CSSE 490 Model-Based Software Engineering: Concluding Domain Engineering

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

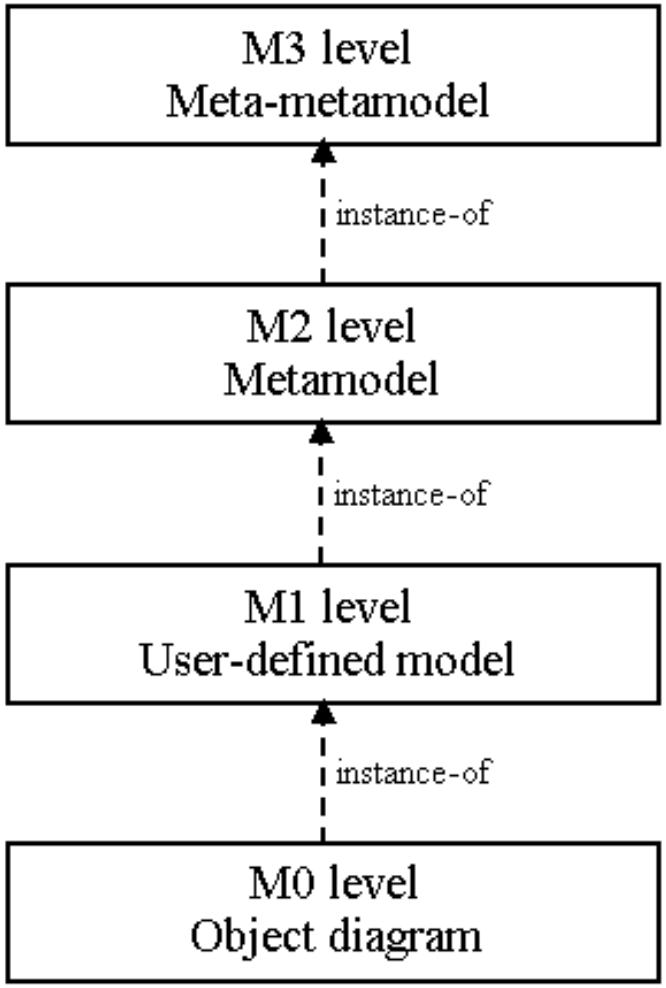
Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Learning Outcomes: Metamodels

Design a metamodel for a model-based software system.

- Consider Term Papers
- Introduce Object Constraint Language (OCL)
- Action Semantics (if time)
- Domain Engineering



Write and Present a Term Paper

- Use IEEE/ACM format for the paper (template provided on Angel)
- Include abstract, introduction, background/ related work, analysis, and conclusion (along with references)
- Target 5-7 pages
  - If you are not a strong writer, use a lot of tables and figures to organize your work
- Use your own words - copied elements without reference are considered plagiarism
- Paper due May 17th, 2011
- Presentation on May 19th, 2011

Topics for Term Paper

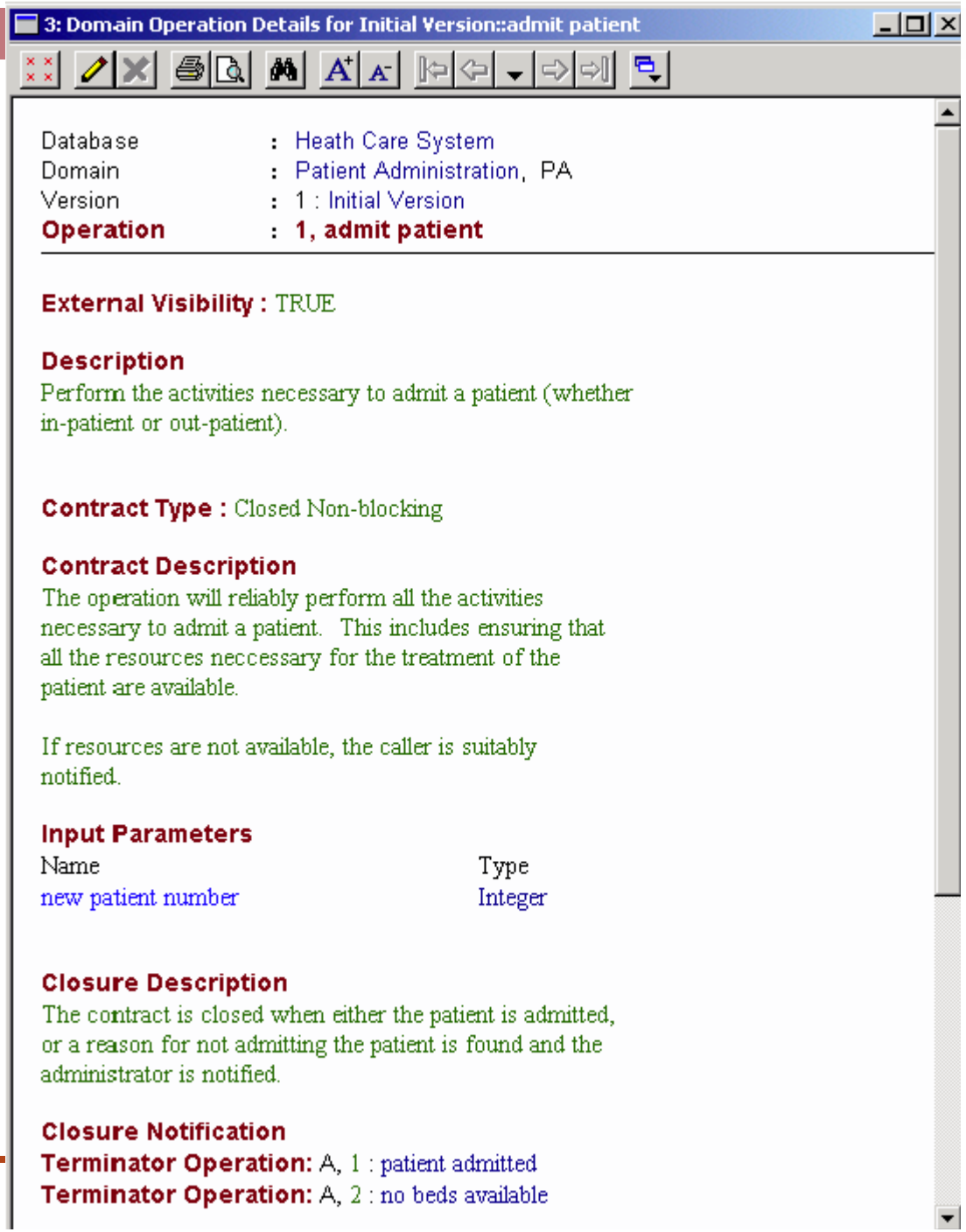
1) Critically analyze the state of software productivity and the potential for Model-Based Engineering to make an impact.
2) Conduct a survey of Model-Based Engineering approaches (e.g., MDA/MDD, MBSE, DSL, MIC, etc.) to compare and contrast them.
3) Survey Model-Based Engineering in other disciplines (e.g., civil, mechanical,) comparing them with MBSE.
4) From a macro-economic perspective, evaluate the cost-benefit of Model-Based Engineering for software.
5) Critically analyze advances in automatic programming from a feasibility perspective and outline how these implications are relevant for software today.
6) Survey applications of "Product-Lines" to software systems and present arguments for a Model-Based Engineering approach.
7) Critically analyze transformation technology in the production of Model-Based/Driven Engineering software solutions.
8) Survey studies of Model-Driven Architecture (MDA) for the state-of the practice and outline key criteria for success and failures.
9) Suggest one that you would be more motivated to do!

