CSSE 490 Model-Based Software Engineering: Introduction to Transformation

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Sometimes, there is uncertain logic!

Logic: another thing that penguins aren't very good at.

Learning Outcomes: MBE Discipline

Relate Model-Based Engineering as an engineering discipline.

- Outline Formal Method approaches
- Discuss underlying representations
- Introduce how formal approaches used in MBSE

Reading Assignment Discussion

- "Capturing and Using Software Architecture Knowledge for Architecture-Based Software Development" by Babar et. al.
- Some questions to answer for the paper:
  - What is the thrust or message of the paper?
  - Why is implicit architectural knowledge important for Model-Based Software Engineering?
  - How could you structure a MBSE repository to handle hard/explicit information and soft/implicit information?
  - How could you apply these concepts to the other areas of software artifacts?

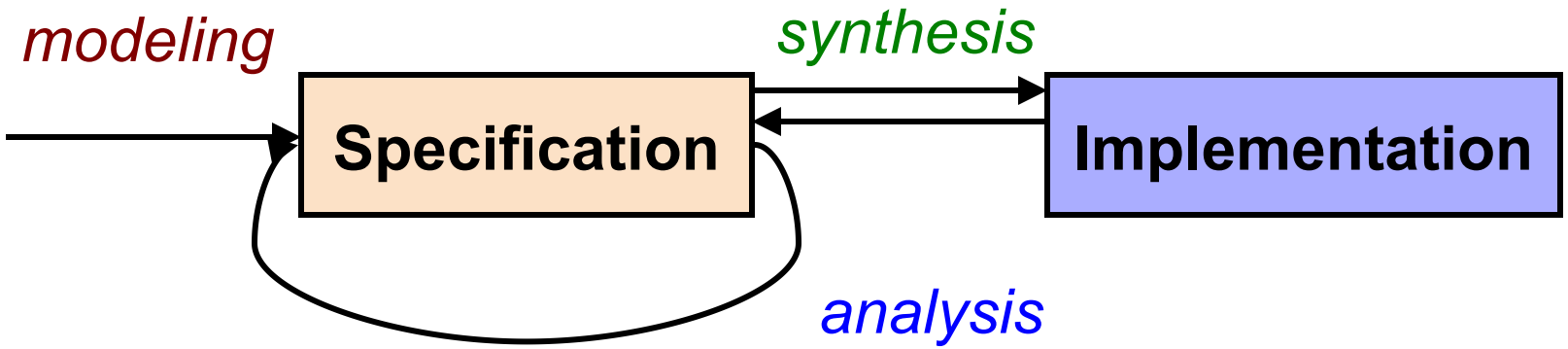
What are some of properties of a formal representation form?

How do they support the process of generating software?

