CSSE 490 Model-Based Software Engineering: Introduction to Domain Engineering

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

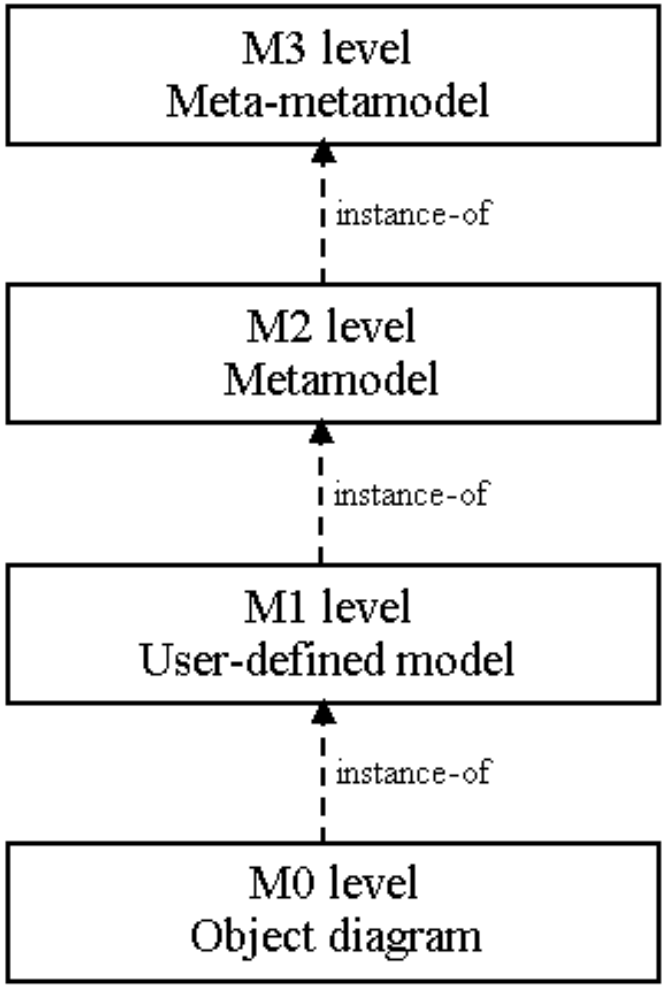
Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Learning Outcomes: Metamodels

Design a metamodel for a model-based software system.

- Exam Discussion
- Milestone 2 Demo
- Looking closer at Mapping
- Introduce Object Constraint Language (OCL)
- Action Semantics (if time)
- Introduce Domain Eng.



Midterm Stats

- Average Score = 85.6
- High Score = 90
- Low Score = 77

- Grade allocations
  - A => 88-100
  - B+ => 85-87
  - B => 79-84
  - C+ => 75-78

Milestone #2

LET'S DEMO!

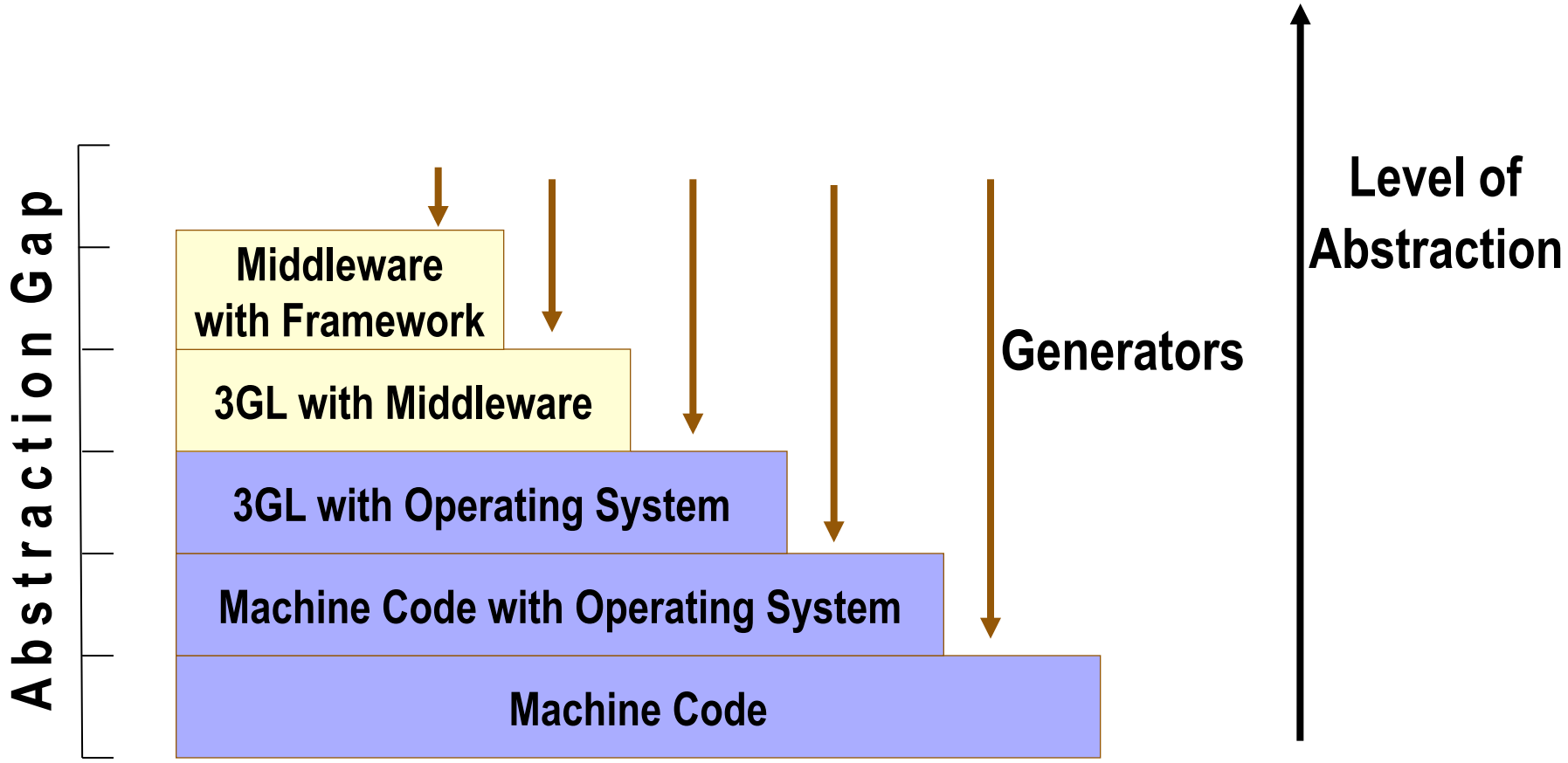
What is the difference between conventional Software Engineering and Model Based Software Engineering?

- Again, think for 15 seconds…
- Let's talk…

Conventional and MBSE

Conventional Software Engineering	Model-Based Engineering
Requirements Analysis Produces requirements for one system	***Domain Analysis*** Produces reusable, configurable requirements for a class of systems
System Design Produces design of one system	***Design*** Produces reusable design for a class of systems and a production plan
System Implementation Produces system implementation	***Implementation*** Produces reusable components, infrastructure and production process

Abstraction Gaps Bain of Mapping



Abstraction or Refinement?

- Mapping techniques between two metamodels often formulate

  1. An **abstraction** (leading to more abstract metamodels) or

  2. A **refinement** (leading to more detailed metamodels)

- Hence, one metamodel is sometimes called an ***abstraction*** or a ***refinement*** of the other
  - When do we call a mapping a refinement?
  - When do we call it an abstraction?

Definitions: Refinement

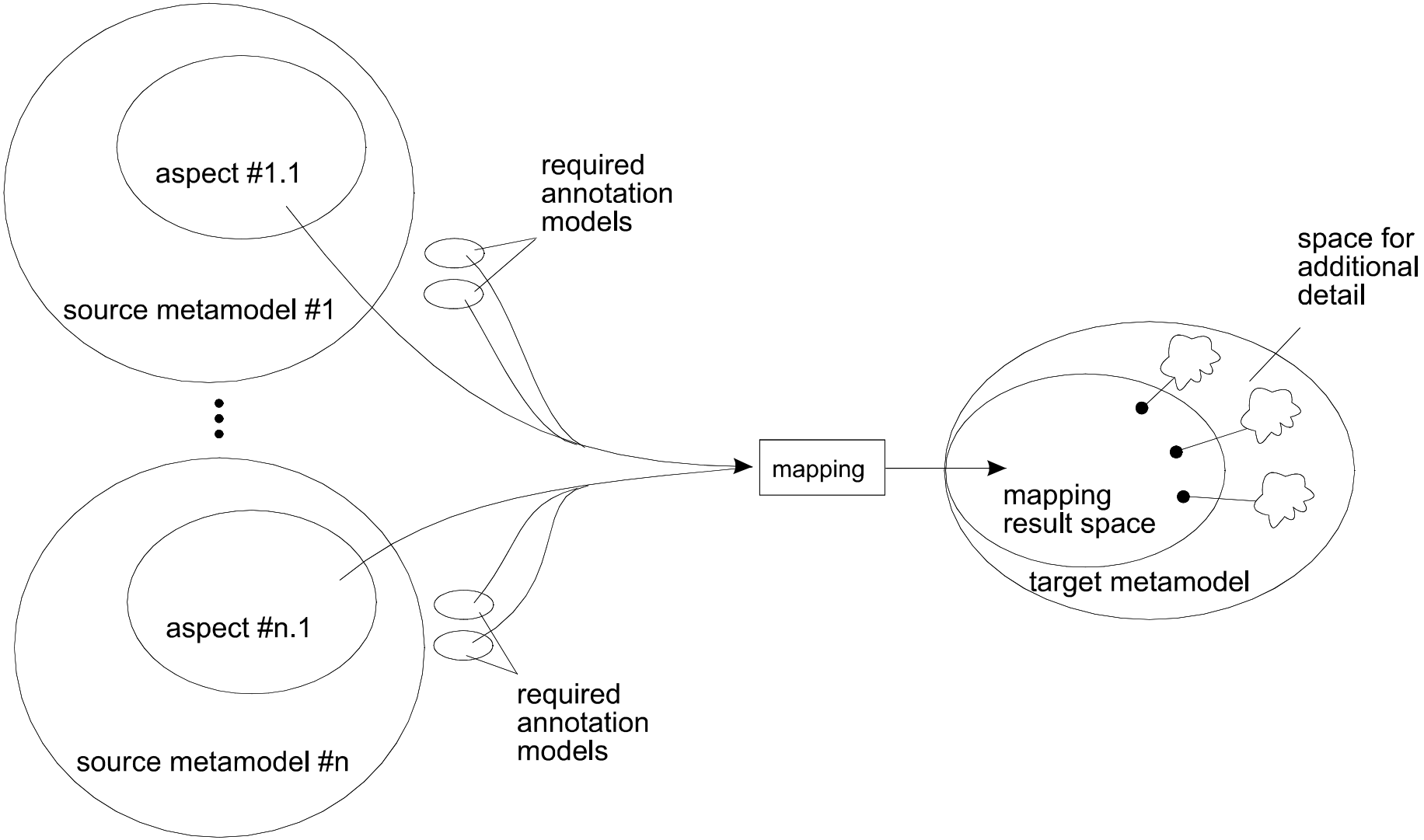
- Let A and B be two metamodels
- B is said to be a refinement of A if

  a "reasonable" (semantic-preserving)

