

# **A Domain Analysis and Modeling Method: Application to NASA's Payload Operations Control Center Domain**

Hassan Gomaa

Department of Information and Software Systems Engineering  
George Mason University  
Fairfax, Virginia 22030-4444  
USA

Phone: (703) 993-1652  
Email: hgomaa@isse.gmu.edu

Presented at  
Ground Systems Architectures Workshop

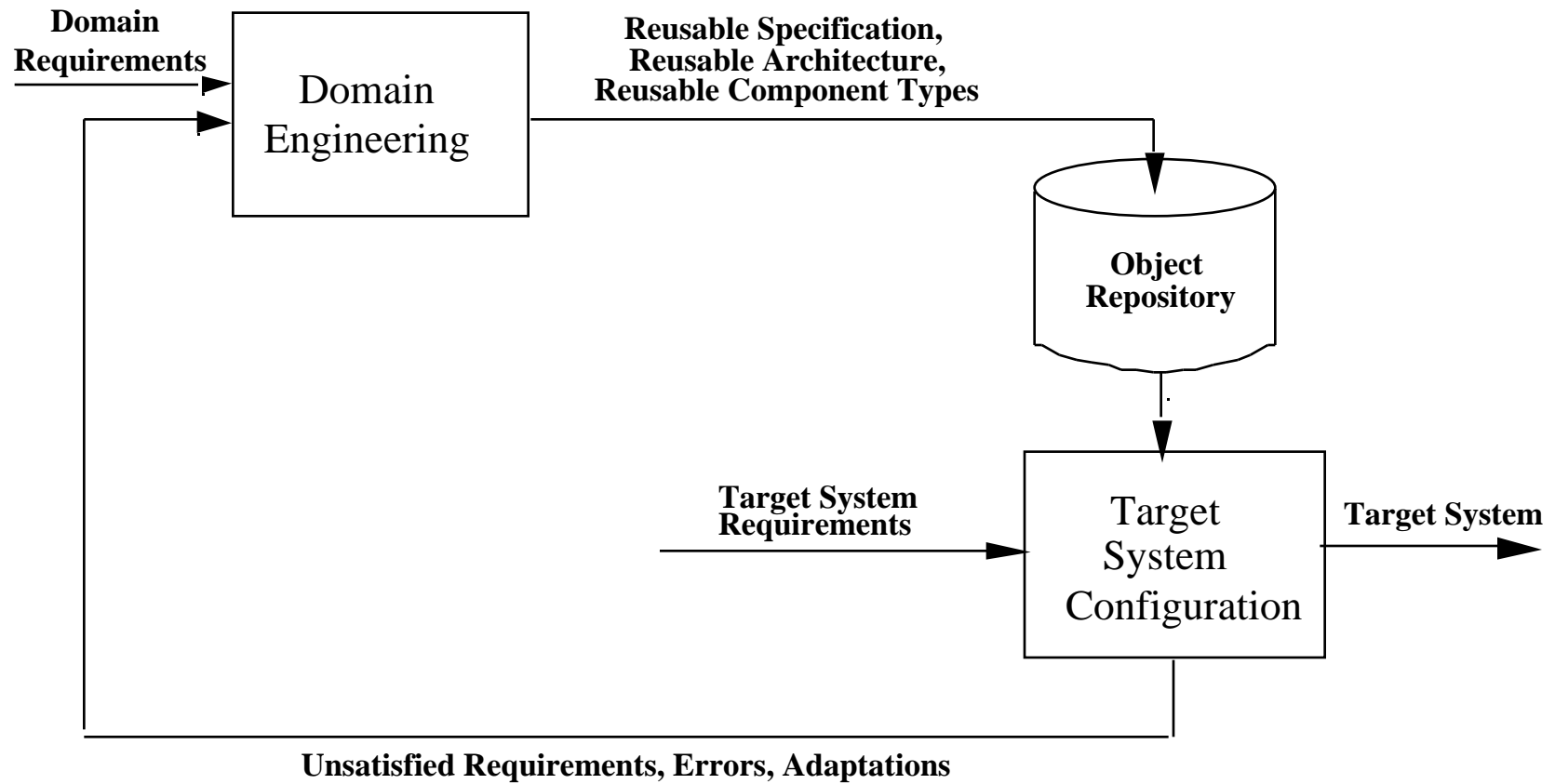
February 1997

**Copyright 1997 Hassan Gomaa**

# Key Concepts

- Evolutionary Domain Life Cycle Model
  - Does not distinguish between Development & Maintenance
  - Life Cycle for FAMILY of Systems
- Domain Engineering