What is needed to identify the key reusable elements of a given application domain?

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

Example: Domain Operation Contract

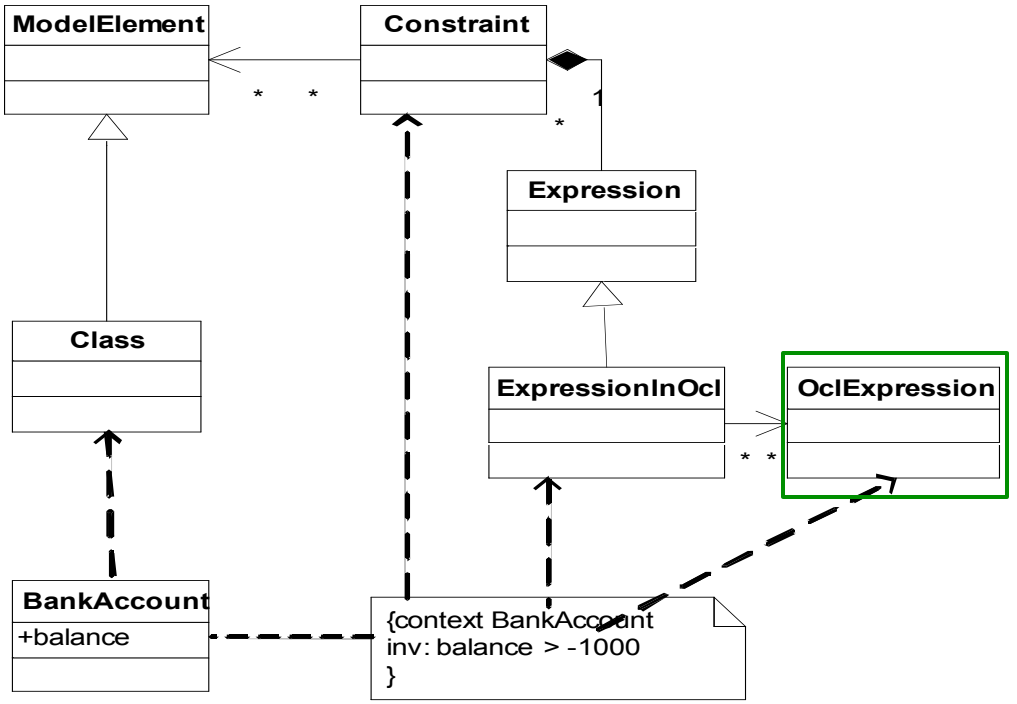
- Note Properties that must be true to admit a patient
- Preconditions
- Post-conditions
- Invariants

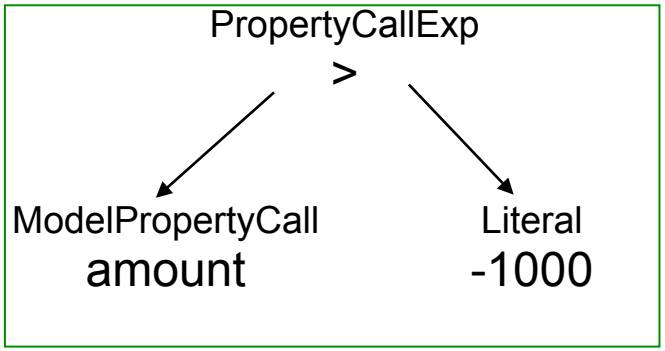


ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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