  “surjective” mapping technique
  (or mapping in the algebraic sense)

  from A to B cannot be provided

Refinement Mapping



Definitions: Abstraction

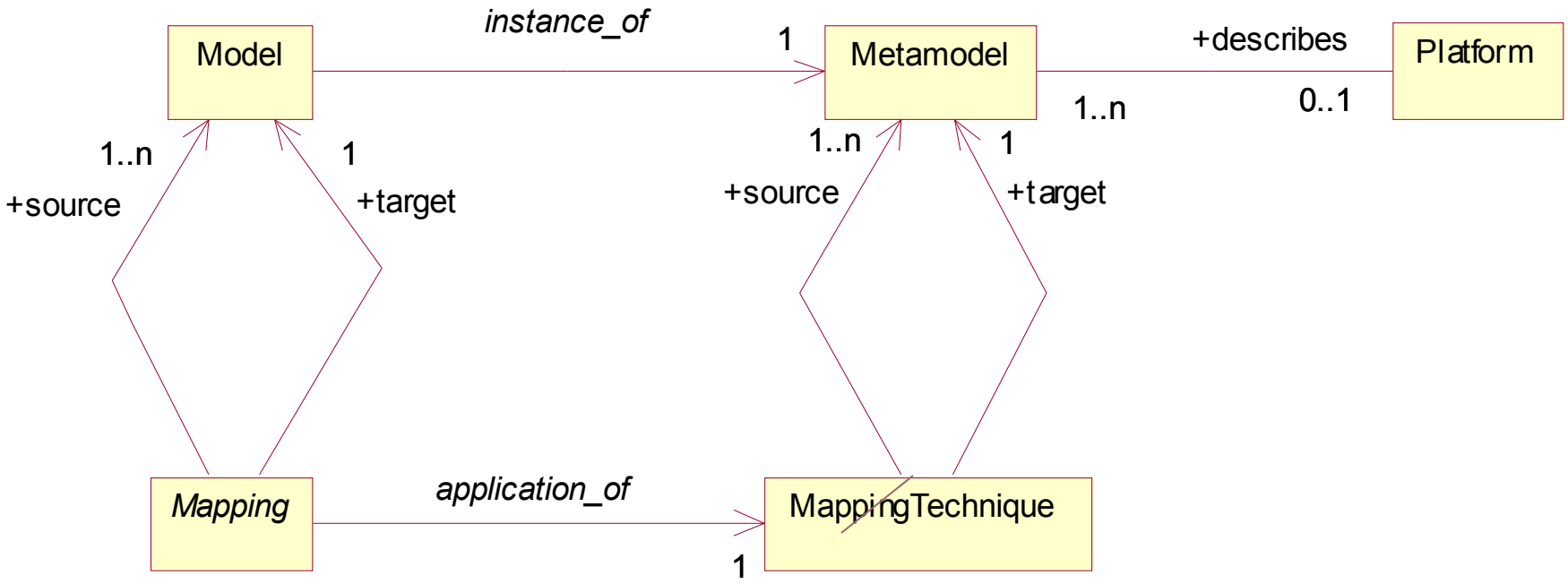
- Let A and B be two metamodels
- B is said to be an **abstraction** of A if

  a "reasonable" (semantic-preserving) surjective and non-injective mapping technique (or mapping in the algebraic sense)