  - Software engineering of family of systems
    - Reusable specifications and architectures
  - Target system configured from reusable architecture
- Prototype Domain Engineering Environment
  - Application domain independent tools
  - Configure distributed applications from reusable architecture

# Evolutionary Domain Life Cycle Model



# Domain Modeling

## Application Domain

Represented by a family of systems

## Domain Model

Problem oriented architecture for Application Domain

Reflects common aspects and variations of family of systems

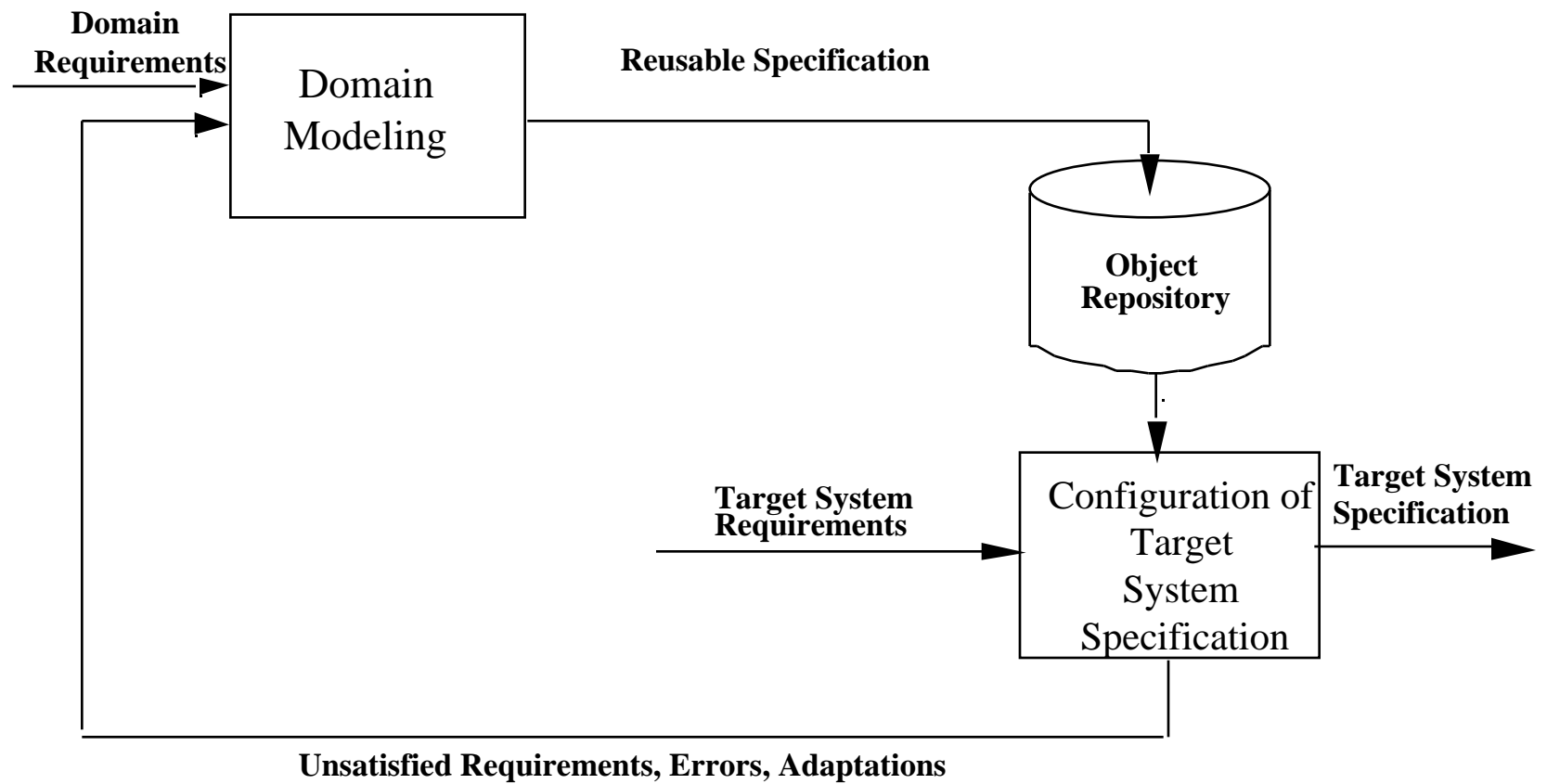
Target system specification created by tailoring domain model