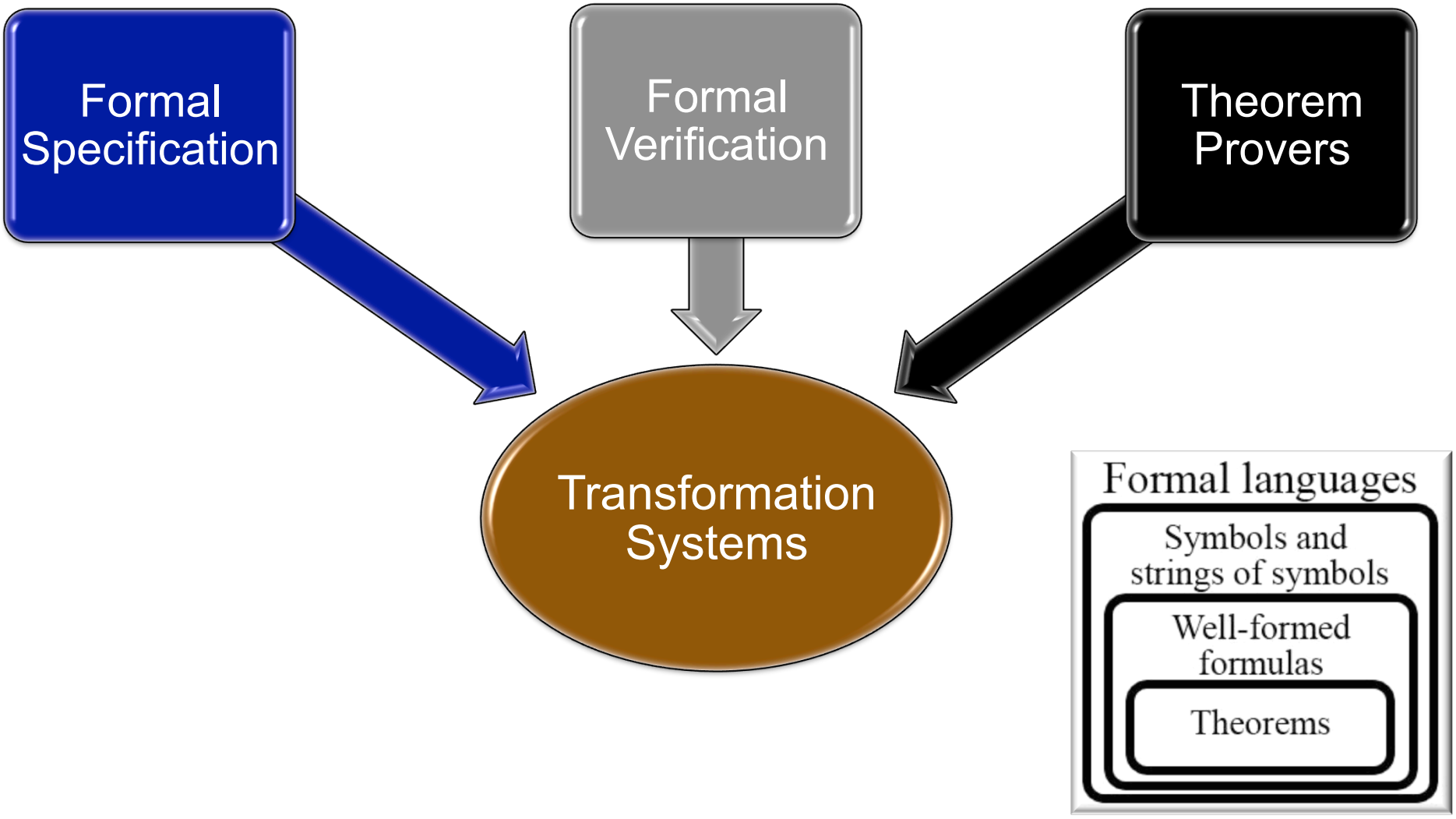
- Think for a minute…
- Turn to a neighbor and discuss it for a minute

Formal Methods

- Formal => Unambiguous, Consistent, Correct
- Formal Method =
  Specification Language + Formal Reasoning
  - Set of techniques supported by precise mathematics and powerful reasoning tools
- Rigorous mechanisms for system
  - Modeling, Synthesis, and Analysis