Simple OCL Examples

- For an Airline Reservation

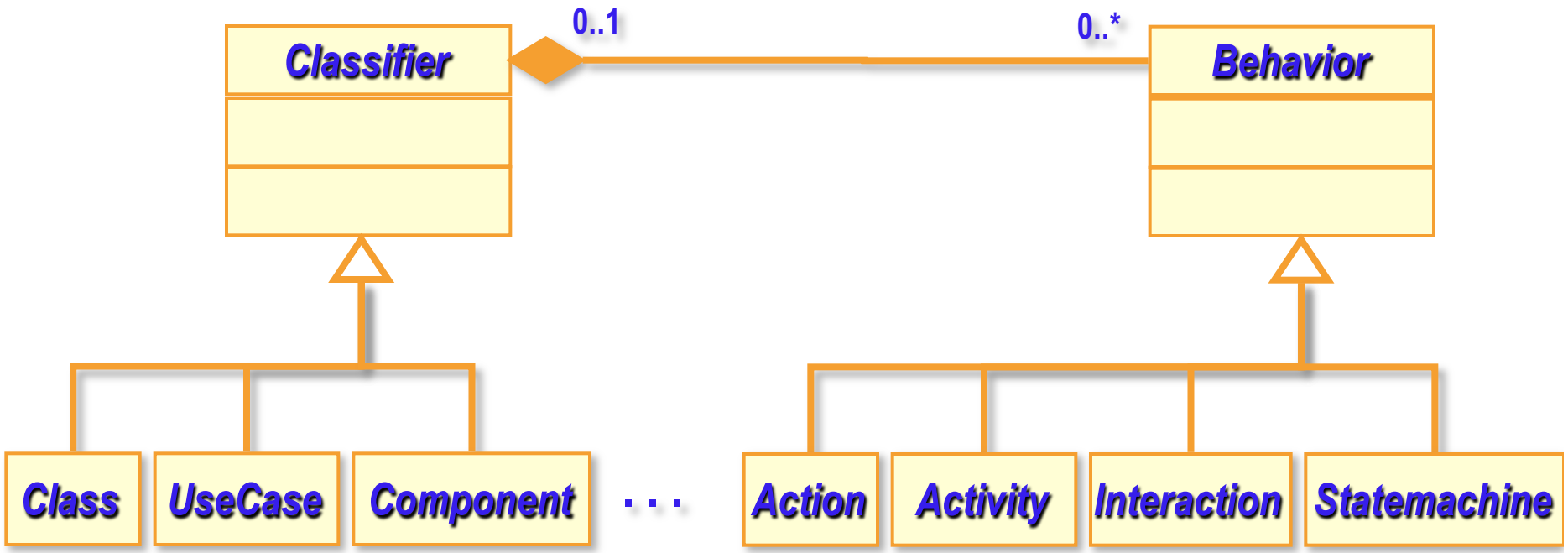
```
context Passenger::book(f : Flight)
pre:  f.maxNrPassengers >
            f.passengers->size
post: f.passengers =
      f.passengers@pre->including(self)
```

- For Airports Served

```
context Airline::servesAirports() : Set(Airport)
pre:        none  --i.e. true
post:       result = flights.destination->asSet
```