## Domain Modeling Method

Object-oriented multiple viewpoint analysis and modeling method

Models common aspects and variations of family of systems

# Domain Modeling



# Multiple Views of Domain Model

## Aggregation Hierarchy

Decompose complex aggregate object types into simple object types

## Object communication diagrams

Real-world objects modeled as concurrent tasks

Objects communicate via messages

## State transition diagrams

Active object defined by state transition diagram

## Generalization / Specialization Hierarchies

Domain object types may be specialized

## Feature / Object dependencies

Object types & features required to support feature

# How Domain Modeling Method Differs

- Focus on analysis and modeling of family of systems
- Categorization of application domain object types
  - Kernel object types
    - Object types in every member of family
  - Optional objects types
    - Object types in only some members of family
  - Variant objects types
    - Specialized kernel or optional object types
    - Used differently by individual family members
- Feature analysis
  - Emphasis on optional domain requirements
  - Basis for differentiating among members of the family

# Feature / Object Dependencies

Categorization of application domain requirements

- Kernel features

- Optional features

- Prerequisite features

For each optional feature, define

- Optional and/or variant object types required to support feature

- Prerequisite features required to support feature

Feature Relationships

- Mutually exclusive features

- Exactly one of a set of features

- One or more of a set of features

# Feature-Based Scenarios

- Feature based event sequencing scenarios
  - Define how objects interact with each other to support feature
  - Shown on object communication diagram (OCD)
    - Shows message communication between concurrent objects
  - Develop one OCD for each feature
- Determine kernel of Domain Model
  - Kernel feature and kernel object types
- Determine optional features
  - Optional and/or variant object types