  from A to B can be provided

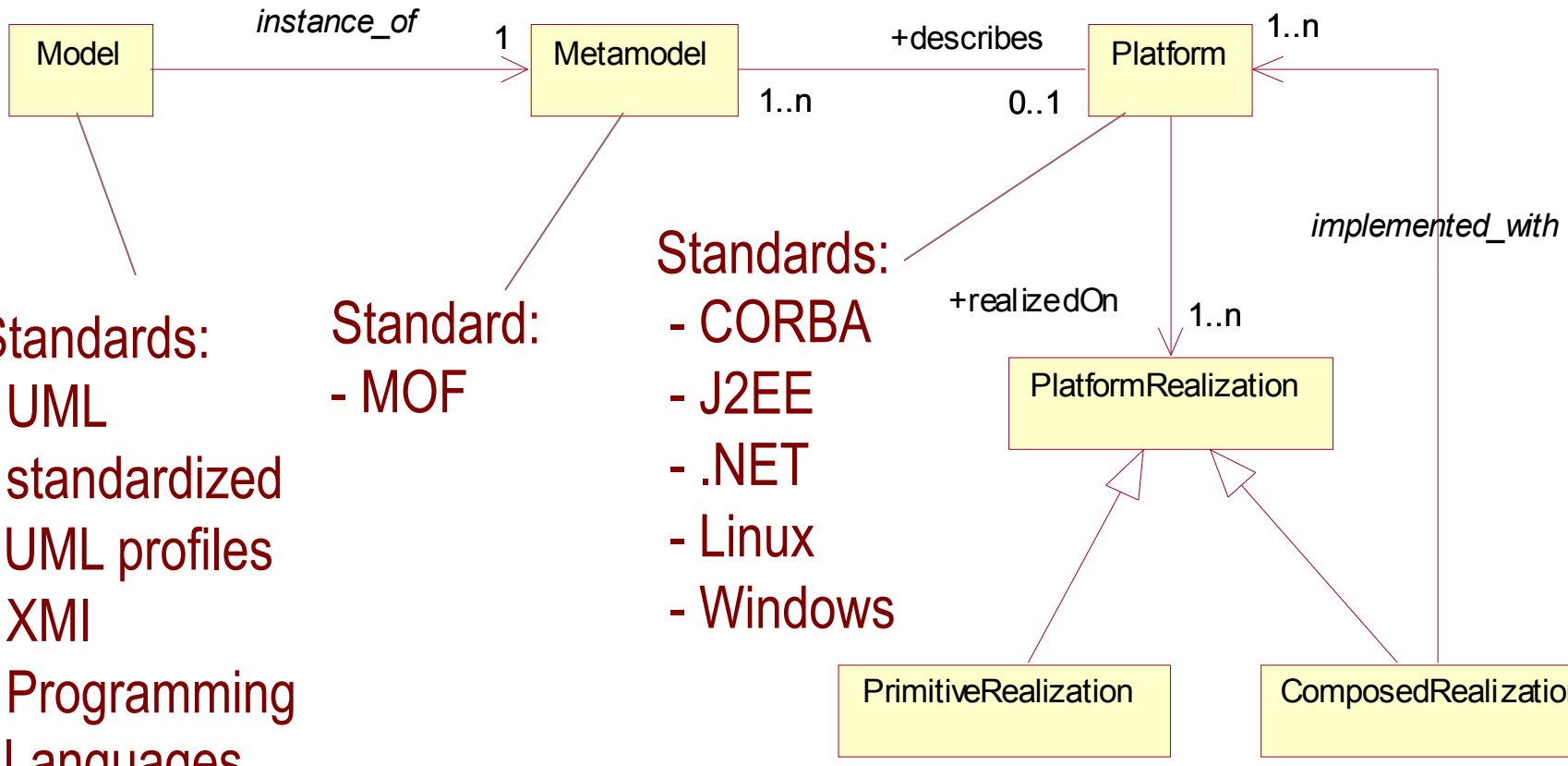
A matter of perspective ☺

Mapping Models

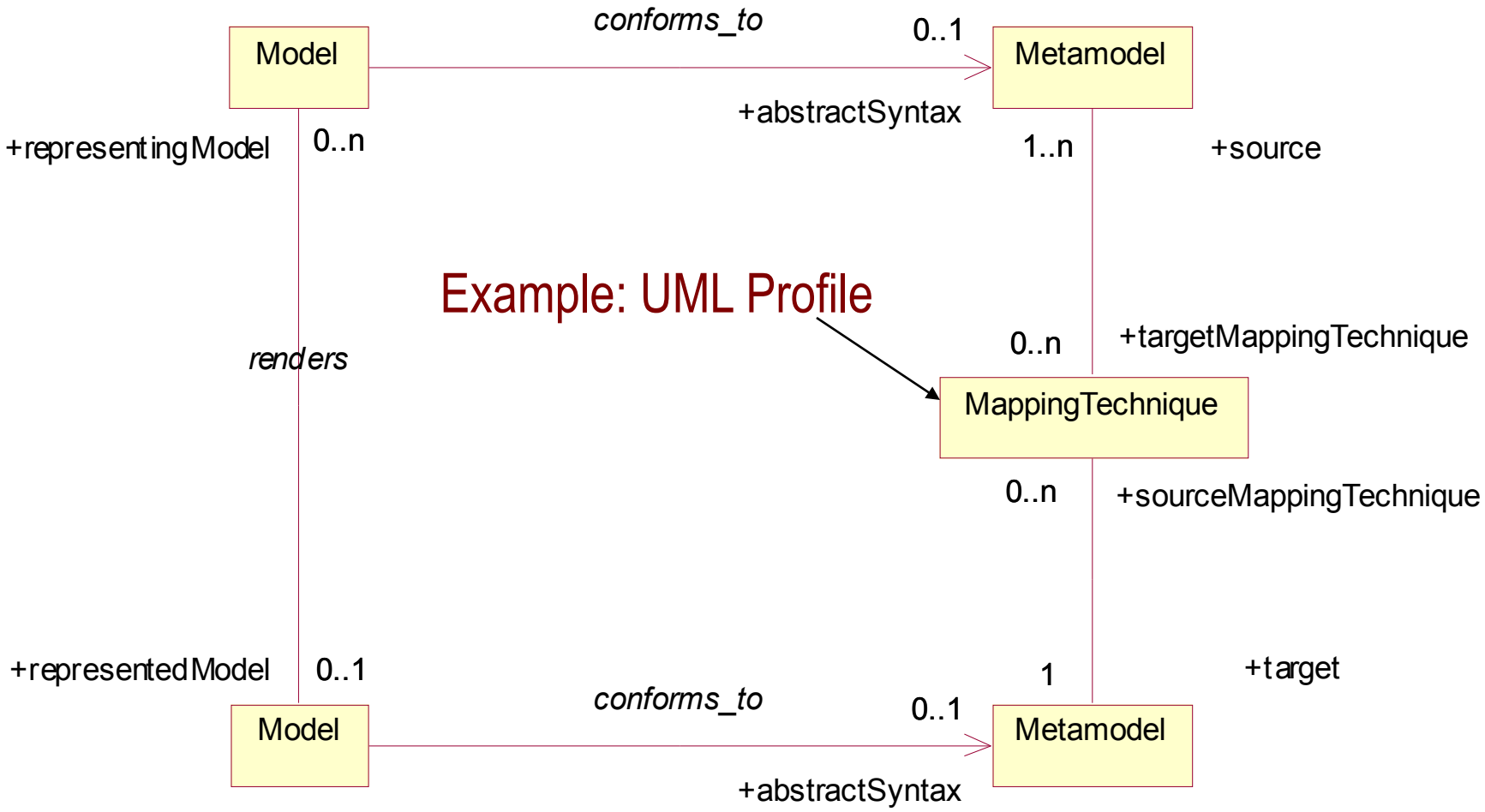


Multiple mappings may be applied successively in a chain

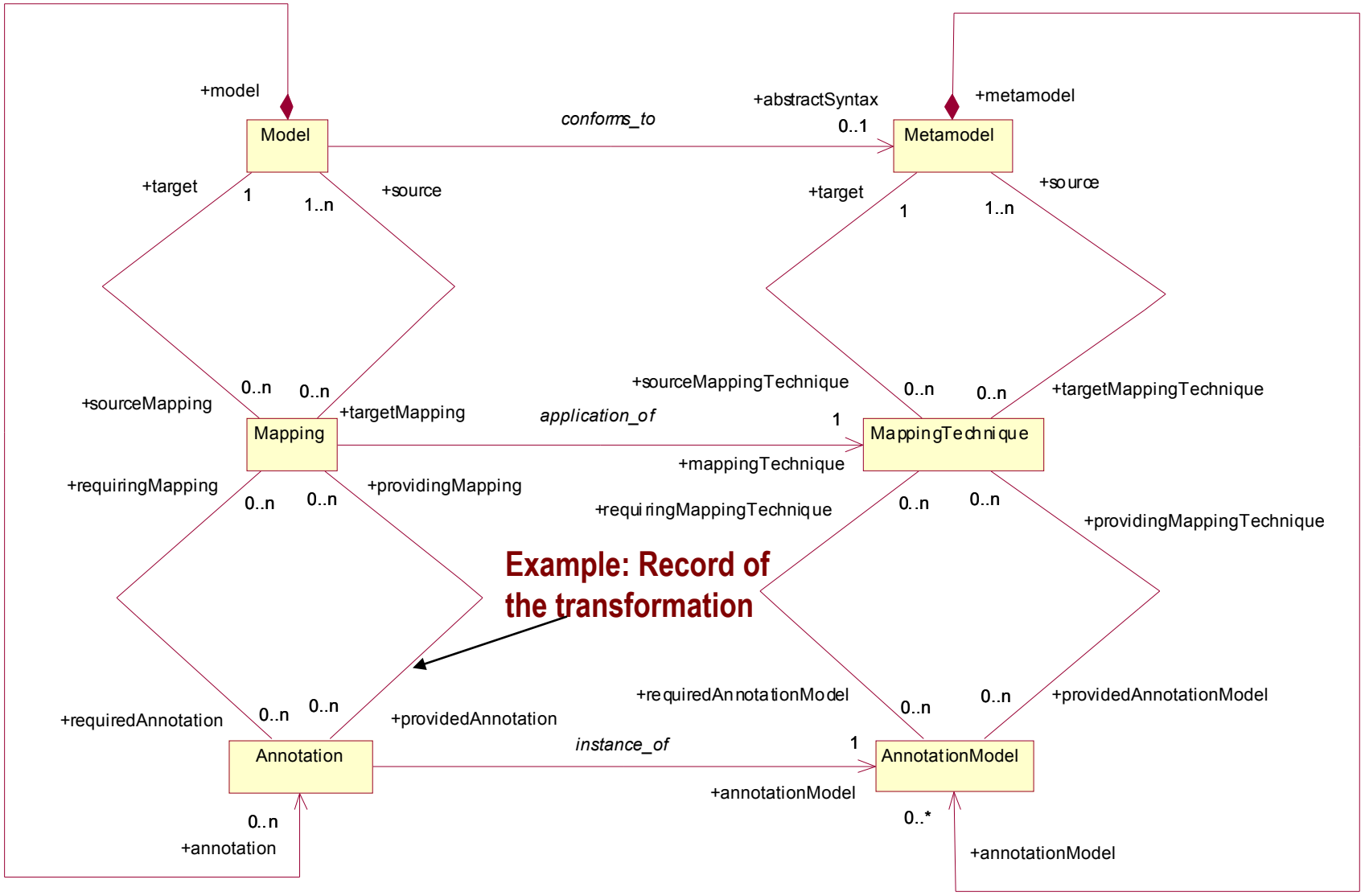
Models, Metamodels, & Platform Stack



Formal: Mapping Techniques



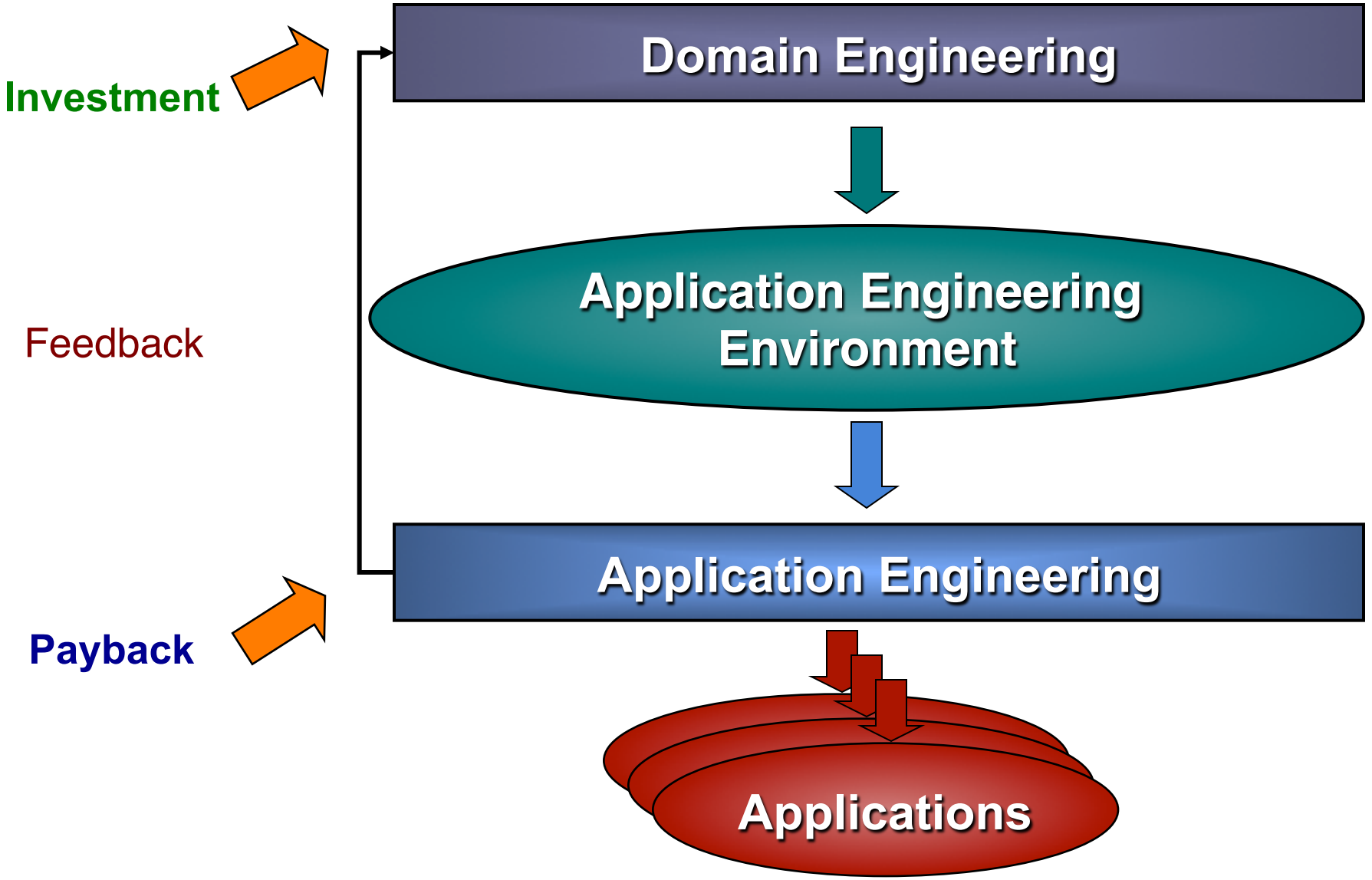
Annotations for Specific Mapping Techniques



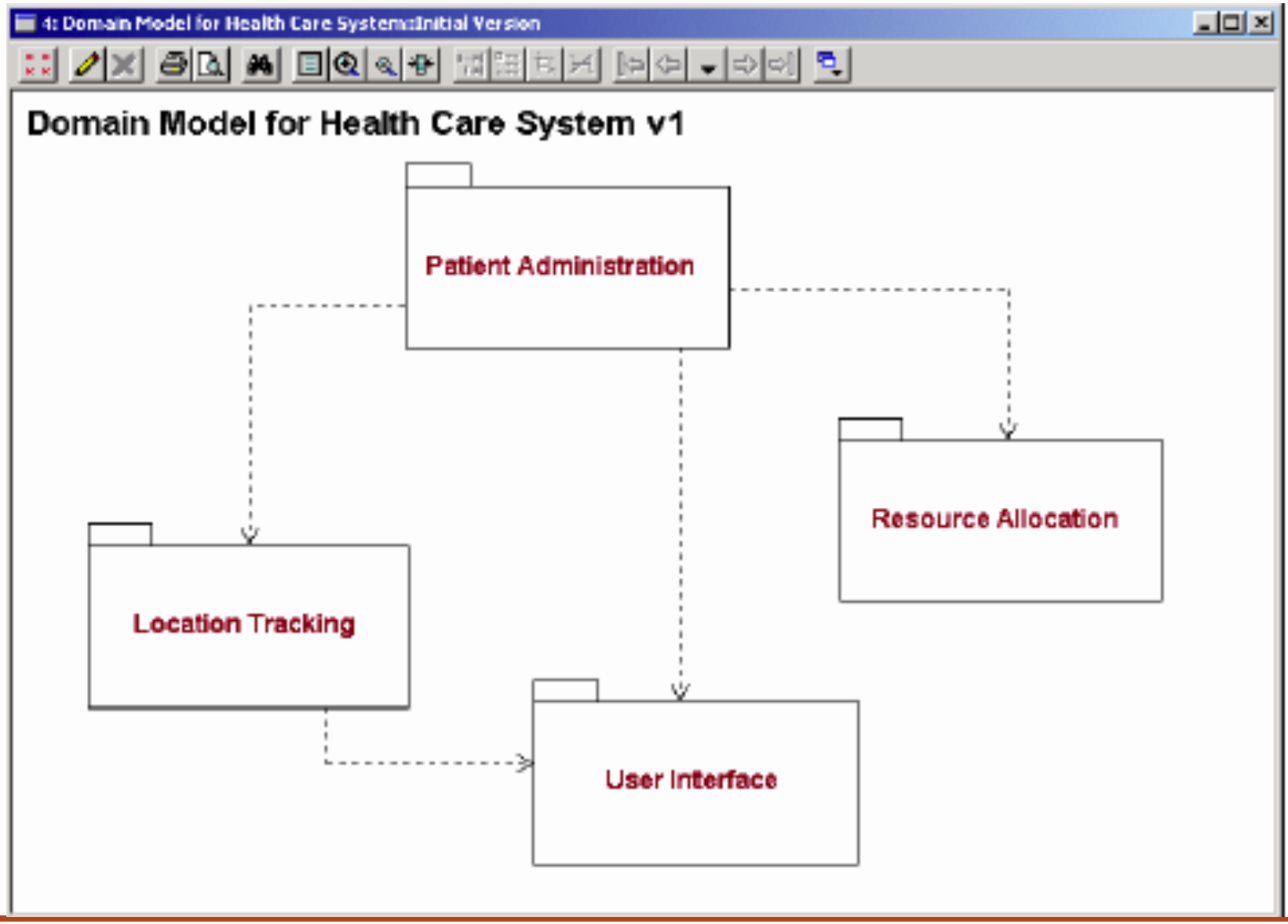
“Analysis” is to “Design” as “Domain” is to _____?