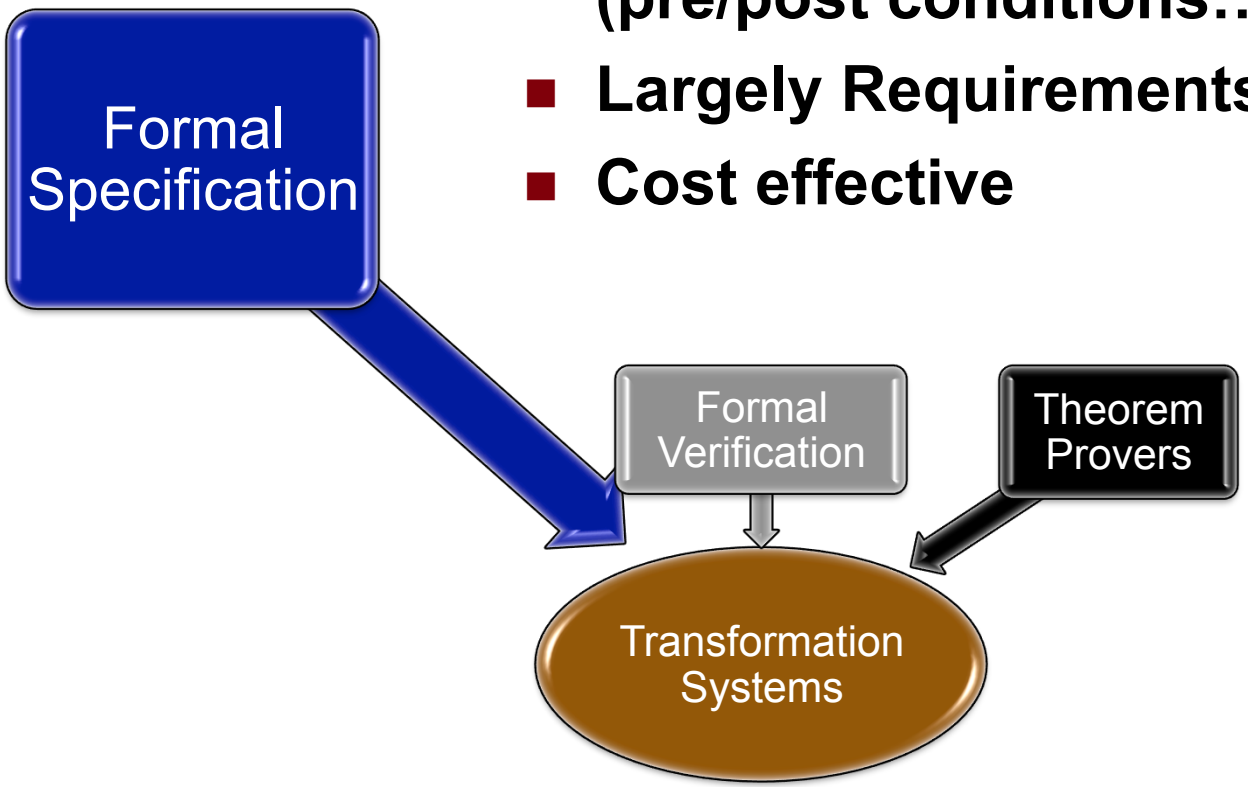


Levels of Formality



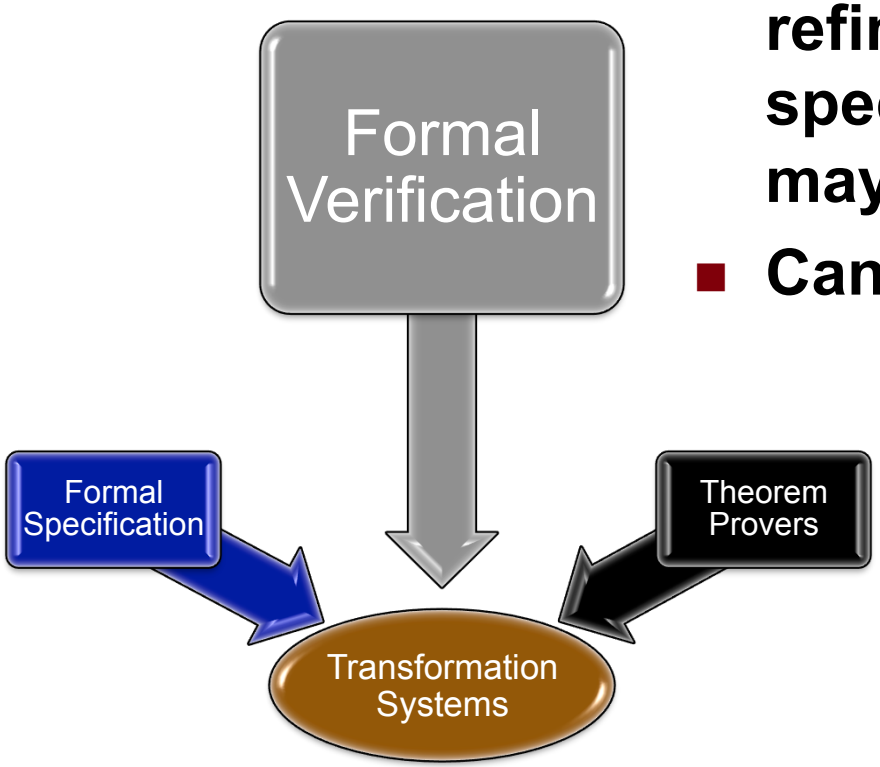
Formal Specification

- Formal Methods "lite" (pre/post conditions…)
- Largely Requirements Specification
- Cost effective