# Domain Model for NASA Payload Operations Control Center (POCC) Domain

Multiple views specified

- Object aggregation hierarchy

- Generalization/specialization hierarchies

- Object communication diagrams

- State transition diagrams

Developed feature / object dependencies

- Kernel, variant & optional object types

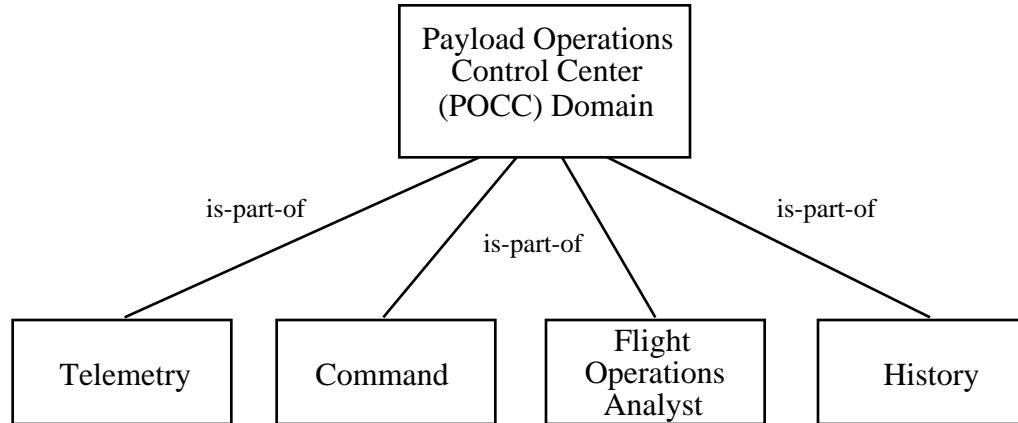
  - required to support each feature

Identified possible variations in POCC domain

- Spacecraft specific

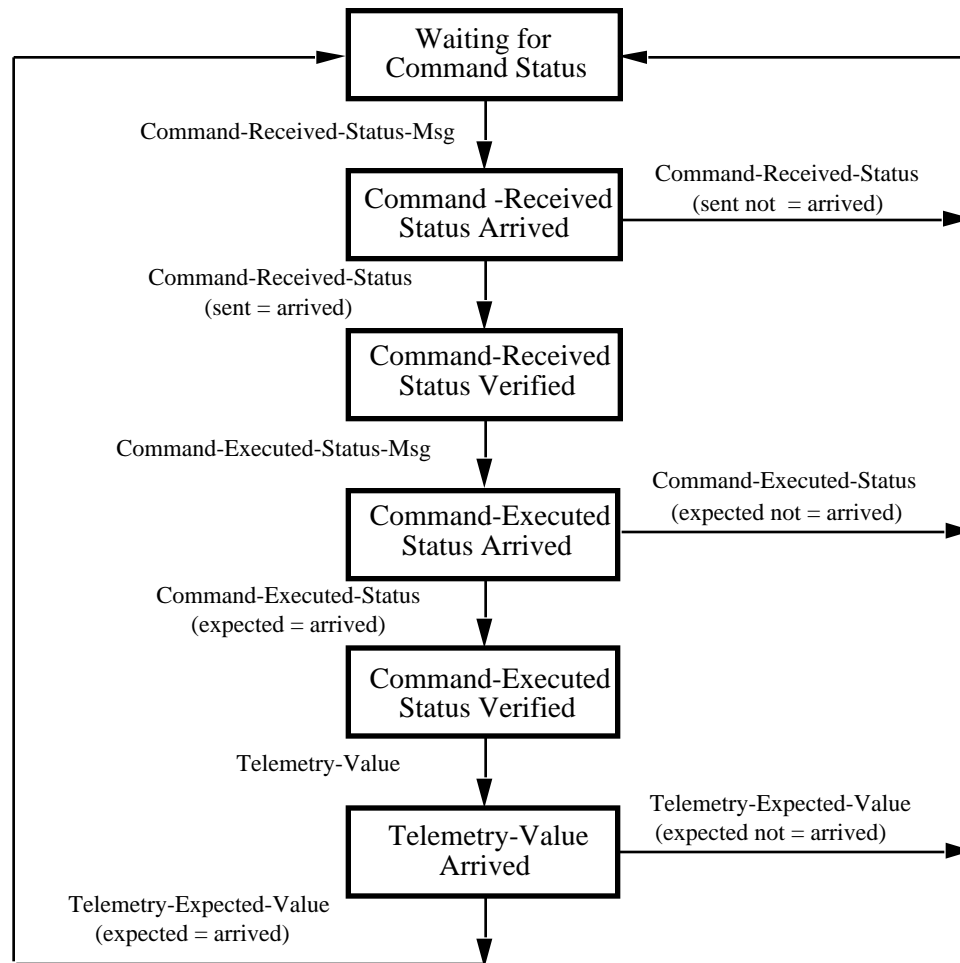
- Observatory experiment specific

# Aggregation Hierarchy from Payload Operations Control Center Domain

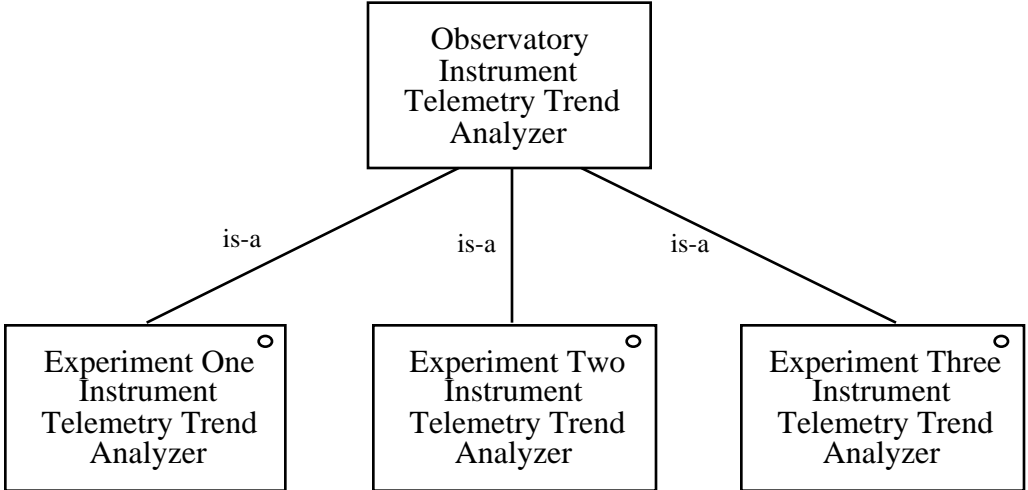




# State Transition Diagram from Payload Operations Control Center Domain



# Generalization/Specialization Hierarchy from Payload Operations Control Center Domain



# Example Feature/Object and Feature/Feature Dependencies from Payload Operations Control Center Domain

## **Example feature-object dependencies:**

- ( Sending Real Time Commands Feature supported-by Real-Time\_Command-Data\_Store\_OS optional )
- ( Sending Real Time Commands Feature supported-by Satellite\_Bound-Command\_Problem\_Resolver\_OS optional )
- ( Sending Real Time Commands Feature supported-by Satellite\_Bound\_Real-Time\_Command\_Processor\_OS optional )