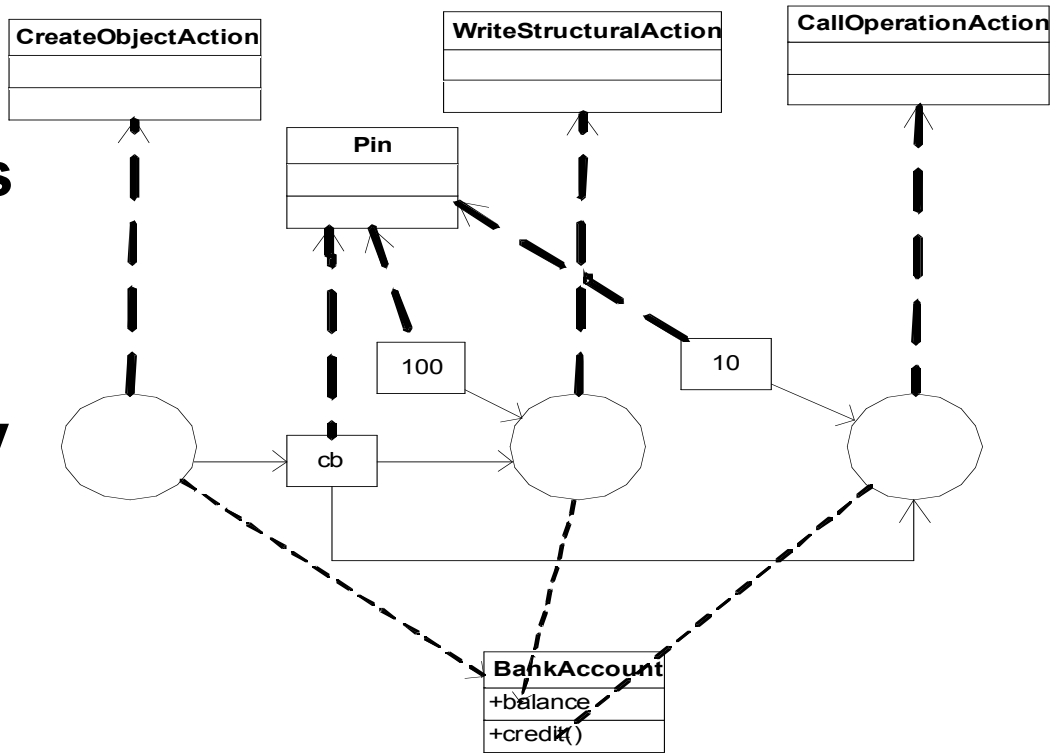
Dealing with Behavior

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



Action Semantics

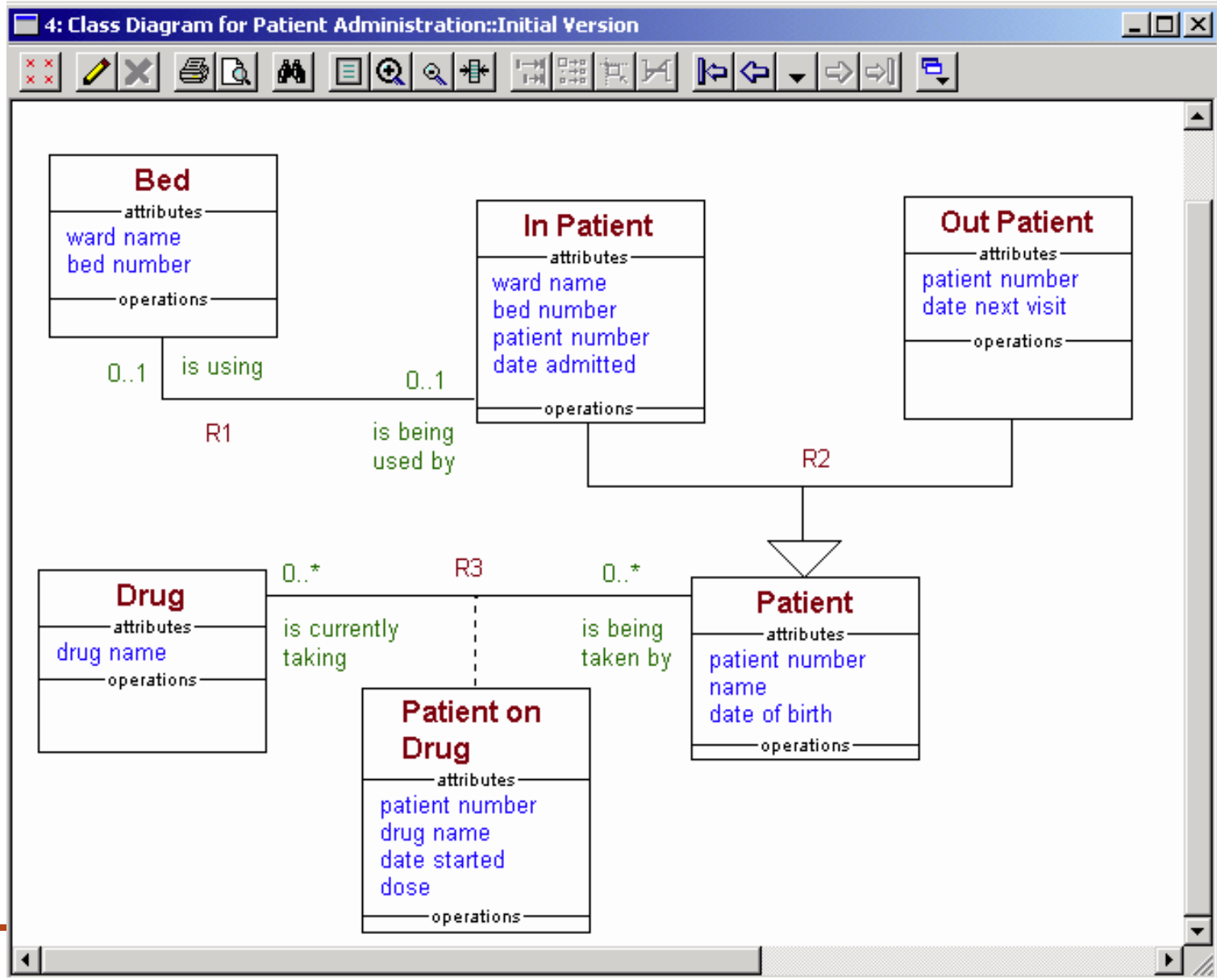
- AS defines the structure of models expressing sequences of actions
- AS was a meta-model and is now completely integrated in UML 2.0
- AS has no concrete syntax (UML diagram)
- The semantic of AS is not formally defined (an RFP is published)



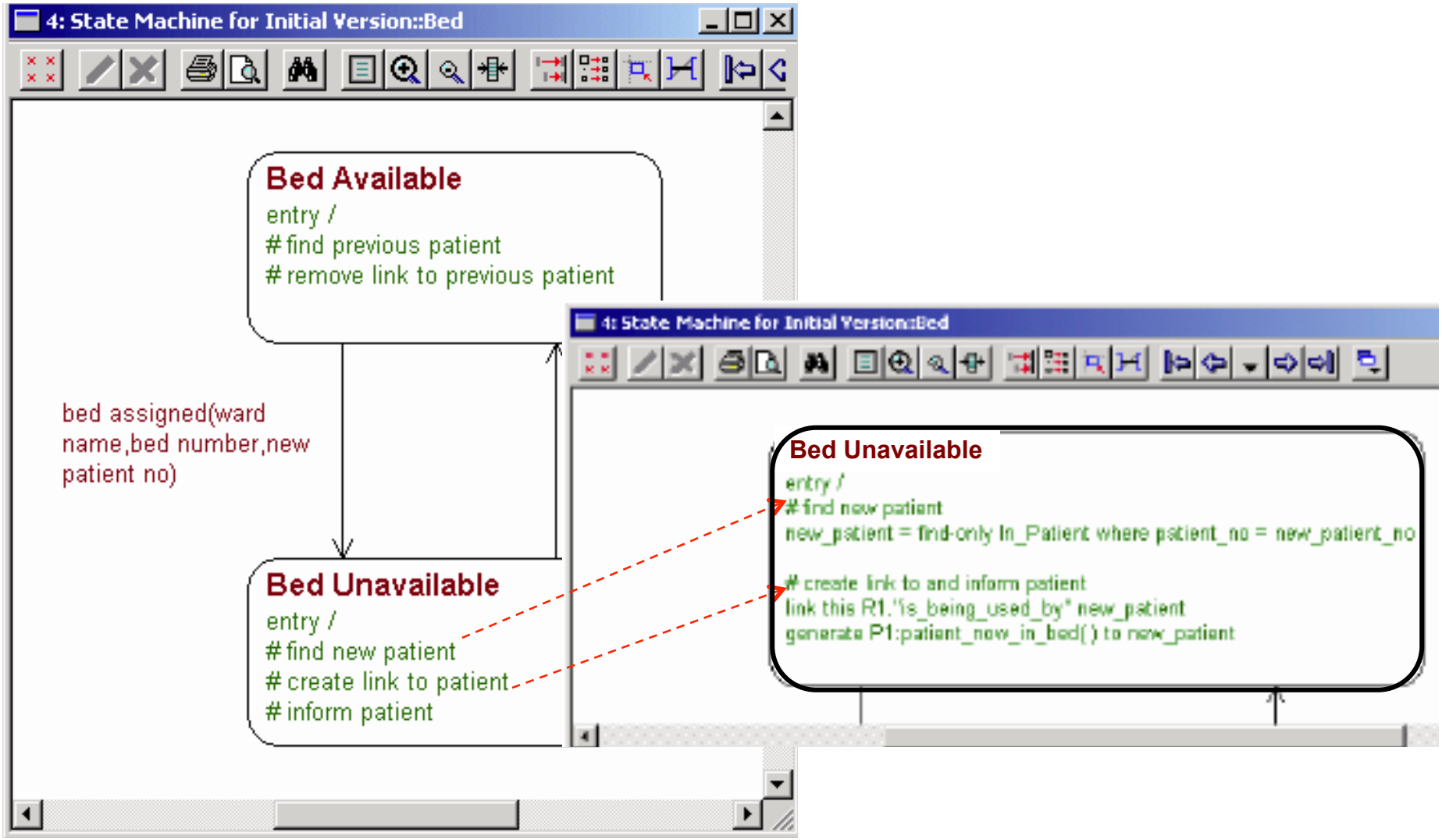
Action Semantics Example

- Activity Diagram with notations for various actions
  - More detailed expression of actions in process
- Could also use Interaction Diagrams, State Machine Diagrams, Pseudo-code, and the like