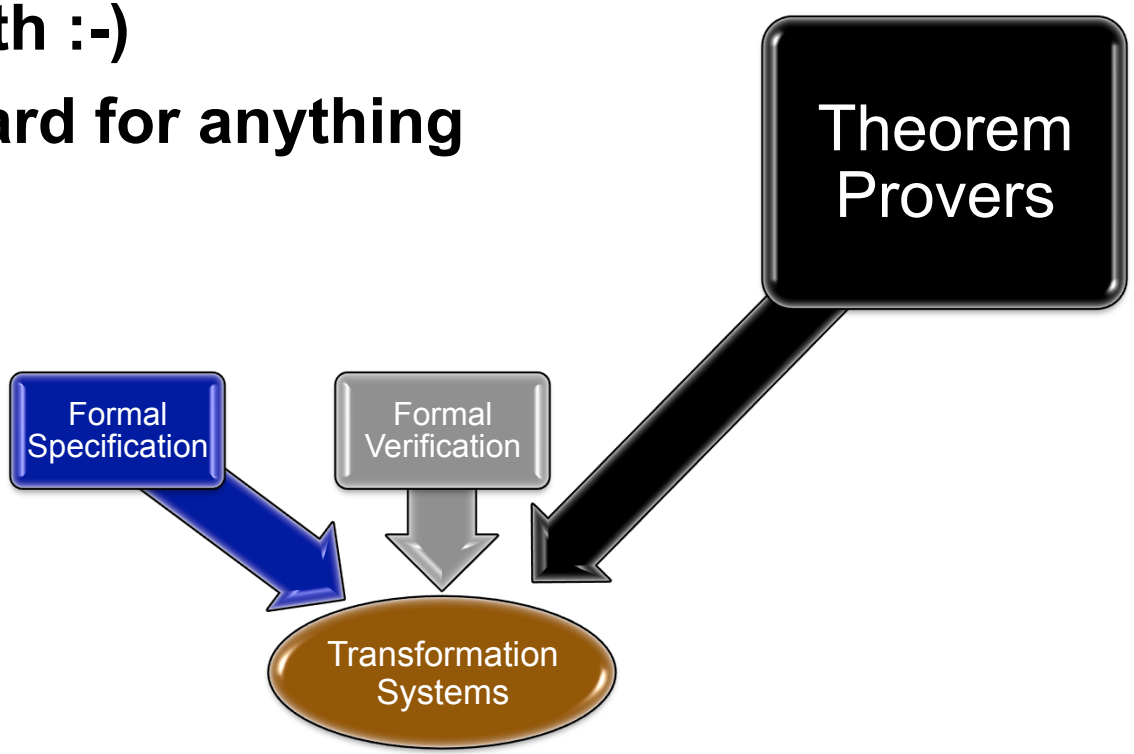
Formal Verification

- Produce program in more formal manner
- Proofs of properties or refinement from the specification to a program may be undertaken
- Can be costly



