of use or operation)**
  - » **Content (externally detectable behavior & information content)**
  - » **Constraints (environmental, performance/reliability, etc.)**
- **Specify each AE work product:**
  - » **Purpose and objectives**
  - » **How used**
  - » **Constraints on form or content**
- **Use annotations to record how decisions representing different needs result in different requirements.**

Example

Product Family Requirements - Application Description

Outputs : <title>Report

Inputs : <title>Data

Functions

Display the active set of <title> items in a tabular format.

Provide an operation to add new <title> items, including the values of its attributes.

The attributes of an <title> item are:

<for i in items <its <i> value

>>

<values-recur ? <Provide an operation to add a new <title> item which is a variant of an existing <title> item.>>

Provide an operation to deactivate an item; deactivated items are omitted from the <title> display.

Product Family Design

A design for products that satisfactorily resolves all conflicts among needs and constraints.

- – Specify the architecture (internal organization) of applications
  - » Identify needed components (static structure)
  - » Specify runtime tasking and communications (dynamic structure)
  - » Specify how components interact (dependency structure)
- – Create annotated outlines for document and testing work products
- – Specify the interface and adaptability of components identified in the architecture
- – Use annotations to specify how decisions control tailoring of the architecture and components

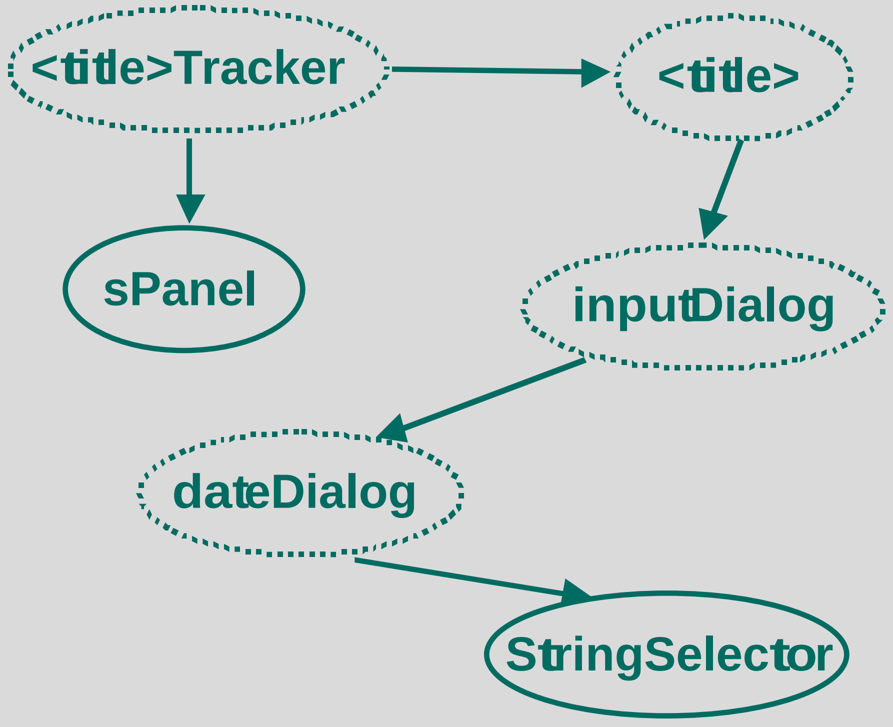
Example

Product Family Design : Architecture

Class Set

<title>Tracker : Frame
<title>
sPanel : Panel
inputDialog : Dialog
dateDialog : Dialog
StringSelector

Dependency Structure



Example

Product Family Design : Component Interface

Component family name: dateDialog

Variability specification:

Range : (past?, future?) { indicates whether dates are restricted }

Interface specification:

***Name*: <Range.past?<past>;<Range.future?<future>;...>DateDialog**

***Types*: Date { internal representation of a date }**

***Programs*:**

UserSelect (Date):Date {allow a user to specify or change a Date}

Today ():Date {get today's date}

ToString (Date):String {convert a Date into a displayable form}

SameDay (Date, Date):Boolean {test whether 2 dates are the same}

EarlierDay (Date, Date):Date {get the earlier of 2 dates}

Example

Product Family Design : Design Mapping

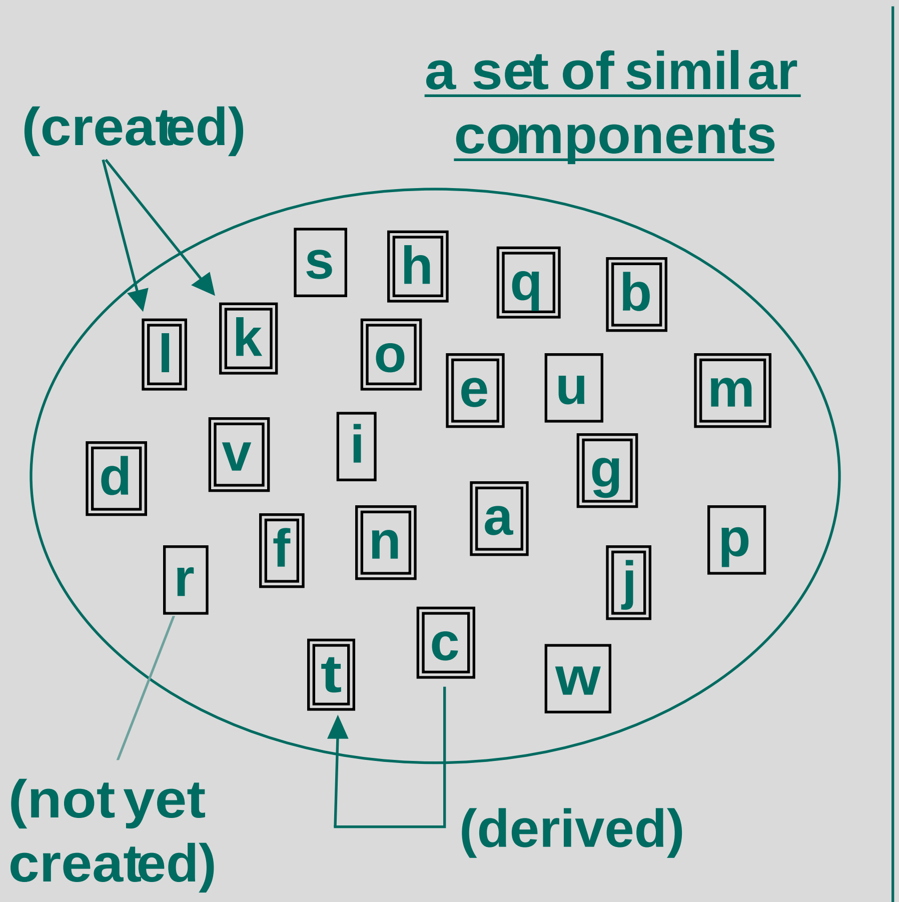
- » **Architecture mapping: how to adapt the Product Family Architecture**
- » **Component mapping: How to select Adaptable Components to instantiate the Product Family Architecture**
- » **Decision mapping: How to determine the values of Adaptable Components' parameters**