- Again, think for 15 seconds…
- Let’s talk…

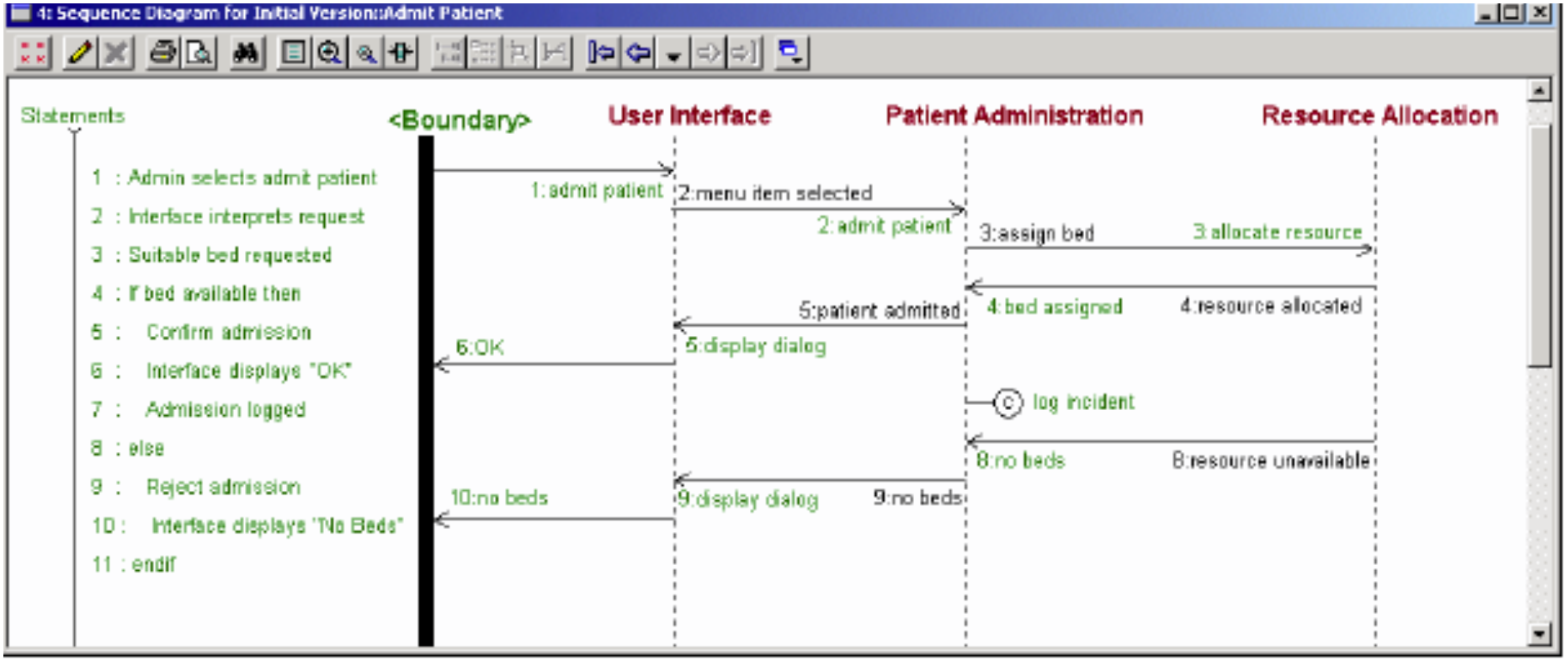
Where is Domain Engineering?



Example: Domain Context

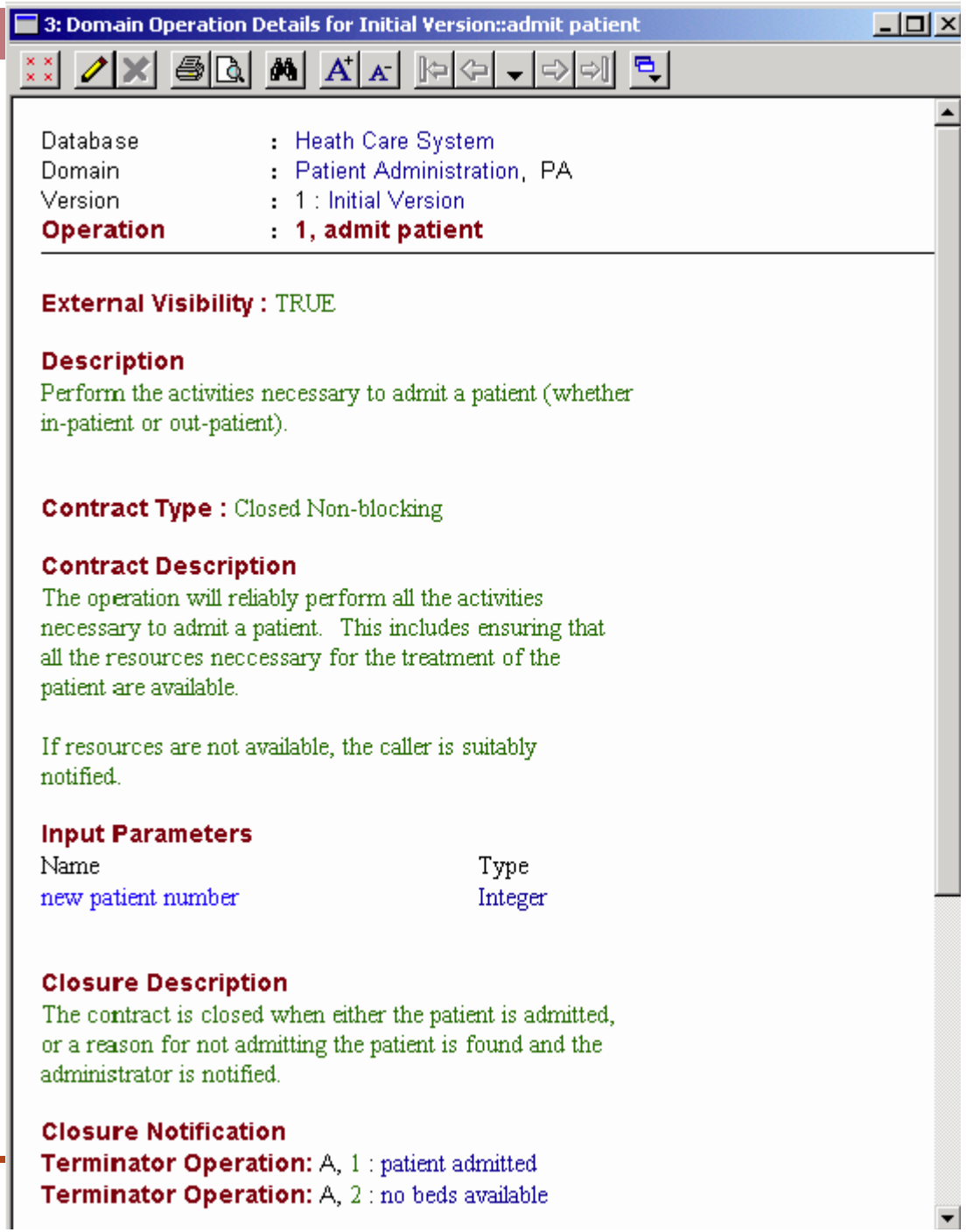


Example: Interactions in Domain



Example: Domain Operation Contract

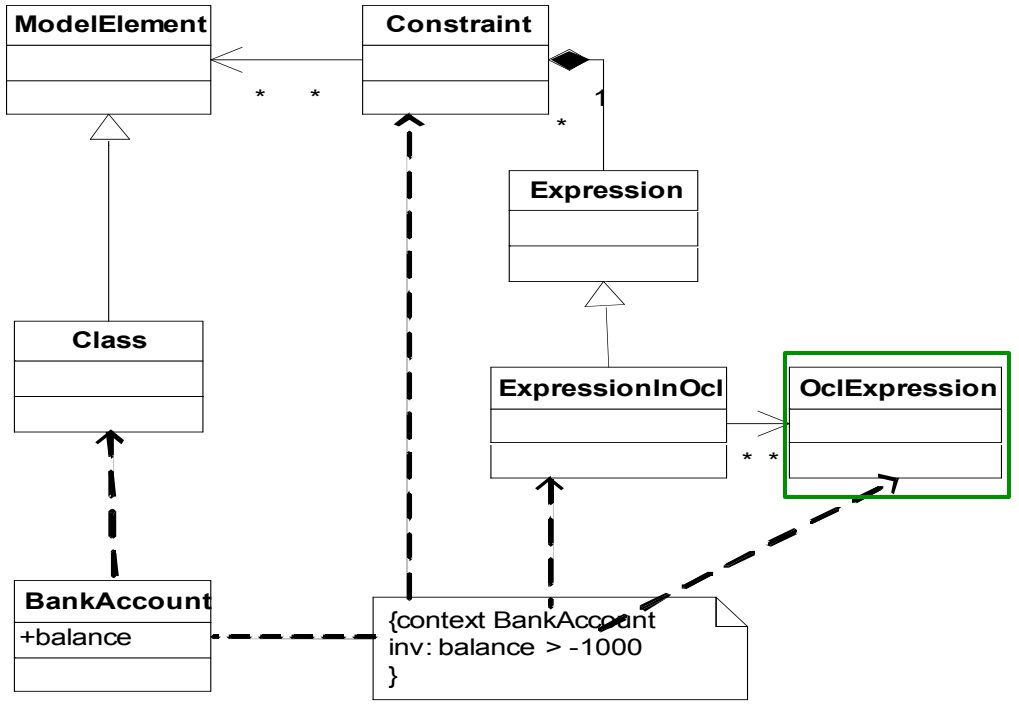
- Note Properties that must be true to admit a patient
- Preconditions
- Post-conditions
- Invariants

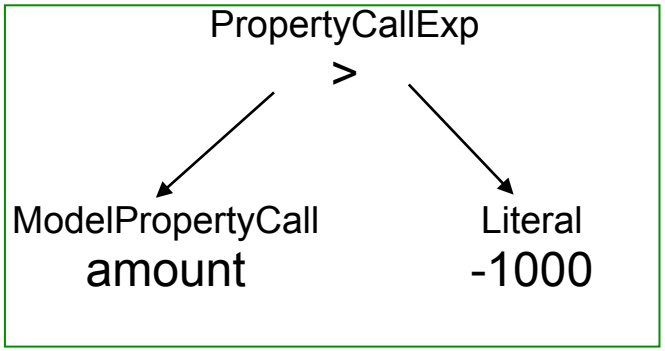


ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Object Constraint Language (OCL)