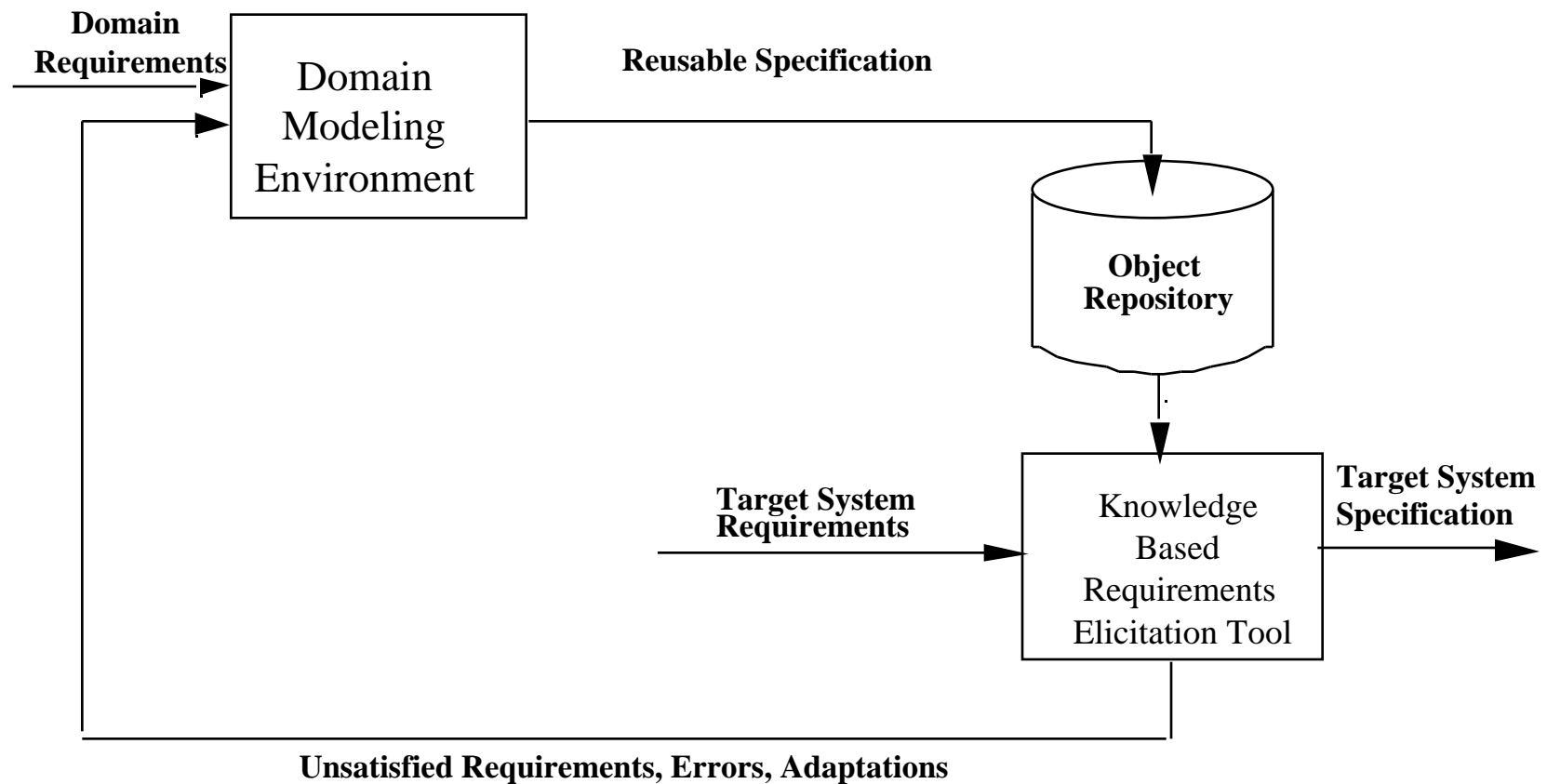
## **Example inter-feature dependency**

- ( Verifying Real Time Commands Feature requires Sending Real Time Commands Feature )

# Prototype Knowledge Based Software Engineering Environment (KBSEE)

- Application domain independent environment
- Domain Modeling Environment
  - Create multiple view graphical representation
  - Check for consistency among multiple views
  - Map views to common underlying representation
  - Store in Object Repository
- Knowledge Based Requirements Elicitation Tool
  - Assists user in selecting consistent set of target system features
  - Configure target system specification from domain model