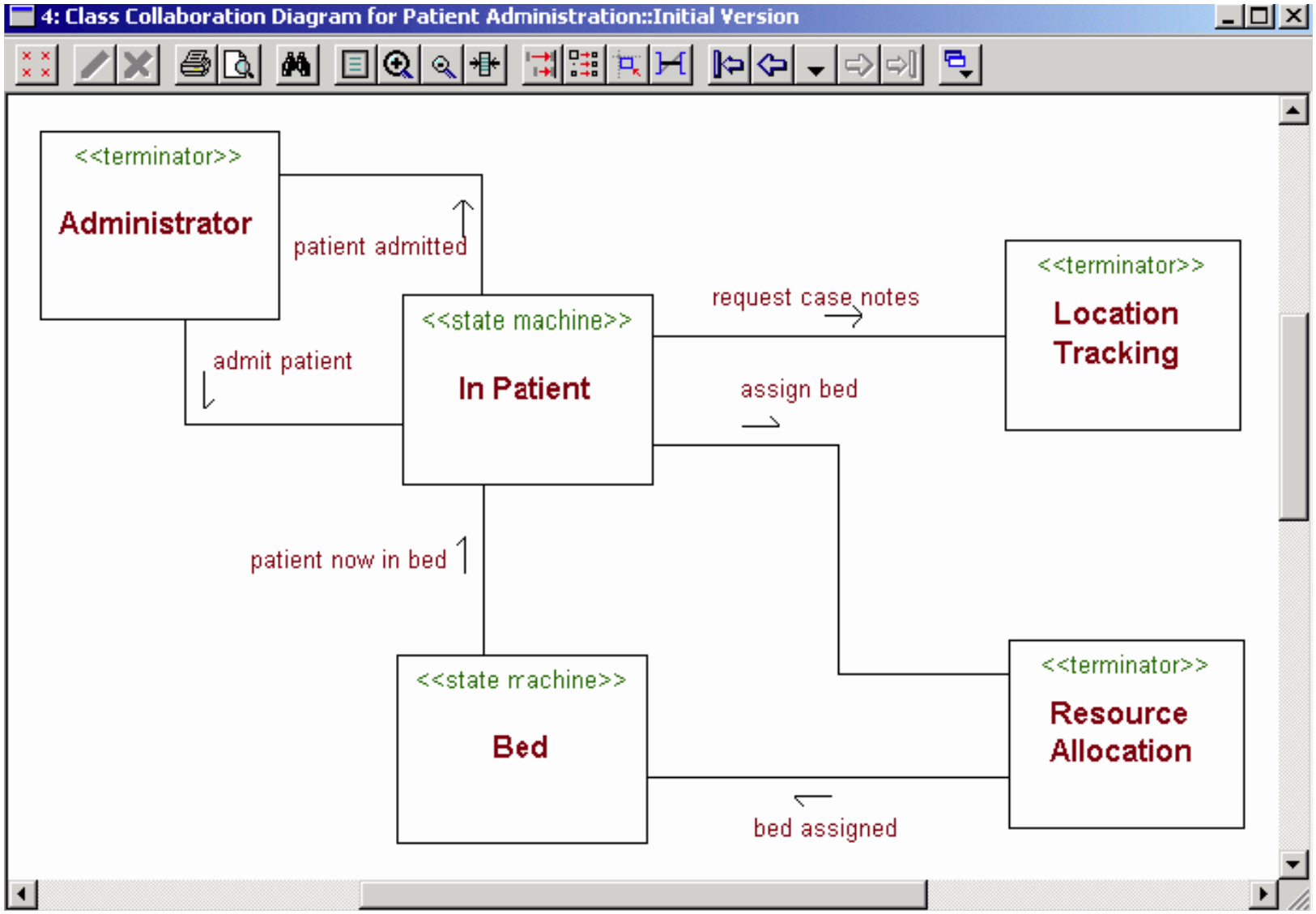
Recall: Domain Model Example



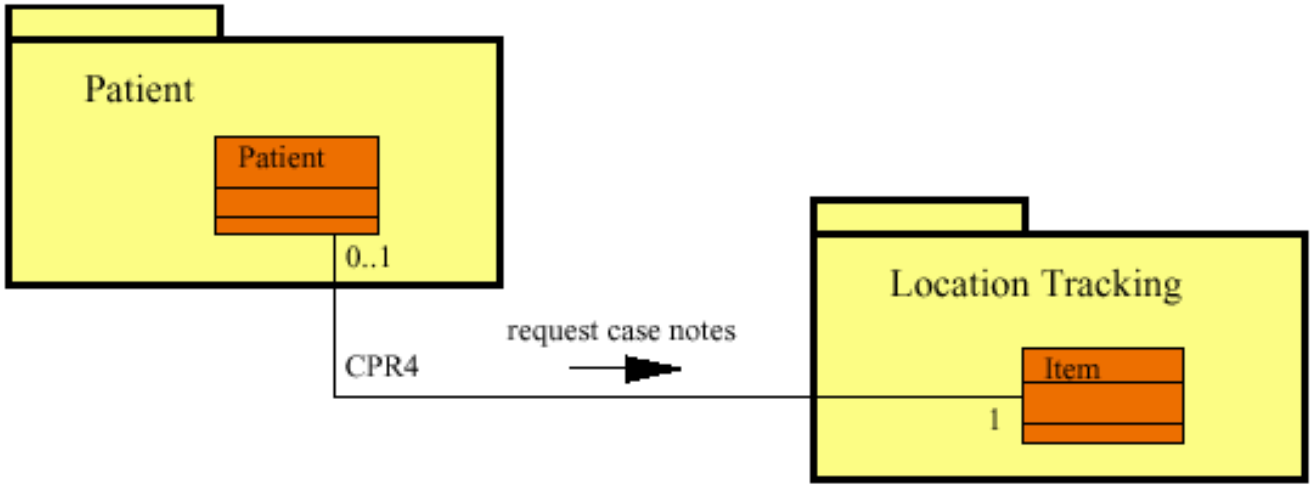
Map to Lower Levels



Platform Independent Model (PIM)