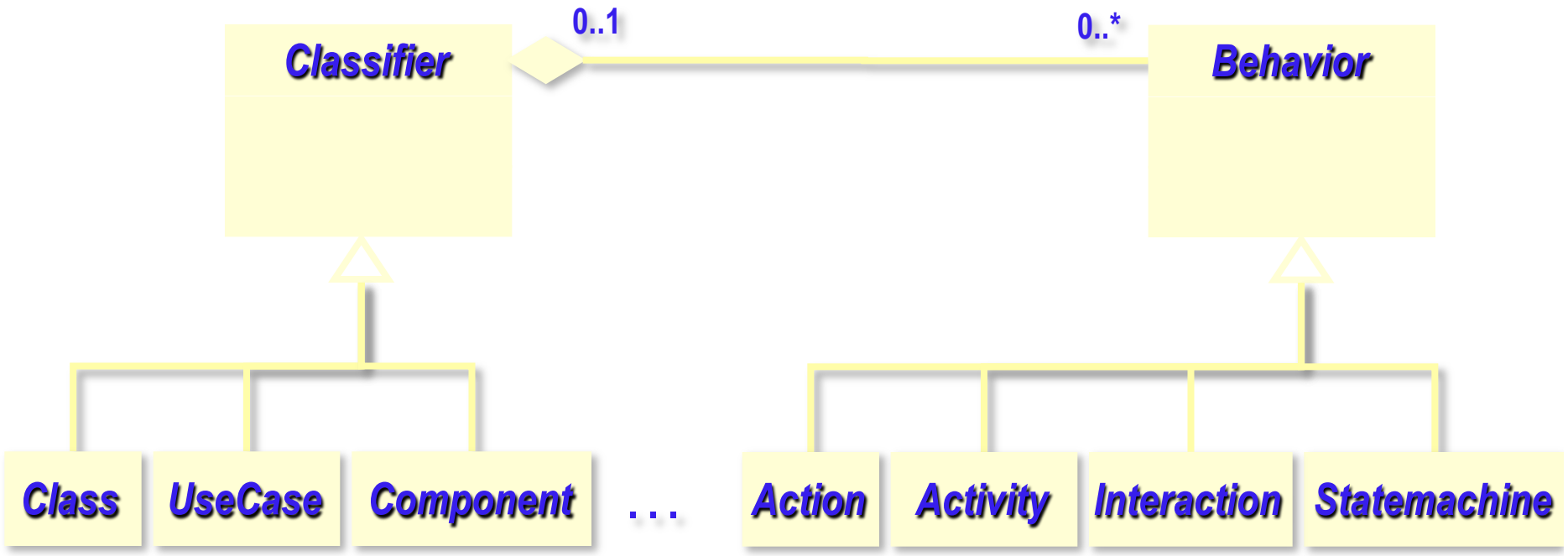
- OCL defines the structure of models expressing constraints
  - Pre and post conditions, Invariants
- OCL is a meta-model instance of the MOF
- The OCL semantic is defined with models (operation without side effect)
- OCL defined a concrete syntax





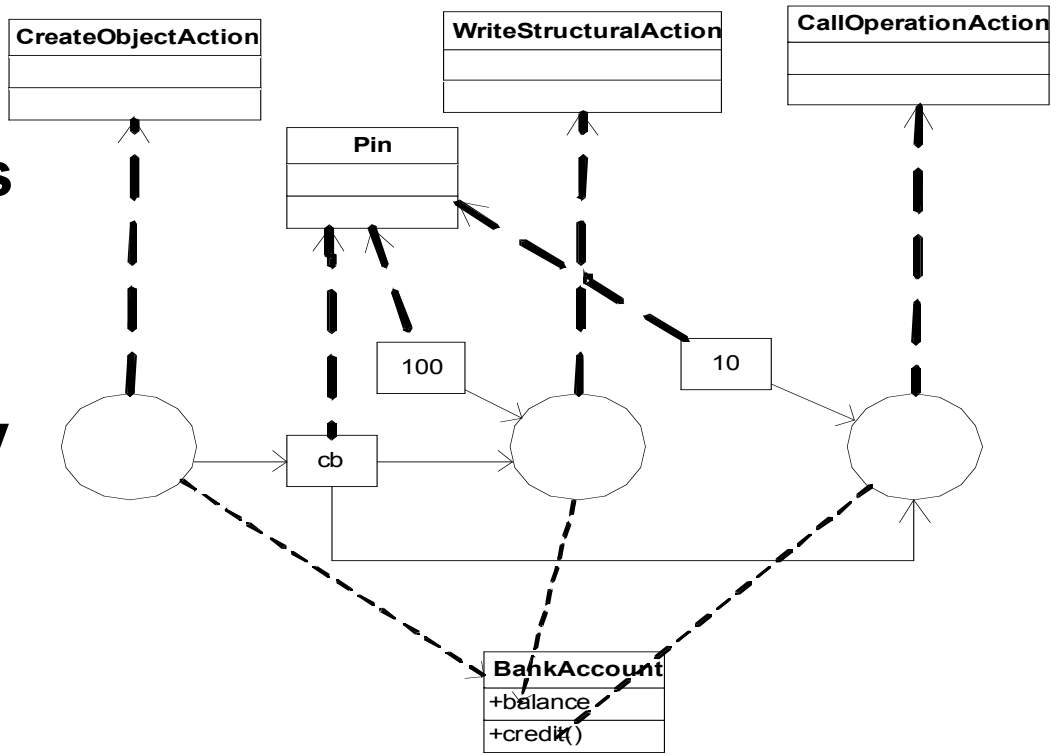
Dealing with Behavior

- Need common semantic base for all behaviors
  - Choice of behavioral formalism driven by application needs



Action Semantics

- AS defines the structure of models expressing sequences of actions
- AS was a meta-model and is now completely integrated in UML 2.0
- AS has no concrete syntax (UML diagram)
- The semantic of AS is not formally defined (an RFP is published)

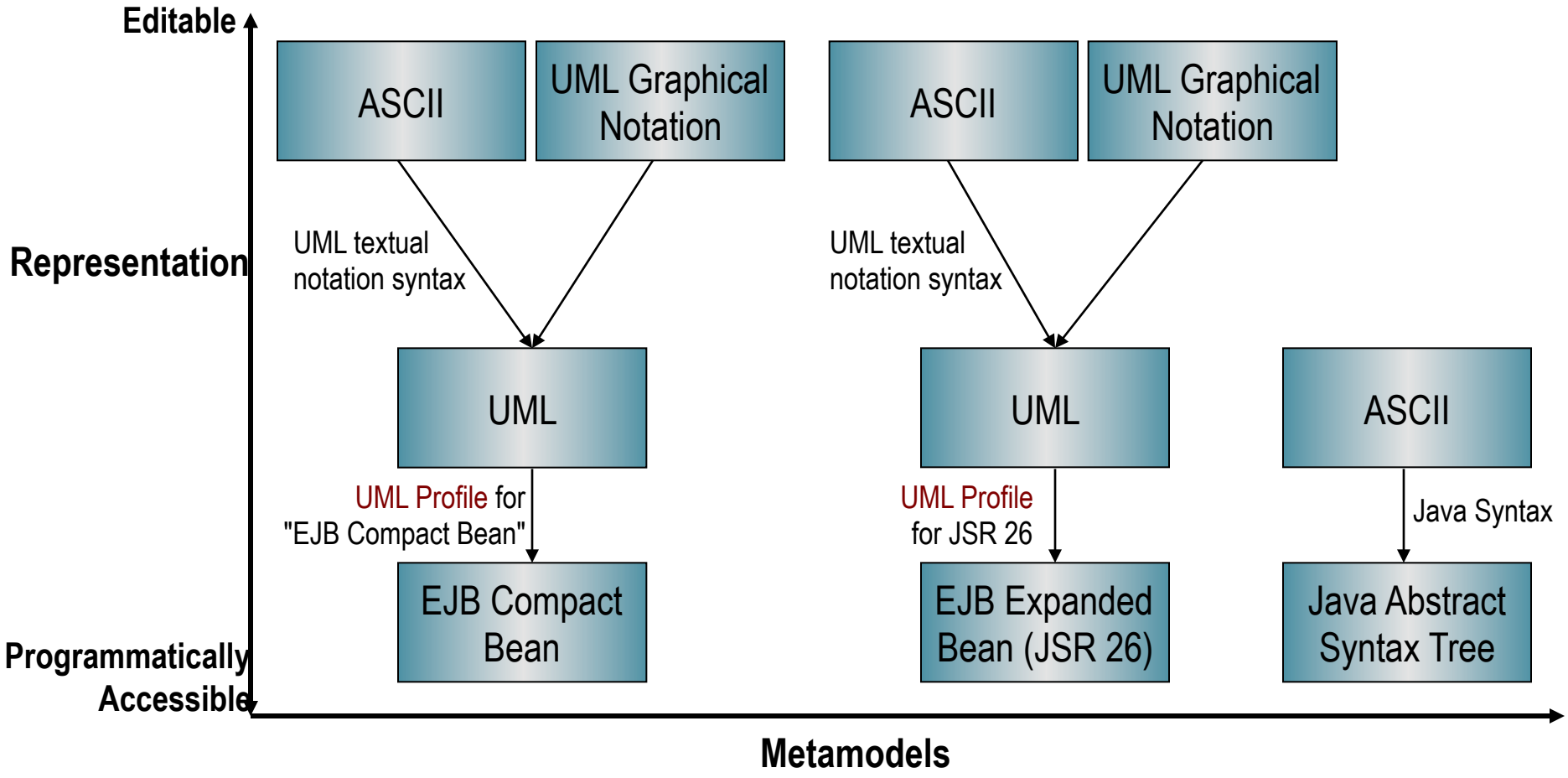


Homework and Milestone Reminders

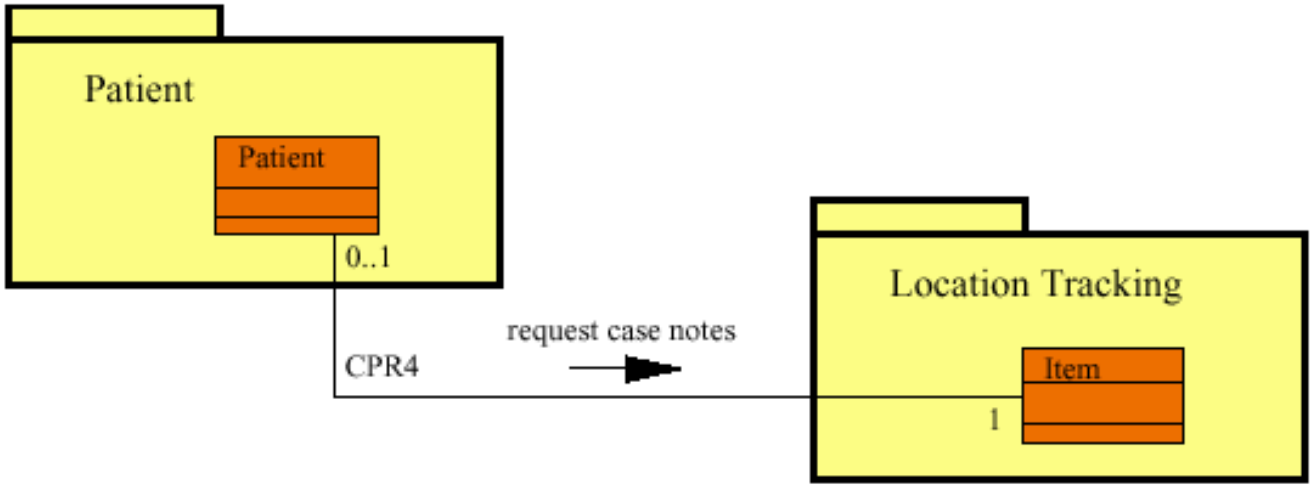
- Milestone 2: Establish a repository and structure for assembling components for your FacePamphlet application
  - Due by 1:35pm Today, April 11th, 2011