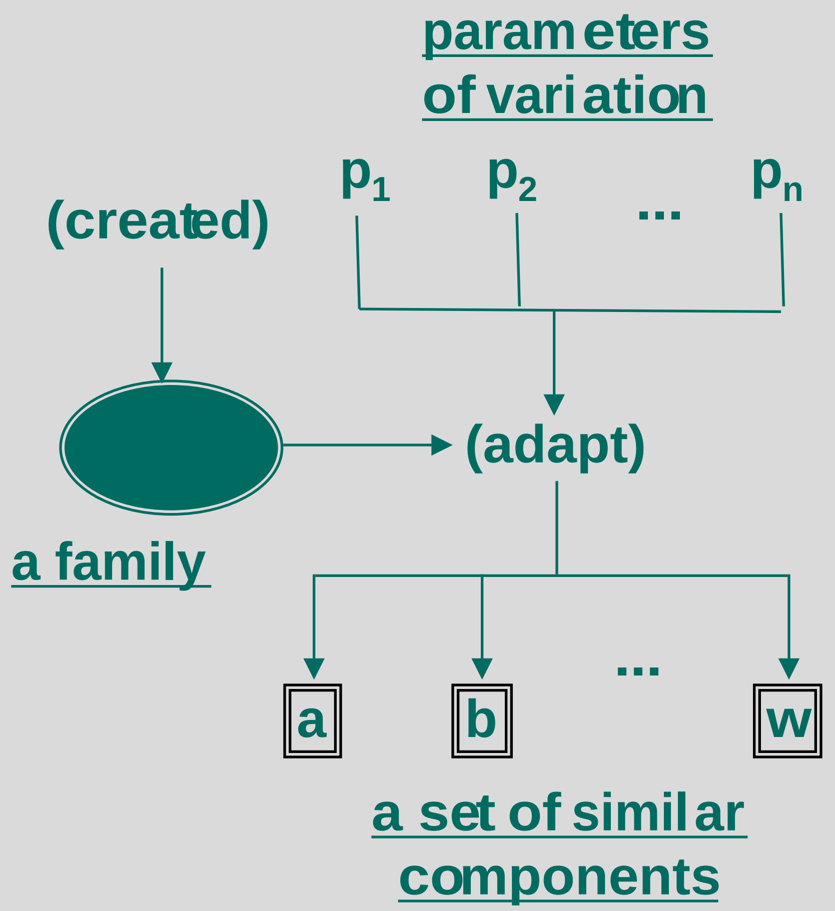
Product Family Implementation

A set of components and composition procedures that are sufficient to enable deriving any describable product.

- Implement component families as specified by the product family design
  - » Design the internal structure of an Adaptable Component conforming to a component design
  - » Implement the Adaptable Component
  - » Verify the Adaptable Component to the component design
- Fill in annotated AE document work product outlines
- Develop outlined AE testing work products
- Document or implement procedures to generate an application and associated AE work products by selecting, instantiating, and composing Adaptable Components

Two Views of a Component Family





Example

Product Family Implementation: Adaptable Component (CCOS order sheet)

```
<{customized computer order specification system}>
<program  ccos (
            platform:(handheld, portable, desktop, server:(files, calc, comm)?)?,
            components:(disks:(primary:text, secondary?:text, removable?:(mega, giga)?),
                        printer?:(inkjet, laser)?),
            customer:(name:text, street:text, city_state_zip:text, telephone?:text),
            xactn:(agent:text, price:text, discount:text, date:text)
            ) <
```

The following order has been authorized for submission to the ACME Customized Computer Order Specification System in keeping with instruction 98-38A76-B, Guidance to Regional Sales Agents for Customized Product Orders. The customer, <customer.name>, hereafter identified as "Customer", accepts that delivery is contingent on availability of requisite constituent and added components and that the order cannot be modified without the mutual consent of ACME Computers Limited subsequent to transmission of this specification to the ACME Manufacturing

Example

Product Family Implementation: Generation Procedure

```
String temp;
temp = platform.getSelectedItem ();
resultText += "platform:("
   + (temp.equals ("Handheld") ? "handheld"
       : temp.equals ("Notebook") ? "portable"
       : temp.equals ("Desktop") ? "desktop"
       : temp.equals ("Server") ? "server" : "");
if (temp.equals ("Server")) {
  temp = svrtyp.getSelectedItem ();
  resultText += ":("
        + (temp.equals ("File Storage") ? "files"
              : temp.equals ("Computation") ? "calc"
              : temp.equals ("Communications") ? "comm" : "")
        + ")";
  }
resultText += ")," + eol;
resultText += "       components:(disks:(primary:" + LEsc +
```