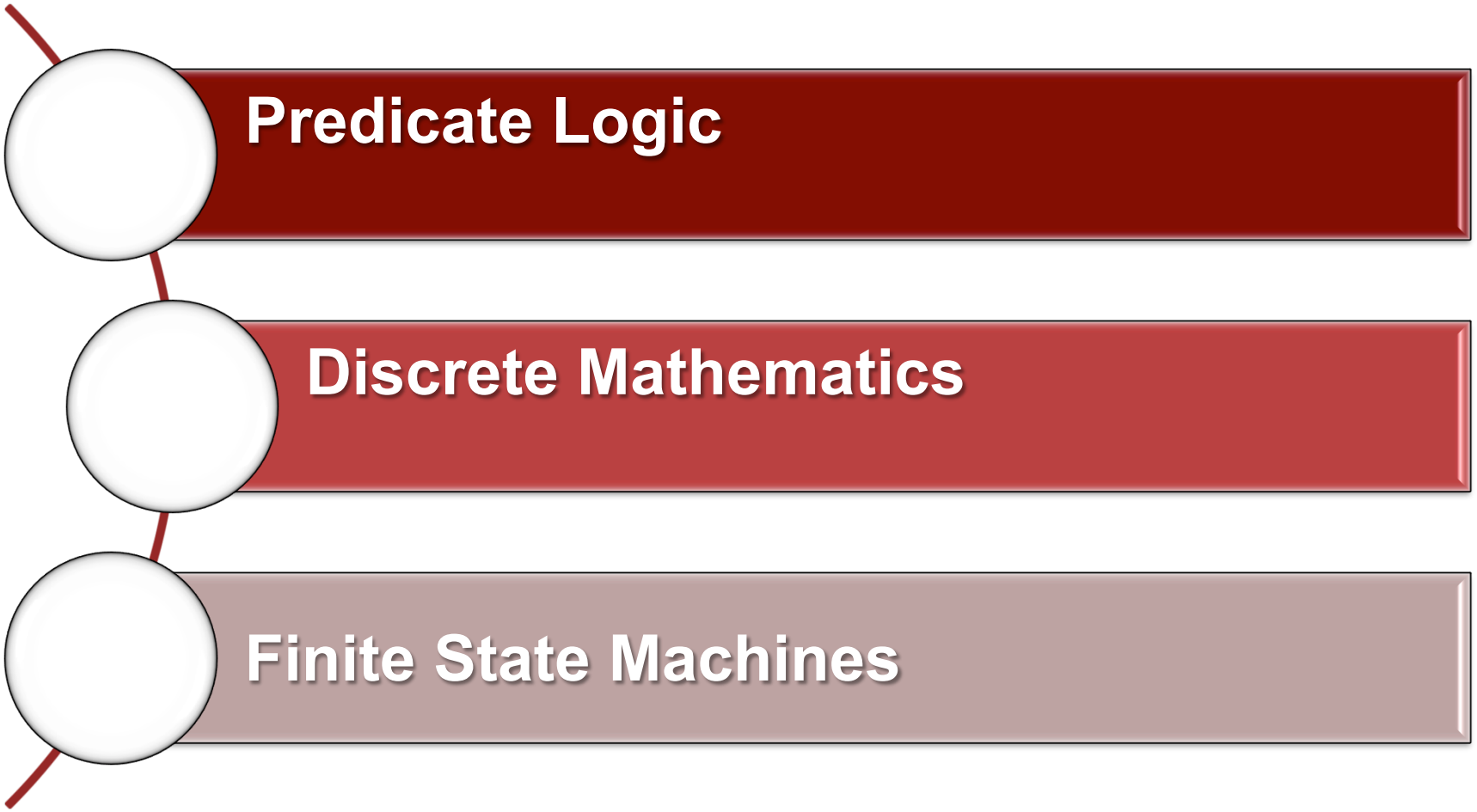
Theorem Provers

- Formal machine-checked proofs
- Gotta like the math :-)
- Expensive and hard for anything but small scope



More-on Provers?

Basic Types of Formalisms



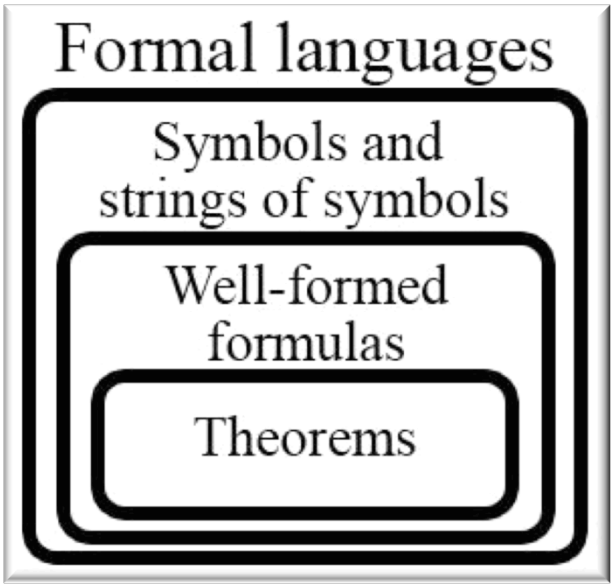
Formal Methods Applicability in Software