- Case Study/Homework: "UML 2: A model-driven development tool" by B. Selic
  - Be prepared to discuss and even lead the discussion
  - Write a brief summary of observations on the paper based on assignment (on Angel)
  - Due by 1:35pm Tuesday, April 12th, 2011

Representing Models: Some Examples



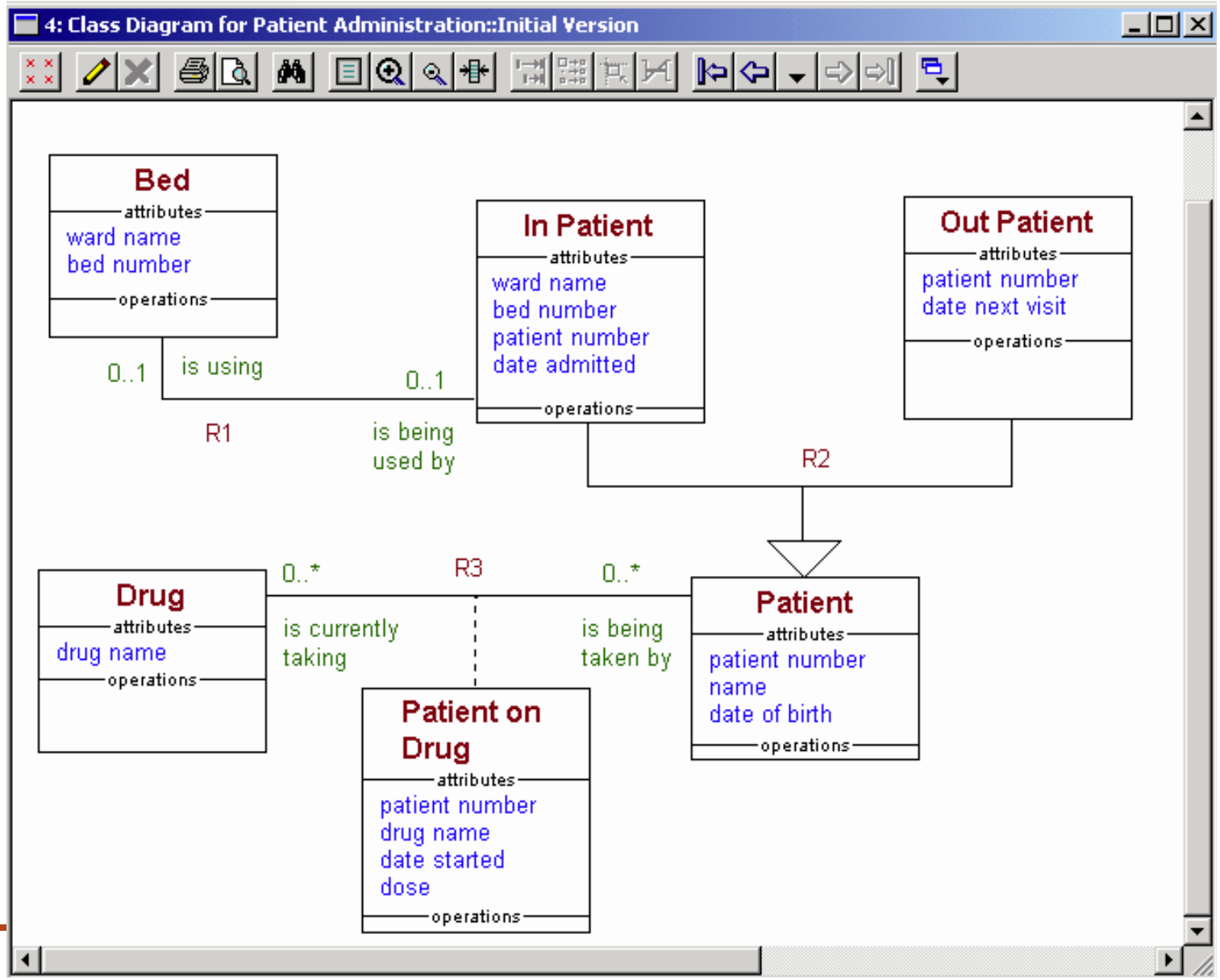
Bridge Mappings: Case Notes



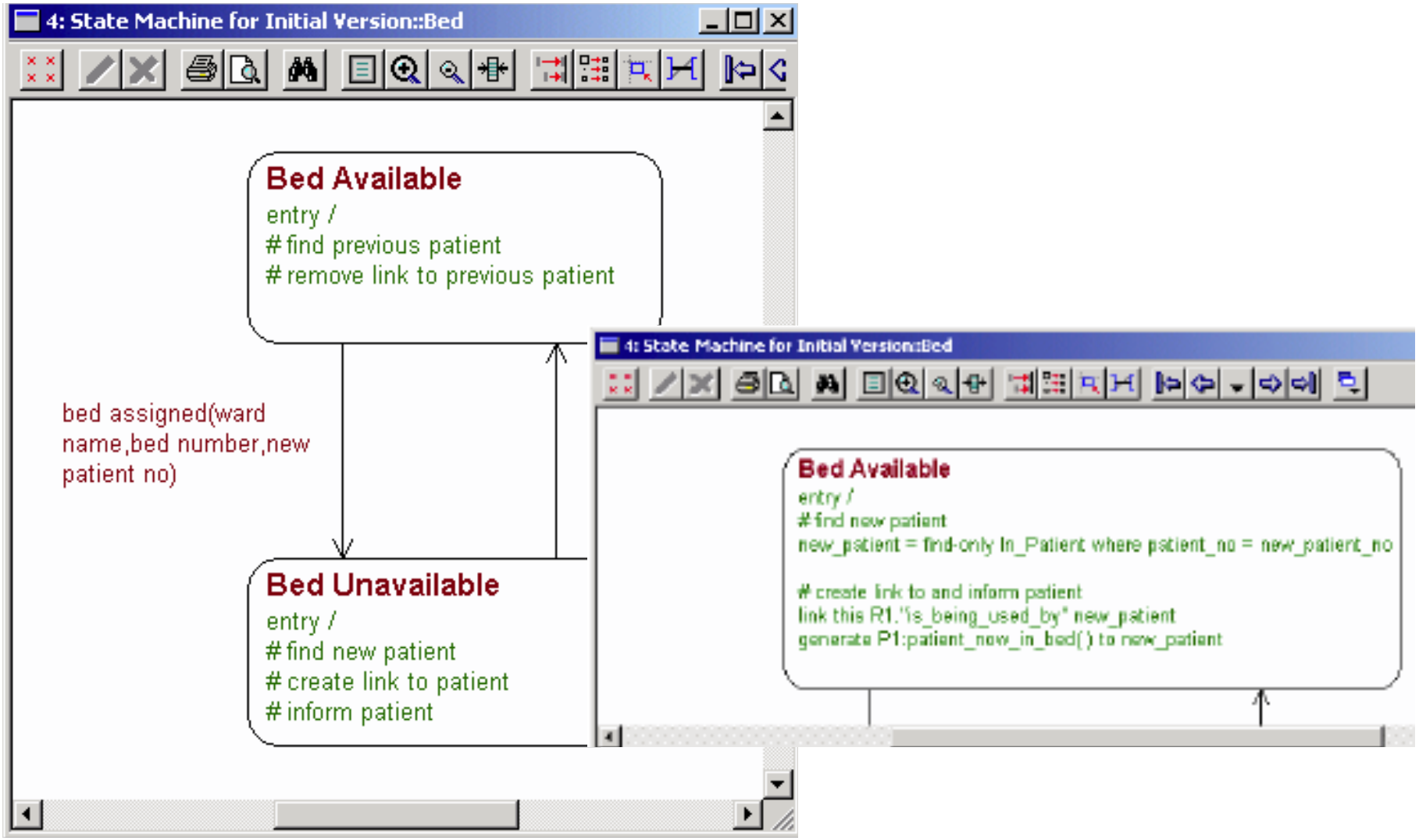

```
define bridge Patient_Administration:LT3_request_case_notes
instance this: Patient
input ward_name: Text
output case_notes_found: Boolean

destination_location = ward_name
counterpart_item = this -> CPR4
$USE LT
  [case_notes_found] = ITM6:locate_item[destination_location]
                                                on counterpart_item
$ENDUSE
```