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
```

Some Open Source Transformers

- Generative Model Transformer (GMT)
  - http://www.eclipse.org/gmt
  - Eclipse project *(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

What is a Product?

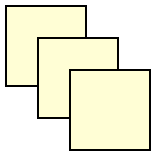
What is a Product Line?

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

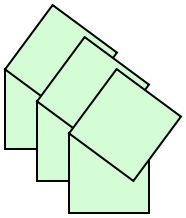
Product Line Philosophy

- 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

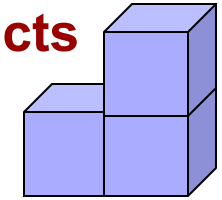
Use of a common asset base



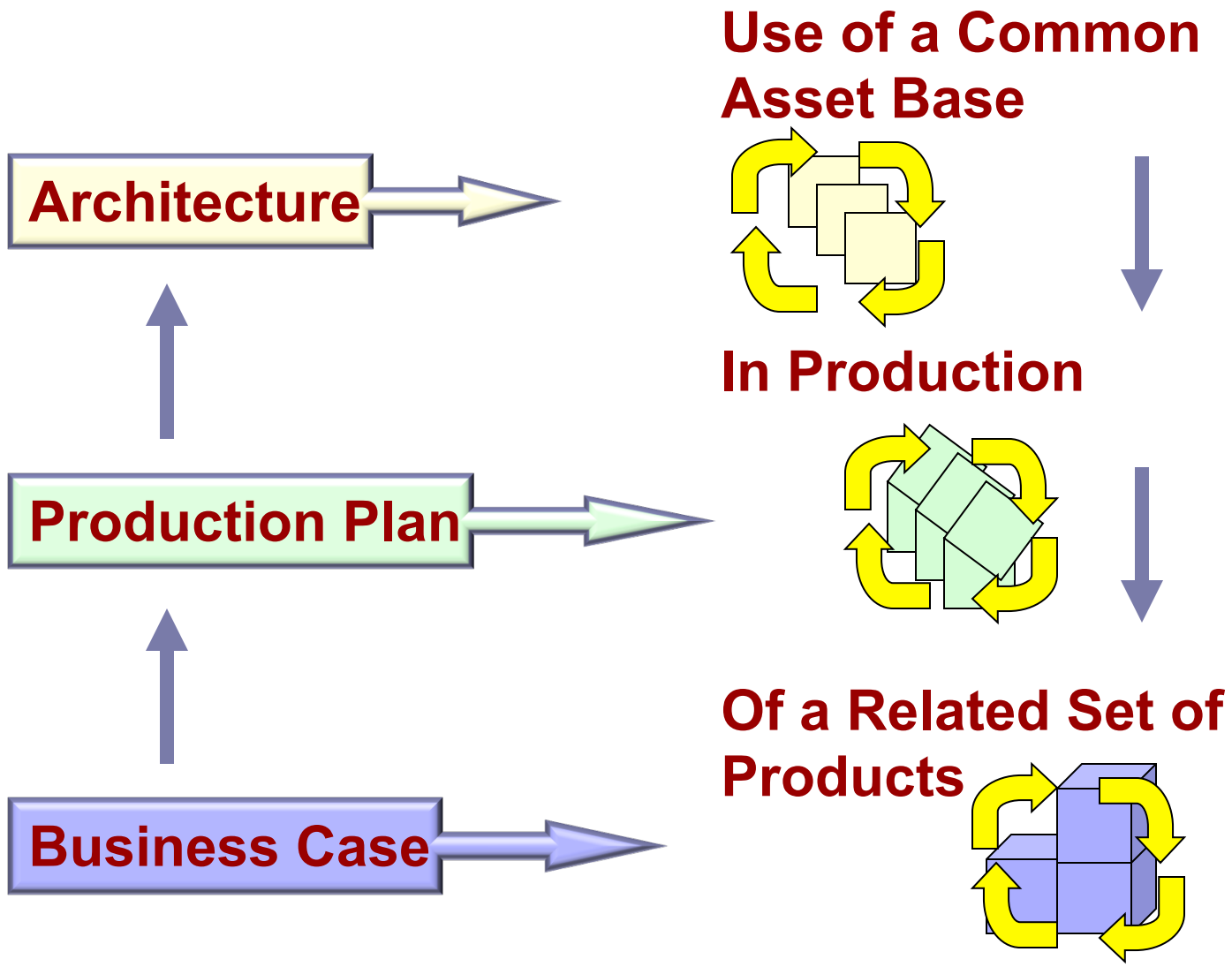
In production



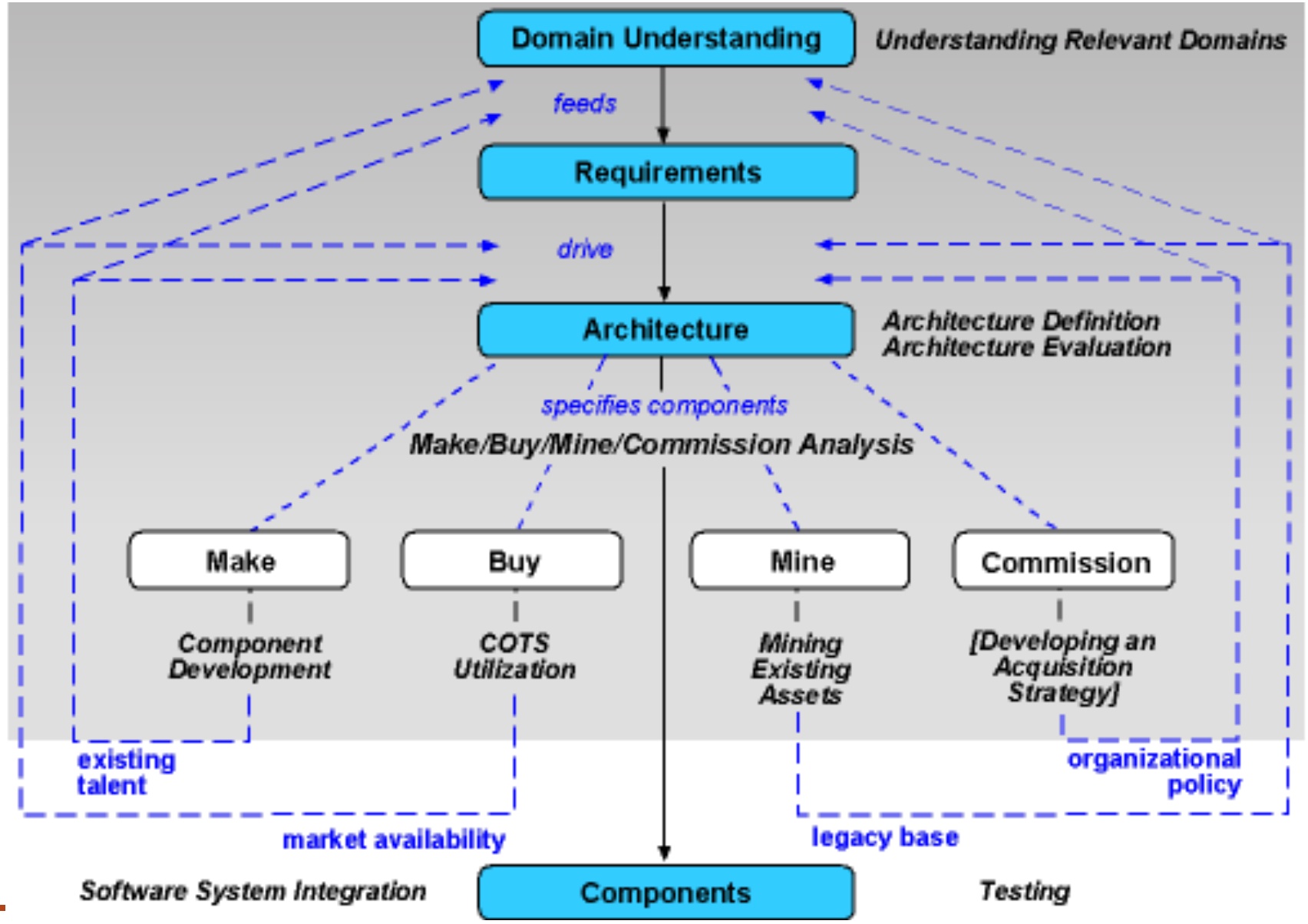
Of a related set of products



Key Product Line Concepts

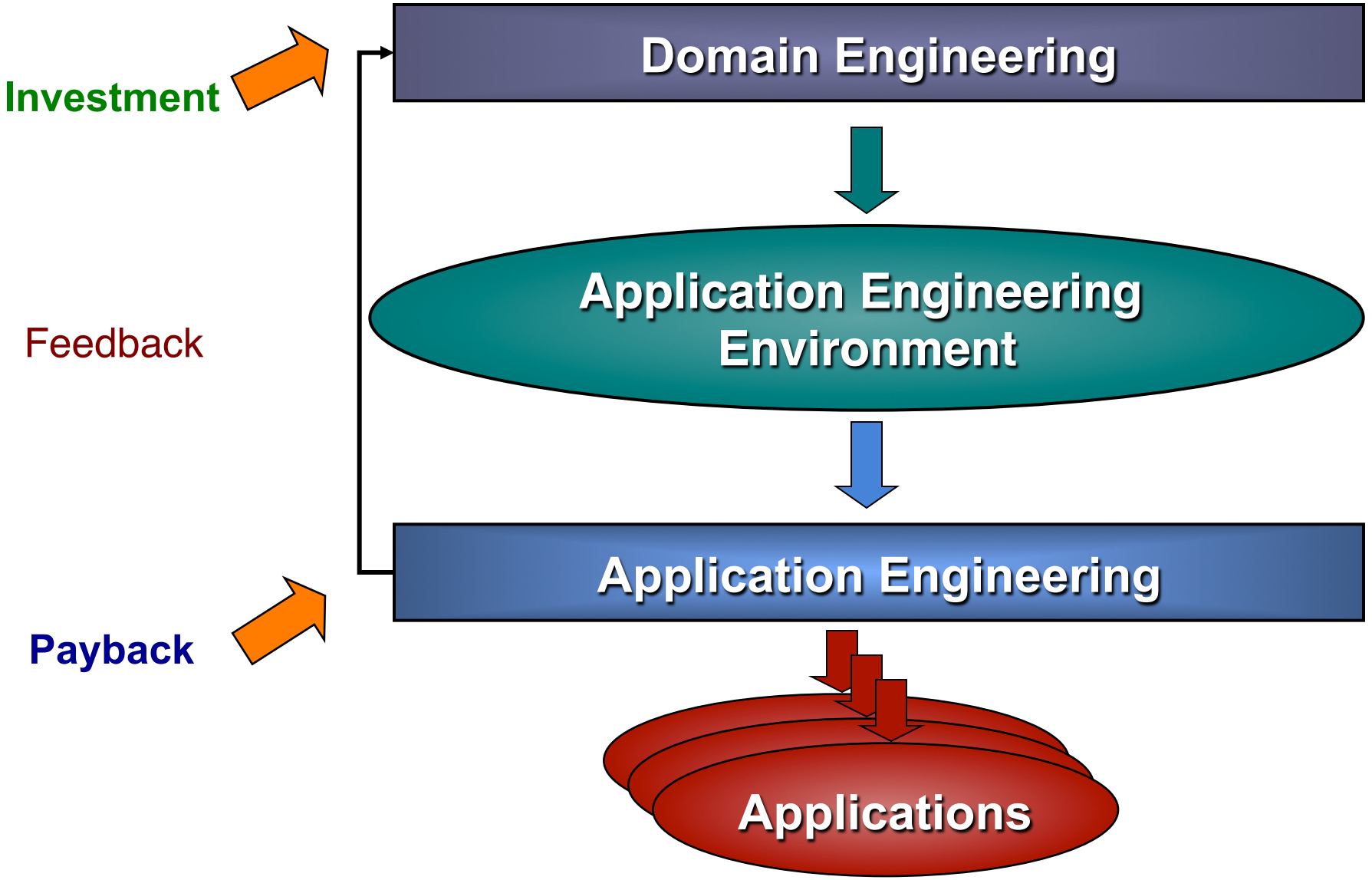


Ecosystem: Key Product Line Activities

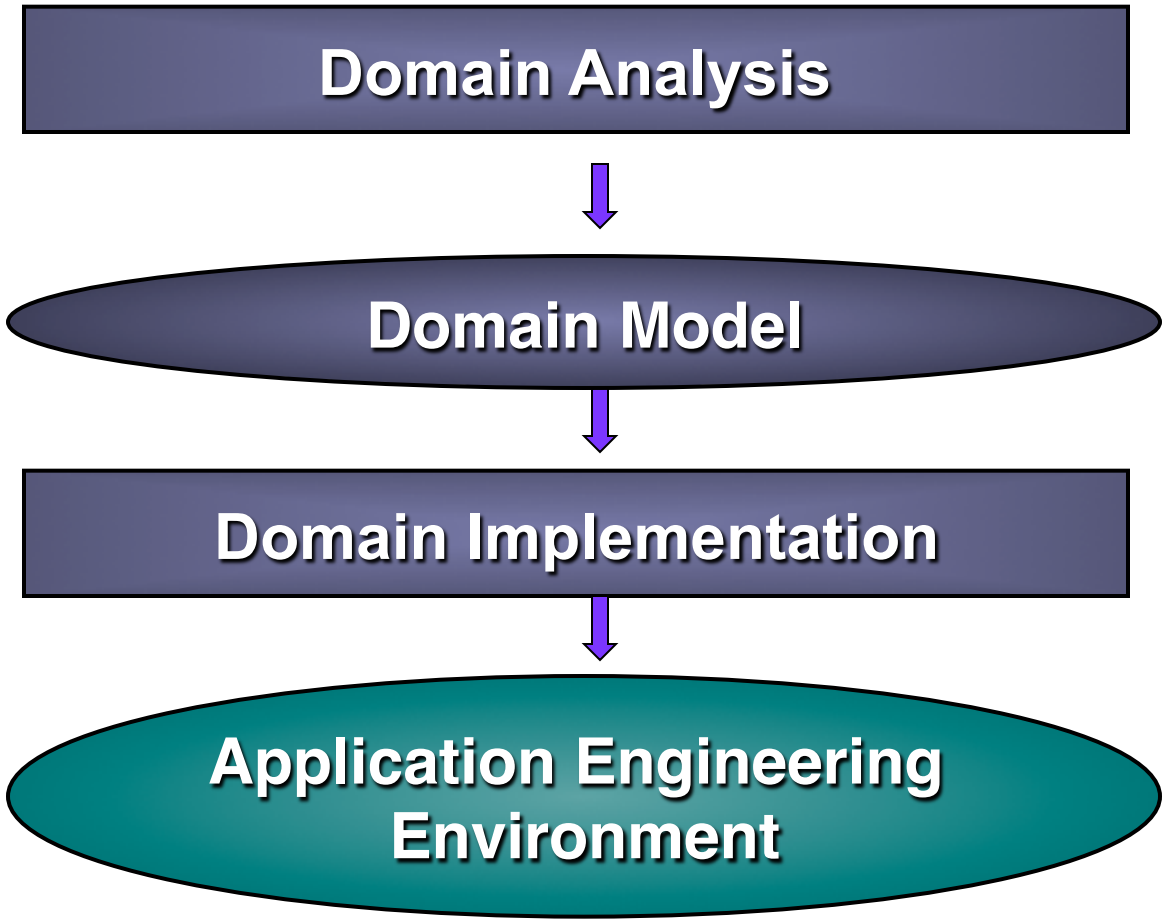


Source: Software Engineering Institute