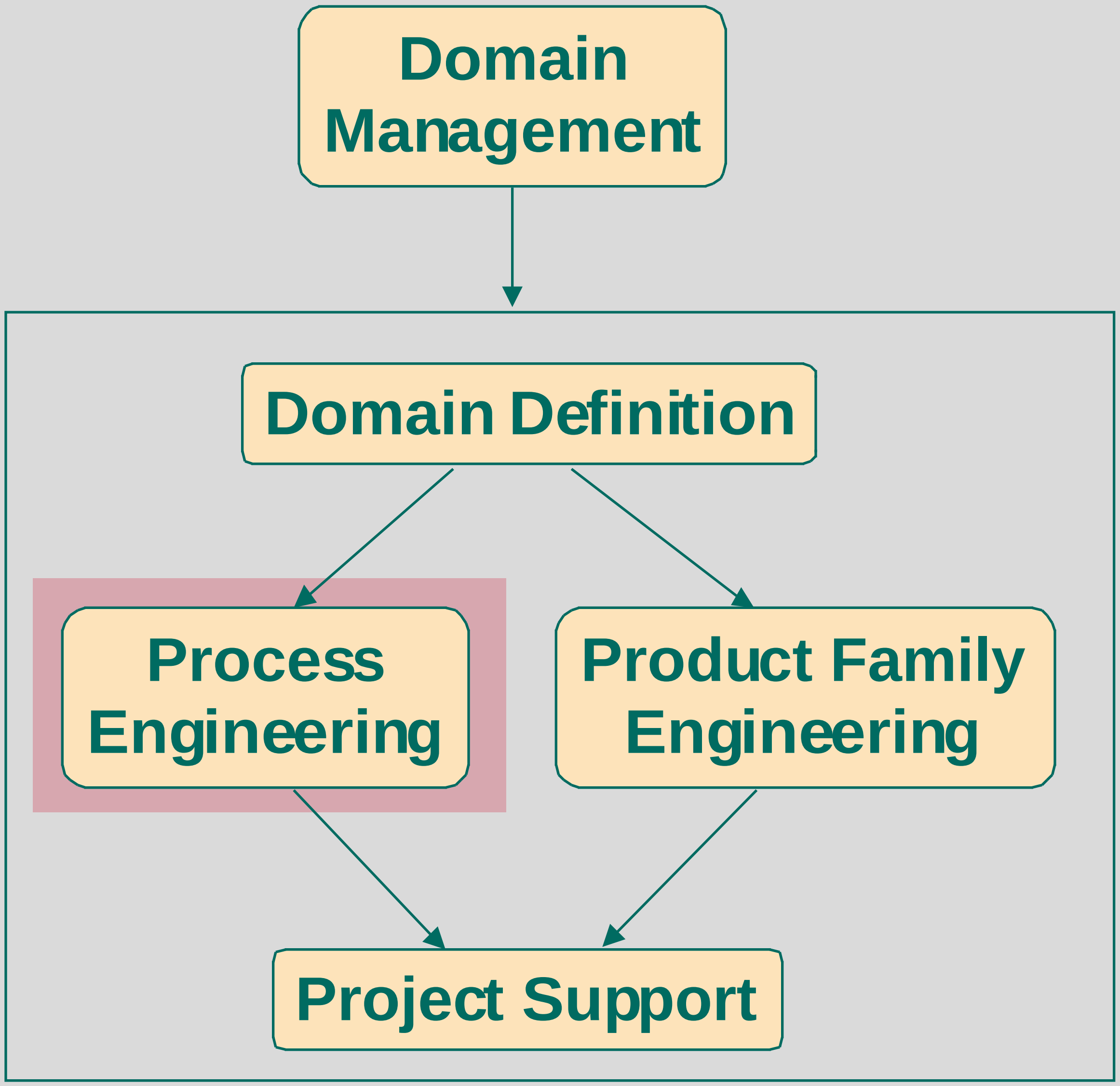
Example

Product Family Implementation: Generation Procedure

(Generated MetaProgram Instantiator)

```
«»
«include "pattern"
»« ccos (
   platform:(handheld),
   components:(disks:(primary:«2.3», )),
   customer:(name:«», street:«», city_state_zip:«»),
   xactn:(agent:«», price:«0», discount:«100»,
      date:«January 1, 1998»)
)»
```