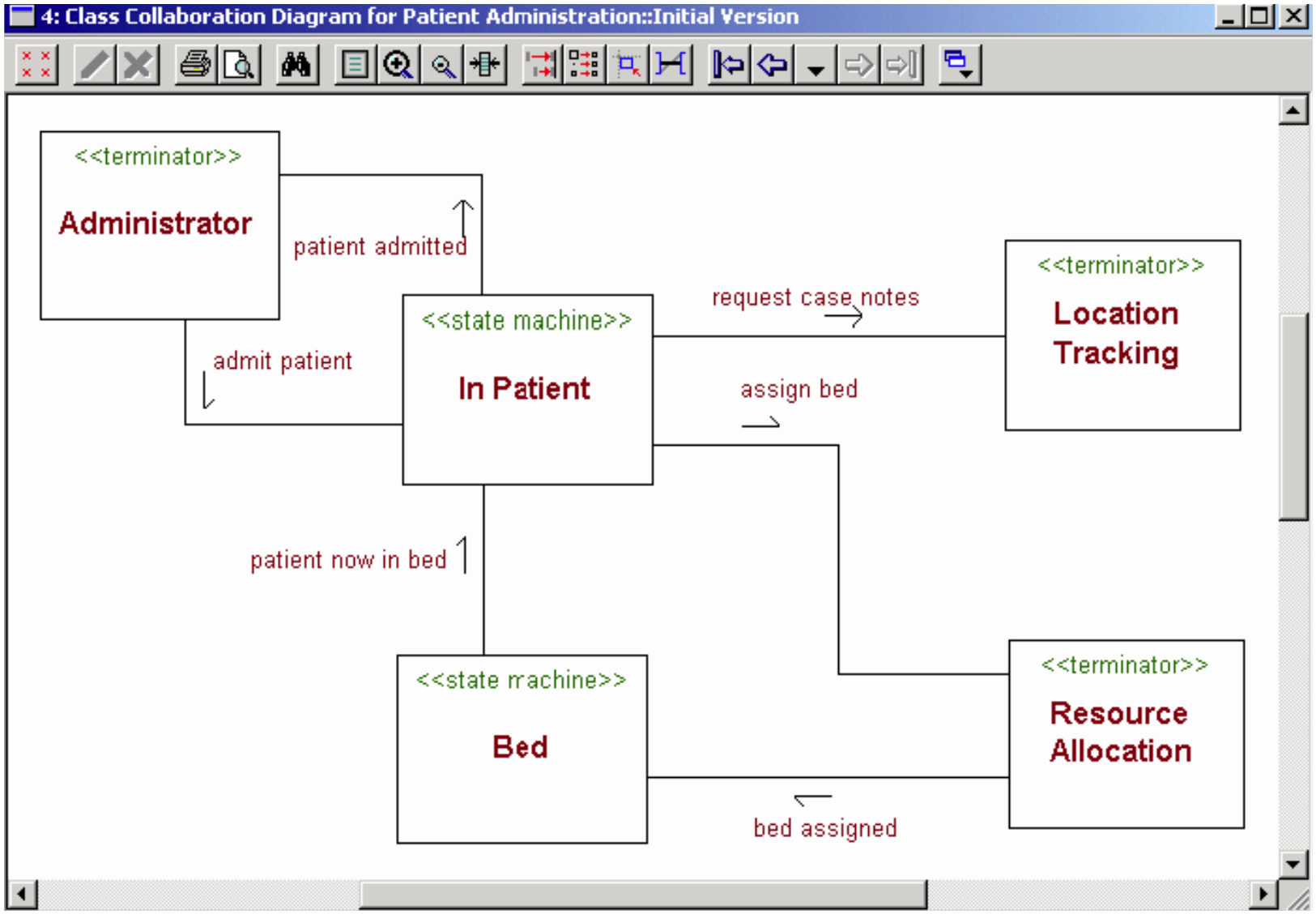
Example: Domain Model



Map to Lower Levels



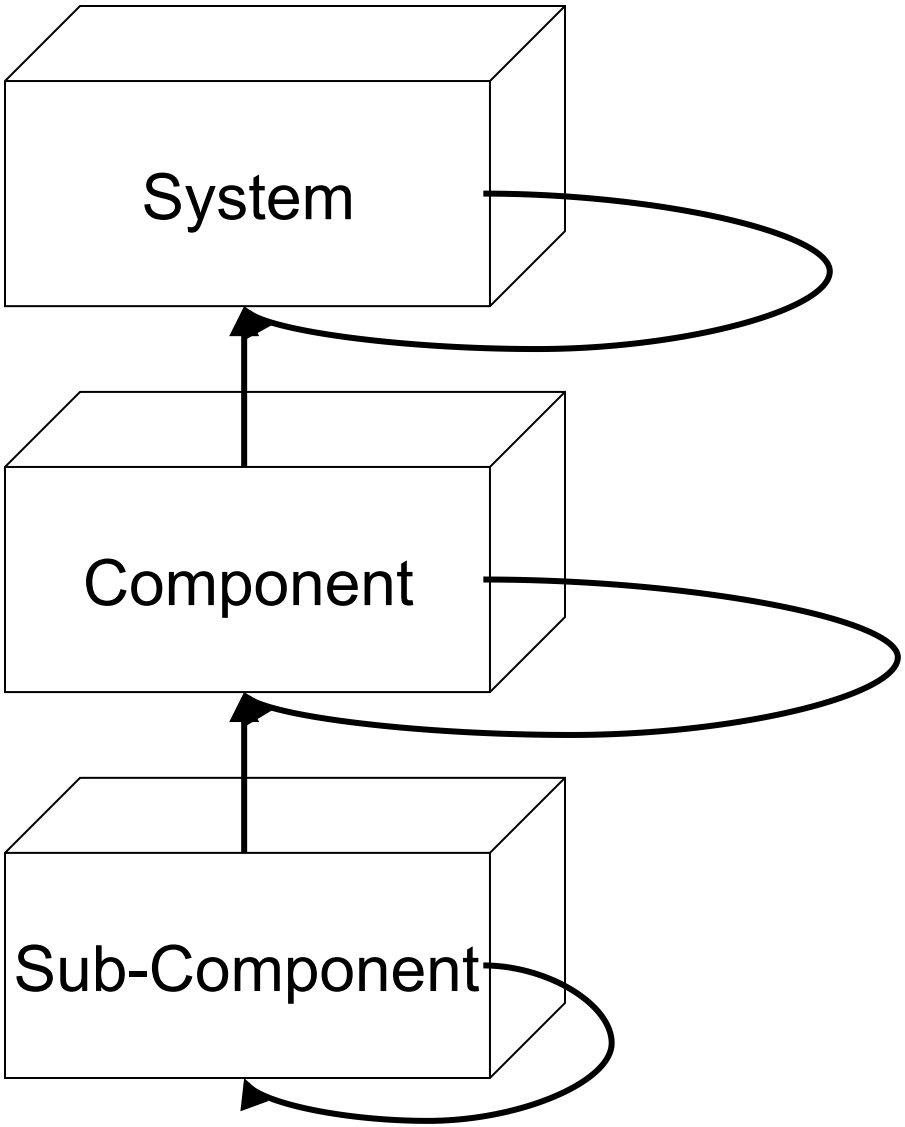
Platform Independent Model (PIM)



Some Open Source Transformers

- Generative Model Transformer (GMT)
  - http://www.eclipse.org/gmt
  - Eclipse project *(vaporware, JUUT-je prototype, UMLX 0.0)*
  - XMI-based (XMI+XMI→XMI, XMI→XMI, XMI→text)
- AndroMDA
  - http://www.amdromda.org
  - Builds on XDoclet, uses Velocity template engine
  - Takes UML XMI input and generates output using cartridges
    - Current cartridges: Java, EJB, Hibernate, Struts
  - Generates no business logic
- Jamda
  - http://jamda.sourceforge.net
  - Takes UML XMI file as input, using Jamda profile
  - Java-based code generators
    - Generates class definitions added to UML model before codegen

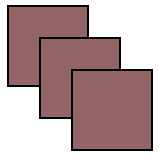
But what about Assembly?



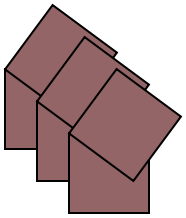
Product Line Philosophy

- Power of a product line lies in its ability to leverage common features despite necessary variances between different systems in the domain
- Viability of the product-line approach depends on predictable variances
- Entails a significant change in mindset
  - Cultural issue poses the greatest challenge to adopting a product-line approach