Recall: A Product Family 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
  - How to identify a family member?
- Define Family of Products
  - What do family members of have in common and how do they vary?
- Design Application Modeling Language
  - What is a good way to model a family member?
- Design Application Engineering Environment
  - What are good mechanisms for using the decision model and the Application Modeling Language?

Defining Family: Commonality Analysis

- Dictionary of terms
  - Terms that define a domain vocabulary
- Commonalities: Assumptions that hold for every member of the family
  - E.g., Every unit must be in 1 of the 4 primary conditions
- Variabilities: Assumptions that define the range of variation for the family
  - E.g., Some unit names have inhibit states
- Parameters of Variation: Quantification of the variabilities
  - E.g., Whether or not a unit name can have an inhibit state: Boolean

Reusable Assets

- Validations
  - E.g., checkers for unit types
- Realizations
  - E.g., generic algorithms for every unit type
- Relationships
  - E.g., data that is used to drive the generic algorithms
  - E.g., design information shared across development

Homework and Milestone Reminders

- Read Chapter 8 on Domain Architectures
- Term Paper Proposal (see Homeowork assignment)
  - Select topic (justify if not in list)
  - Provide a descriptive Title
  - Provide a short description of your topic
    - Provide a "going in position" or "stand" regarding what you want to convey or learn.
    - Outline the major controversial issues or trade-offs inherent in the topic (bullets are acceptable).
  - The proposal should be no longer than one page (not including cover page ☺).
  - Due by 9:00am Monday Morning, April 25th, 2011.
- Milestone 3: Early Transformation Environment (see Milestone 3 assignment)
  - Due by 11:55pm, Friday, April 29th, 2011.