Domain Engineering Process



Process Engineering

Context: Domain Definition, Product Family

Needed Expertise:

Analyzing, designing, and documenting engineering processes and procedures

Building and documenting software tools

Responsibility: Define a standardized Application Engineering process for the domain

Work Product: AE Process (Requirements, Infrastructure)

AE Process
Requirements

A specification for how application engineering projects should operate, given domain capabilities

– Define the activities, work products, and work flow for an Application Engineering process.

» Document the current AE process

» Identify externally imposed work products

» Define a revised AE process by combining, eliminating, or simplifying steps through reuse or tool support

– Define a dialog for creating an application model to describe a needed product.

» Define forms for specifying related sets of decisions

» Order form access based on specified Decision Model constraints

Example

Process Requirements

Current process: waterfall with informal requirements and design, instance coding, and ad hoc testing

Improvements:

- Generate family-based requirements and design documents
- Generate family-based instance code
- Generate family-based test suite for first level testing

Steps:

- Specify a decision-based application model
- Generate instance documents, code, and tests
- Evaluate generated work products for needed application model changes
- Apply minimal handtailoring to achieve customer acceptance
- Baseline the application model and required hand changes
- Report short-falls to Domain Engineering

AE Process
Infrastructure

Documentation and tools for the AE Process as specified in AE Process Requirements

- Document the AE Process in the form of organizational policies and procedures
- Create a detailed domain user's guide for application engineers to follow in using the domain
- Develop automated support for the process
- Create training courses for instructing application engineers in proper domain practices
- Provide materials (tools and manuals) for deploying and supporting AE Projects in use of the domain

A Minimal Application Engineering Environment

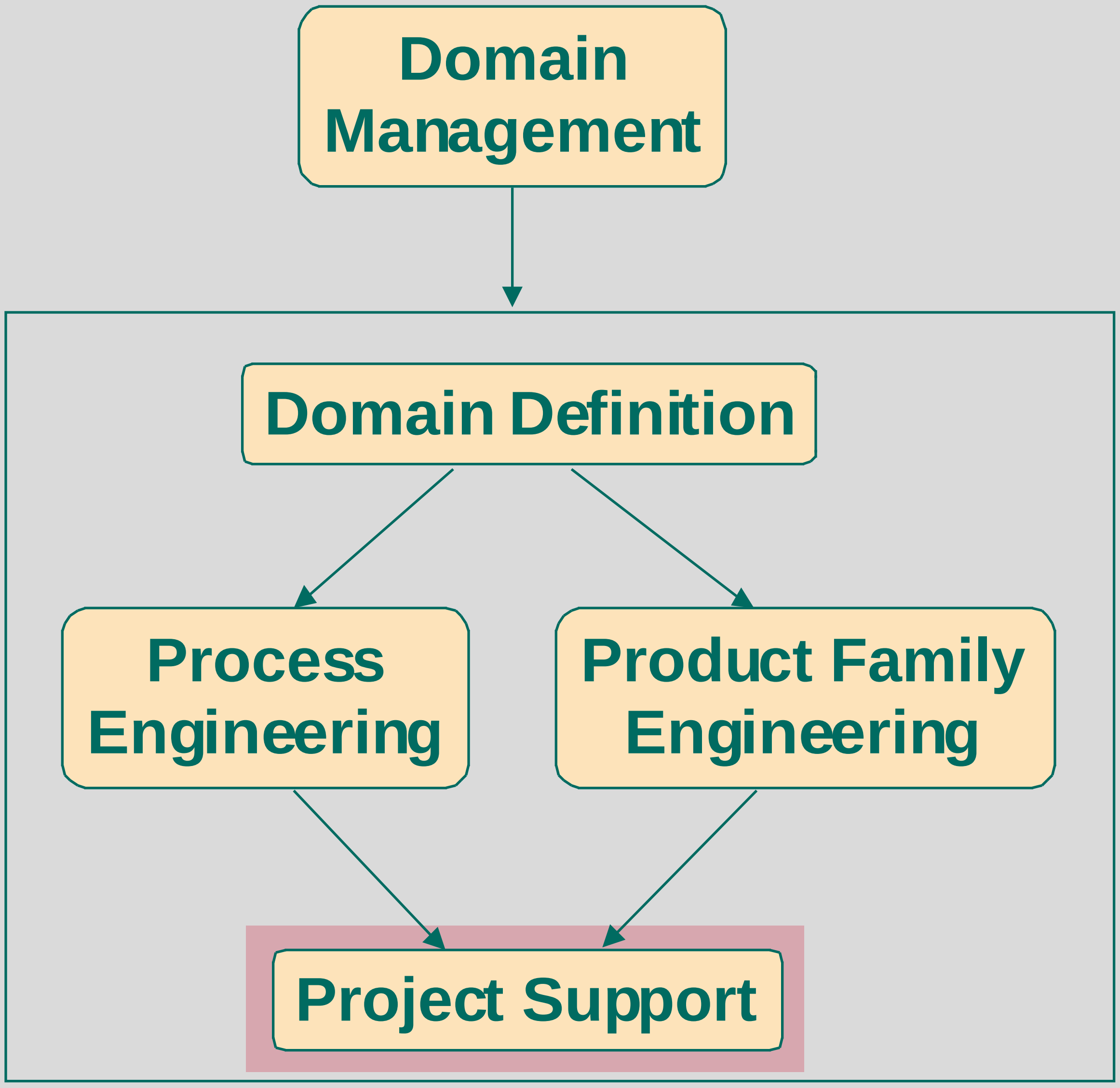
- A tool for interactive dialogs (to create, store, view, and update Application Models)
- File-based storage for Adaptable Components
- A tool to instantiate work product component families (Adaptable Components) using an Application Model
- A tool to compose (e.g., compile and link) instantiated components into deliverable work products