- System models
- Constraints
- Requirements specifications
- Designs
- Automated implementation
- Model-Based Software Engineering

Formal Specification Languages 1/2

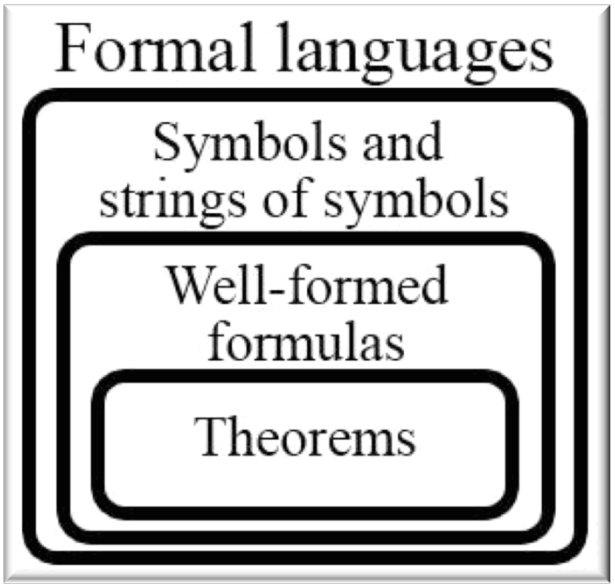
- **Axiomatic** – operations defined by logical assertions
  - e.g., Anna, Esterel, Lustre, RAISE, Object Constraint Language (OCL)

- **State transition** – operations defined in terms of computational states and transitions
  - e.g., Abstract State Machines (ASMs), State Charts, Petri-Nets, Data Flow



Formal Specification Languages 2/2

- **Algebraic** – operations defined by equivalence relations
  - e.g., Common Algebraic Specification Language (CASL), Larch, OBJ, LOTOS, π-calculus
- **Abstract model** – operations defined in terms of a well-defined mathematical model
  - e.g., QVT, Alloy, B, Z, VDM++, Communicating Sequential Processes (CSP), Rebeca Modeling Language

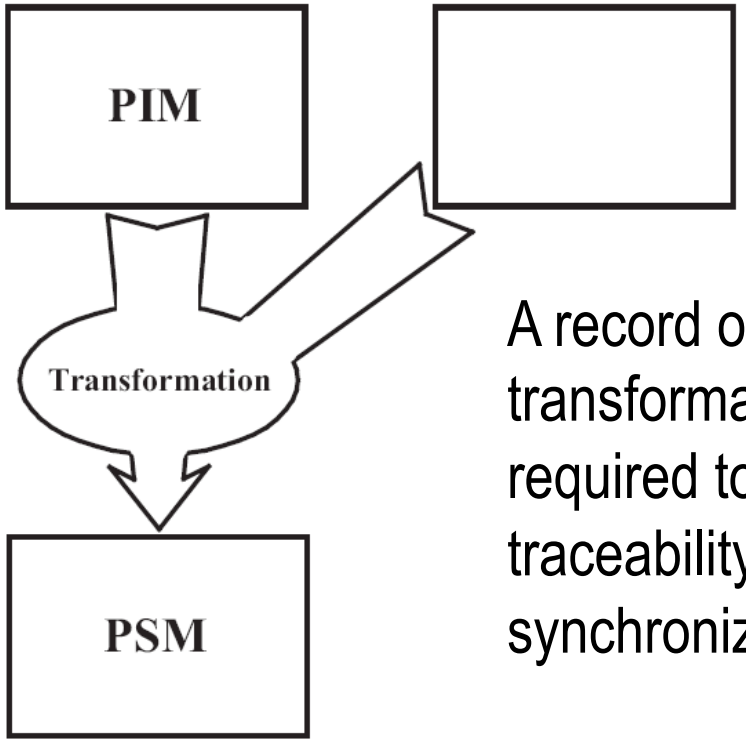


What Comprises MDA?