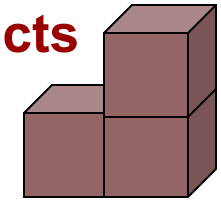
Use of a common asset base



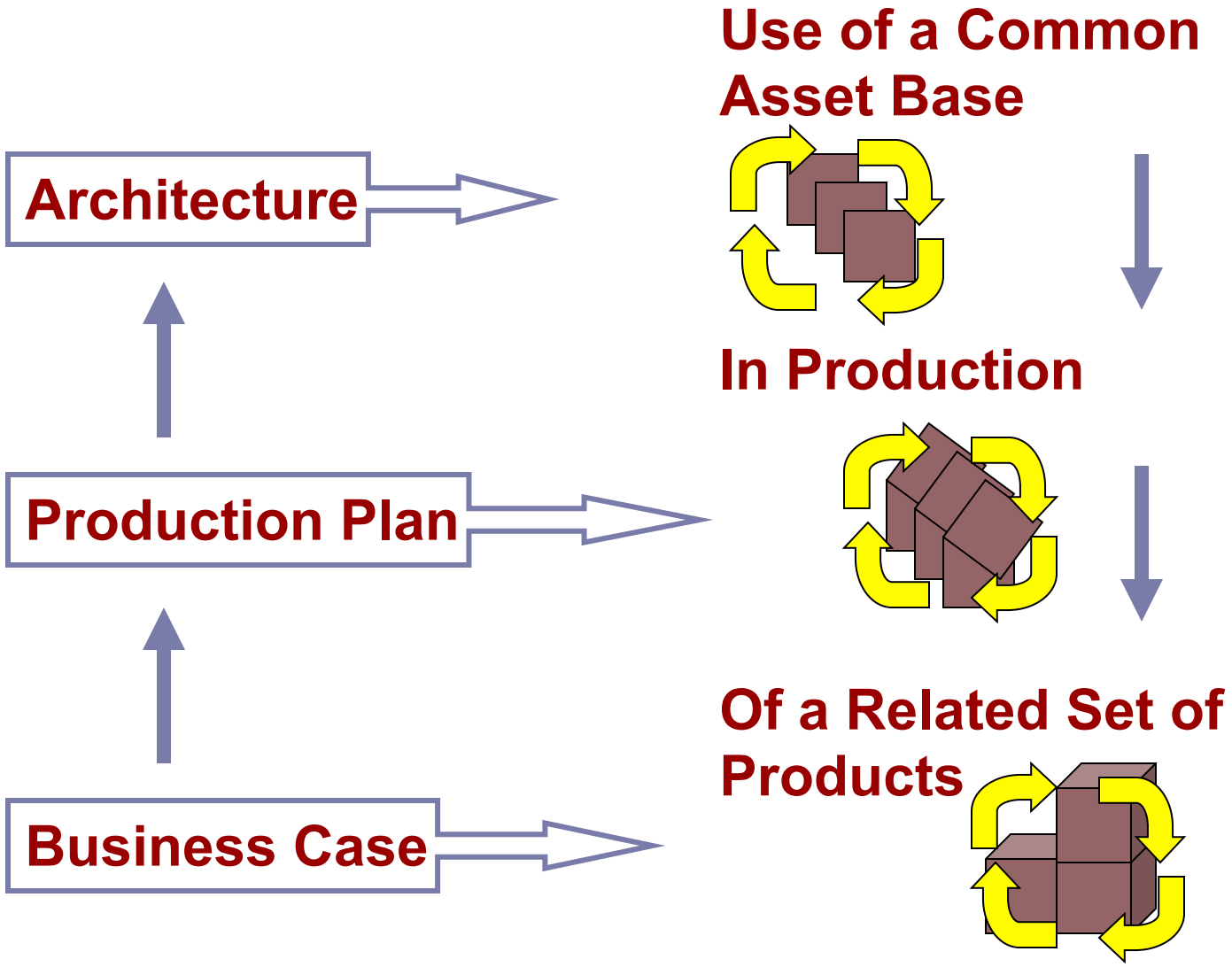
In production



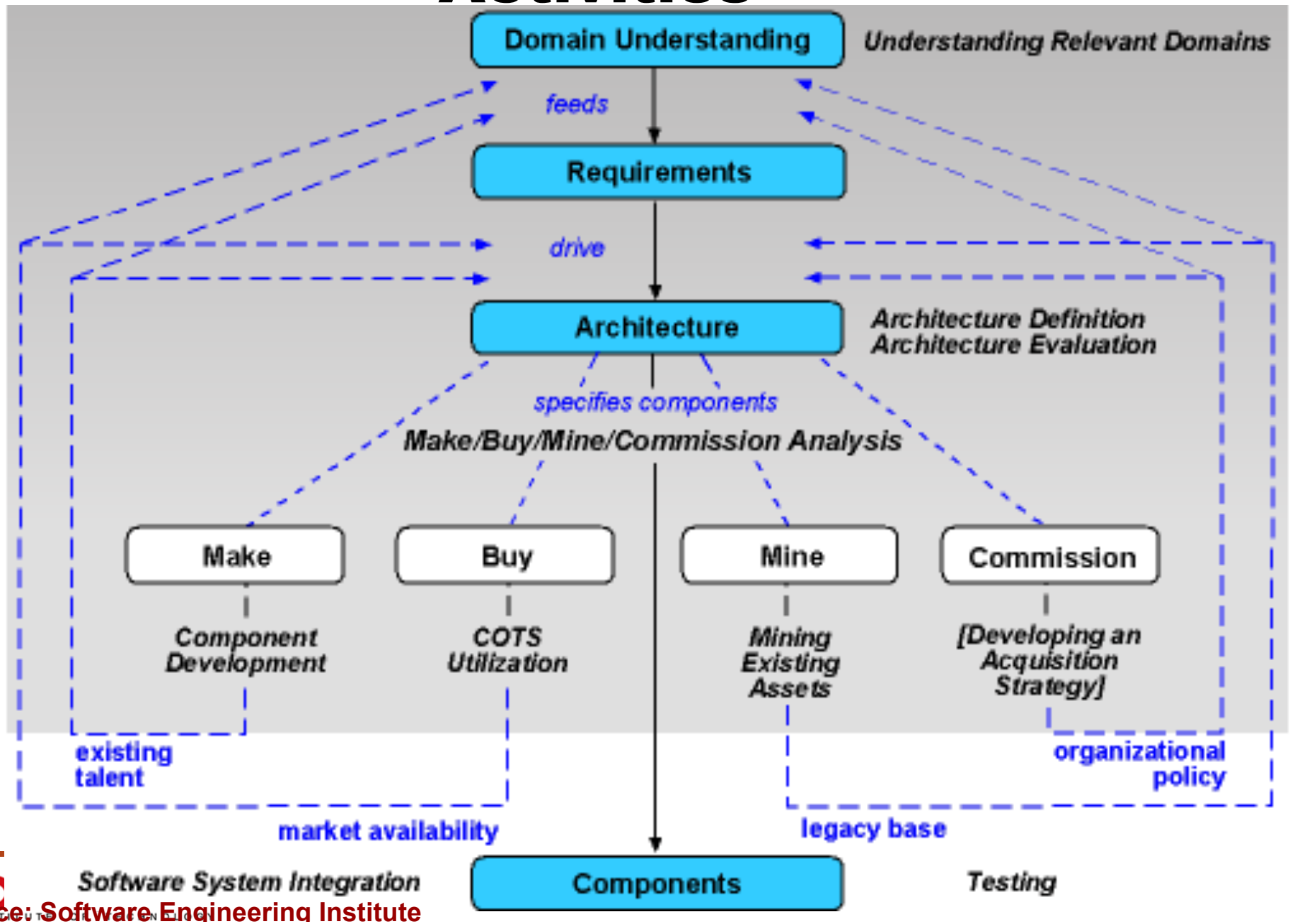
Of a related set of products



Key Product Line Concepts

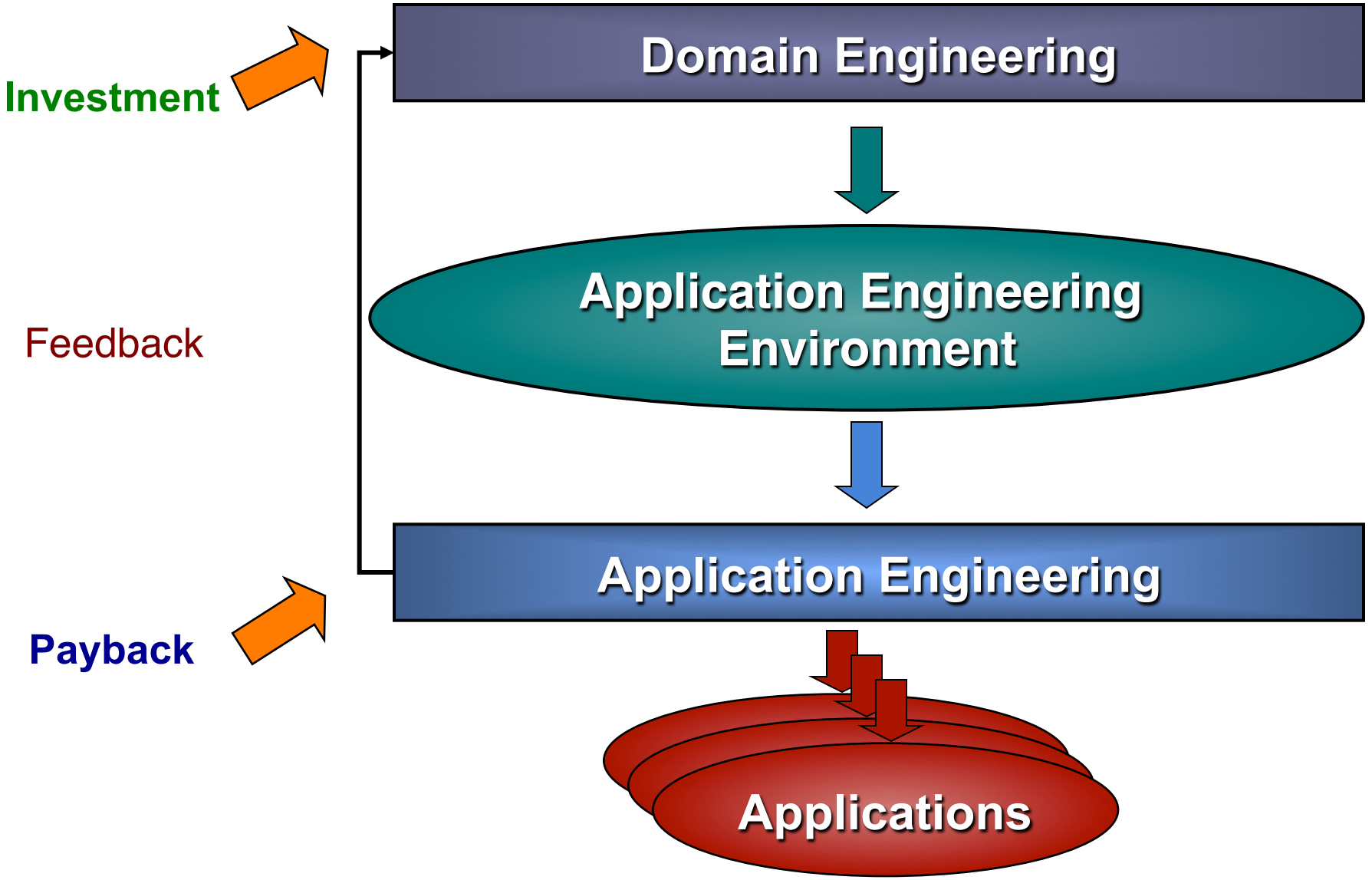


Ecosystem: Key Product Line Activities

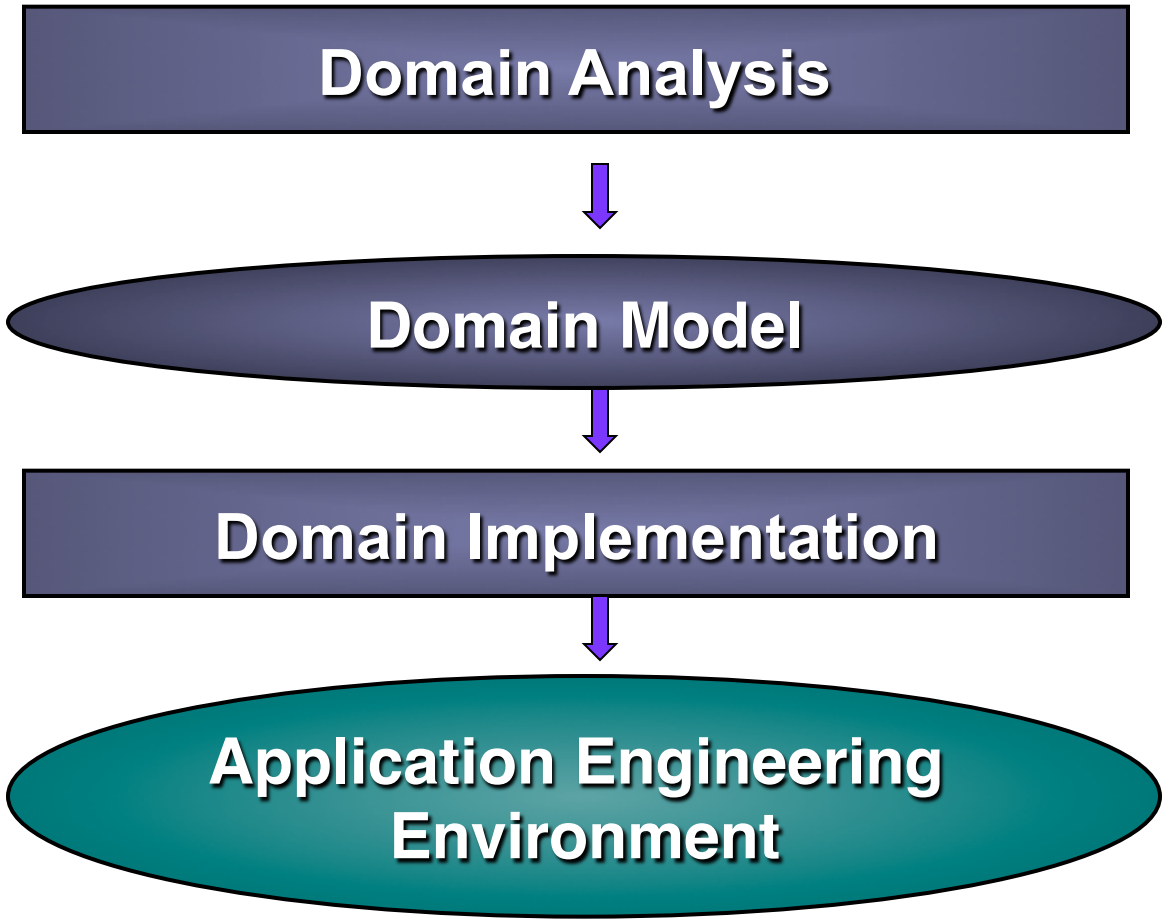


Source: Software Engineering Institute

A FAST Process



Domain Engineering



Analysis Document, Application Modeling Language

Tools, Process

The Domain Model

- Conceptual Framework
  - Family Definition
    - Commonalities and Variabilities Among Family Members
    - Common Terminology for the Family
    - Abstractions for the Family
  - Economic Analysis
  - Application Modeling Language (AML)
    - Language for stating requirements
- Mechanism for translating from AML to Code
  - Alternative 1: Compiler
  - Alternative 2: Composer

Building The Conceptual Framework

- Qualify The Domain
  - Is it economically viable?
- Define The Decision Model
  - What decisions must be made to identify a family member?
- Define The Family
  - What do members of the family have in common and how do they vary?
- Design The Application Modeling Language
  - What is a good way to model a family member?
- Design The Application Engineering Environment
  - What are good mechanisms for using the decision model and the Application Modeling Language?

Defining The Family: Commonality Analysis

- Dictionary of terms
  - Technical terms that define a vocabulary for the domain
    - Primary Condition: The availability of a unit: working: ready, unready, or unusable
- Commonalities: Assumptions that hold for every member of the family
  - Every unit must be in one of the four primary conditions.
- Variabilities: Assumptions that define the range of variation for the family
  - Some unit names have inhibit states.
- Parameters of Variation: Quantification of the variabilities
  - Whether or not a unit name can have an inhibit state: Boolean

Reusable Assets

- Validations -- generic algorithms for every unit type
- Realizations -- generic algorithms for every unit type
- Relationships
  - data that is used to drive the generic algorithms
  - design information shared across development

Case Study/Homework:

"UML 2: A model-driven development tool"
by Bran Selic

- What are the alternatives?
- How hard are they to implement?
- Is there support from the community?

MOF Action Semantics

- EMF has limited Behavioral Modeling support
- Action semantics capture the behavior of a model (i.e., how the model behaves)
- Actions semantics has been proposed for UML 2.0.
  - Variants appear in Executable UML
- Let's talk more about Action semantics and Object Constraint Language (OCL) on Monday