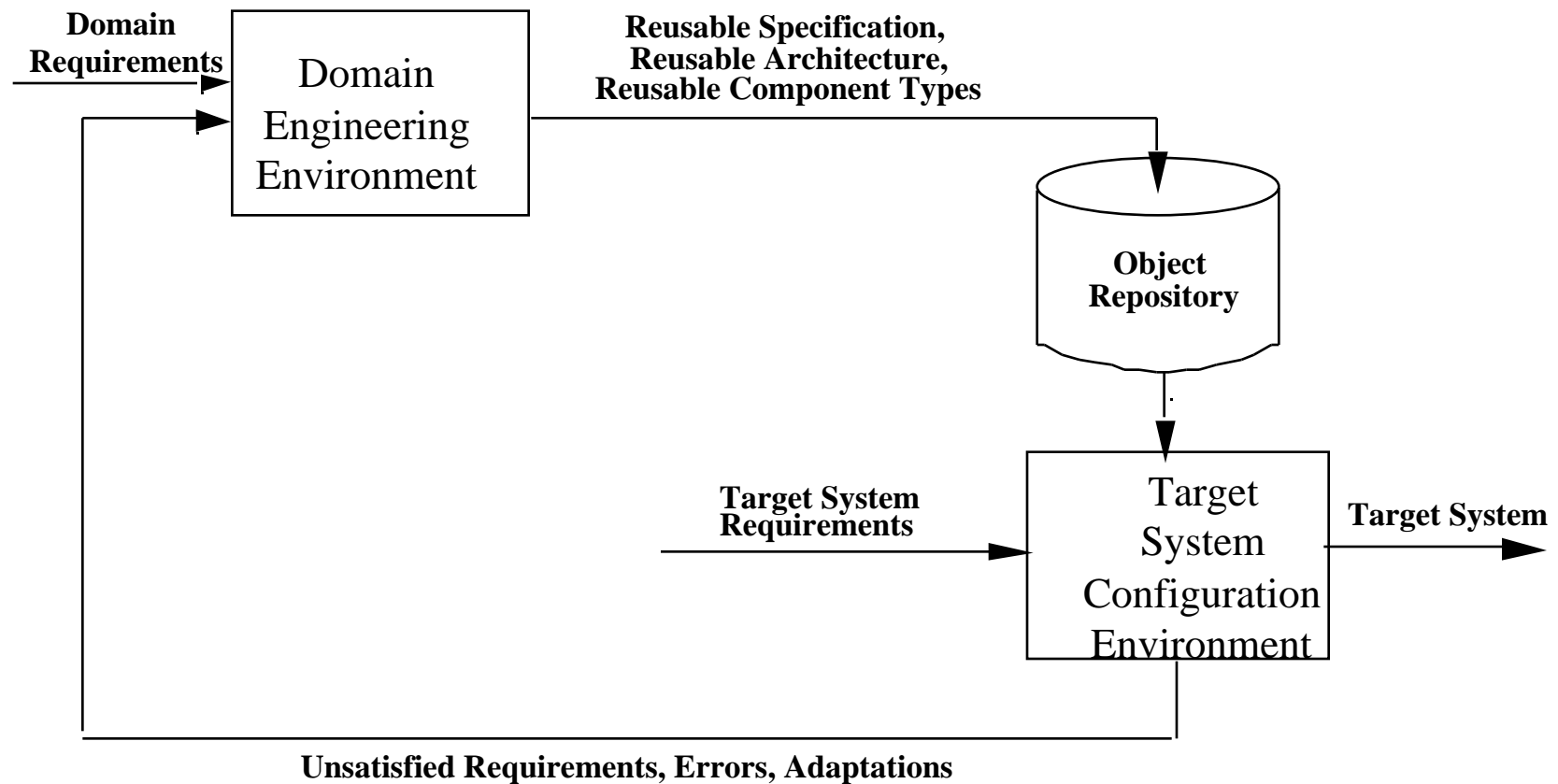
# Prototype Knowledge Based Software Engineering Environment (KBSEE)



# Distributed Application Development Environment

- Support design & configuration of distributed applications
- Integrates domain modeling environment with distributed configuration environment
- Map from domain model to reusable architecture
  - Objects mapped to architecture component types
  - Features mapped to design fragments
    - Interconnected architecture components
- Target system architecture configured from reusable architecture

# Distributed Application Development Environment



# Conclusions

- Domain Engineering
  - Supports reuse of domain specifications and domain architectures
  - Used for several application domains
    - NASA Spacecraft Command and Control
    - Factory Automation
    - Earth Observing System Data and Information System
- Prototype Software Engineering Environment
  - Application domain independent environment
  - Configure executable distributed target systems
    - From domain models and domain architectures
- Current Work / Future Plans
  - Support for entire life cycle
  - Design and Build next generation SEE
  - Apply to other application domains