Example

AE Process : Application Engineering Environment

- A domain-specific (Java-implemented) tool for application model definition
- MTP plugin for Adaptable Component instantiation
- MTP-instrumented text and Java code files representing Adaptable Components for the domain
- CodeWarrior Integrated Development Environment to select, instantiate, and compose (or compile and link) Adaptable Components into work products

Domain Engineering Process



Project Support

Context: **Domain Definition, Product Family, AE Process**

Needed Expertise:

- **Verification and validation techniques**
- **Training delivery**
- **Customer support**

Responsibility: **Ensure that the domain is effective both for the business and for client AE projects**
(Domain Verification, Domain Validation, Domain Delivery)

Project Support

Domain Verification

An independent evaluation of the consistency, completeness, correctness, and quality of Domain Engineering work products.

- **Verify adherence to management-prescribed standards.**
- **Verify the Domain Definition to domain objectives.**
- **Verify consistency of Product Family Engineering work products**
- **Verify consistency of Process Engineering work products.**
- **Verify Product Family/AE Process compatibility and compliance with the Domain Definition.**
- **Document problems for DE correction.**

Project Support
Domain Validation

An independent evaluation of the effectiveness and utility of the integrated Domain product.

– Evaluate satisfaction of business objectives.

– Evaluate the quality and effectiveness of the Domain from the perspective of current and near-future AE projects.

– Use the Domain to build test applications that require new, improved, or corrected Domain capabilities.

– Use the Domain to build a set of standard applications to evaluate derived attributes (such as performance, reliability, safety, availability, usability).

– Document problems for DE correction.

Project Support
Domain Delivery

Assistance to AE projects to enable effective use of Domain capabilities to best meet project objectives.

- Transport and install the validated Domain for each AE project
- Assist AE projects in the understanding and proper use of the specified AE process and associated Domain capabilities.
- Document needed Domain improvements and evolution based on current AE project experiences and expected future customer and project needs.