- MDA is not a single specification, but a collection of related OMG specifications:
  - Unified Modeling Language (UML™) 2.0
    - Infrastructure
    - Superstructure
    - Object Constraint Language (OCL)
    - Diagram Interchange
    - Profiles
  - Meta-Object Facility (MOF)
  - XML Meta-Data Interchange (XMI)
  - Common Warehouse Meta-model (CWM)
  - Query View Transformation (QVT)

Transformations

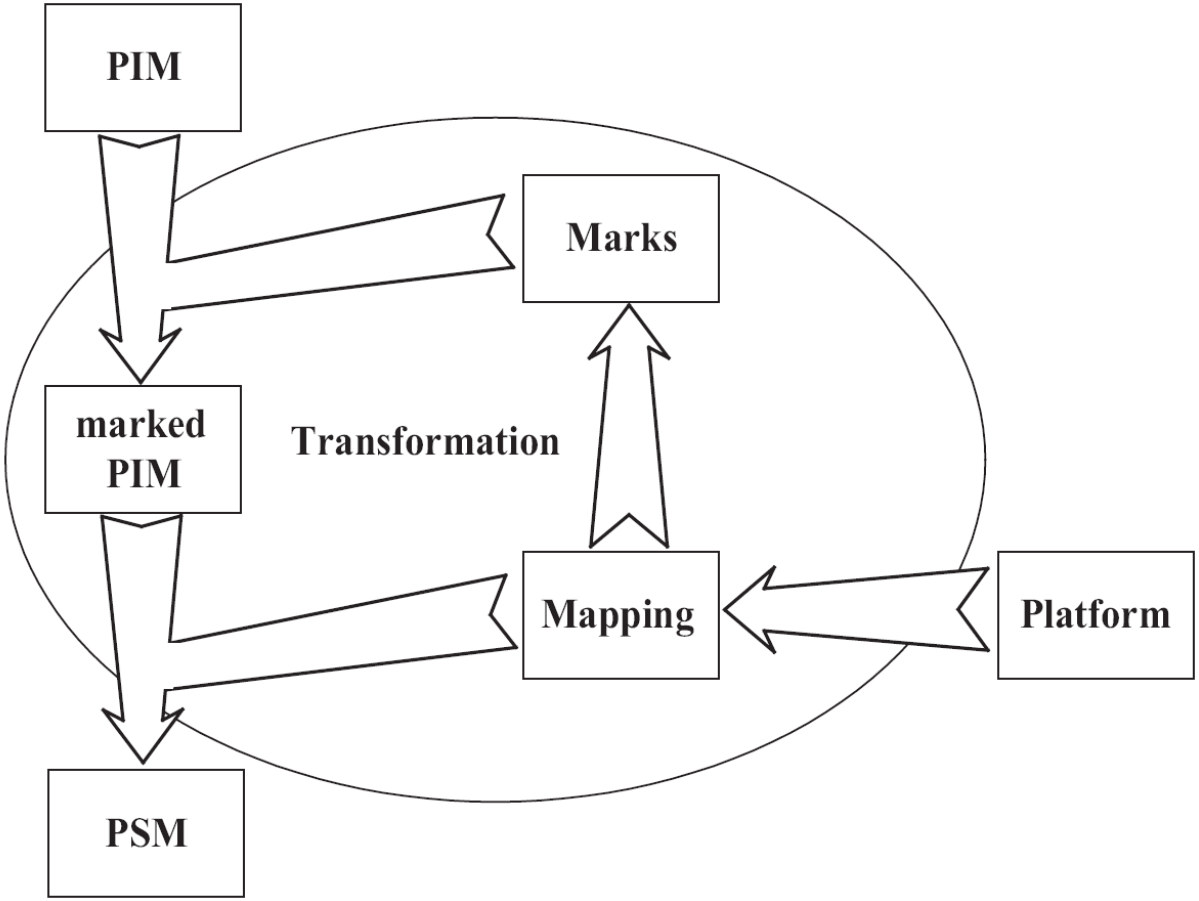
- This is the challenge for MDA!
- Query / View / Transformation (QVT) is the answer



A record of transformation is required to maintain traceability and synchronization.

Model Marking Transformation

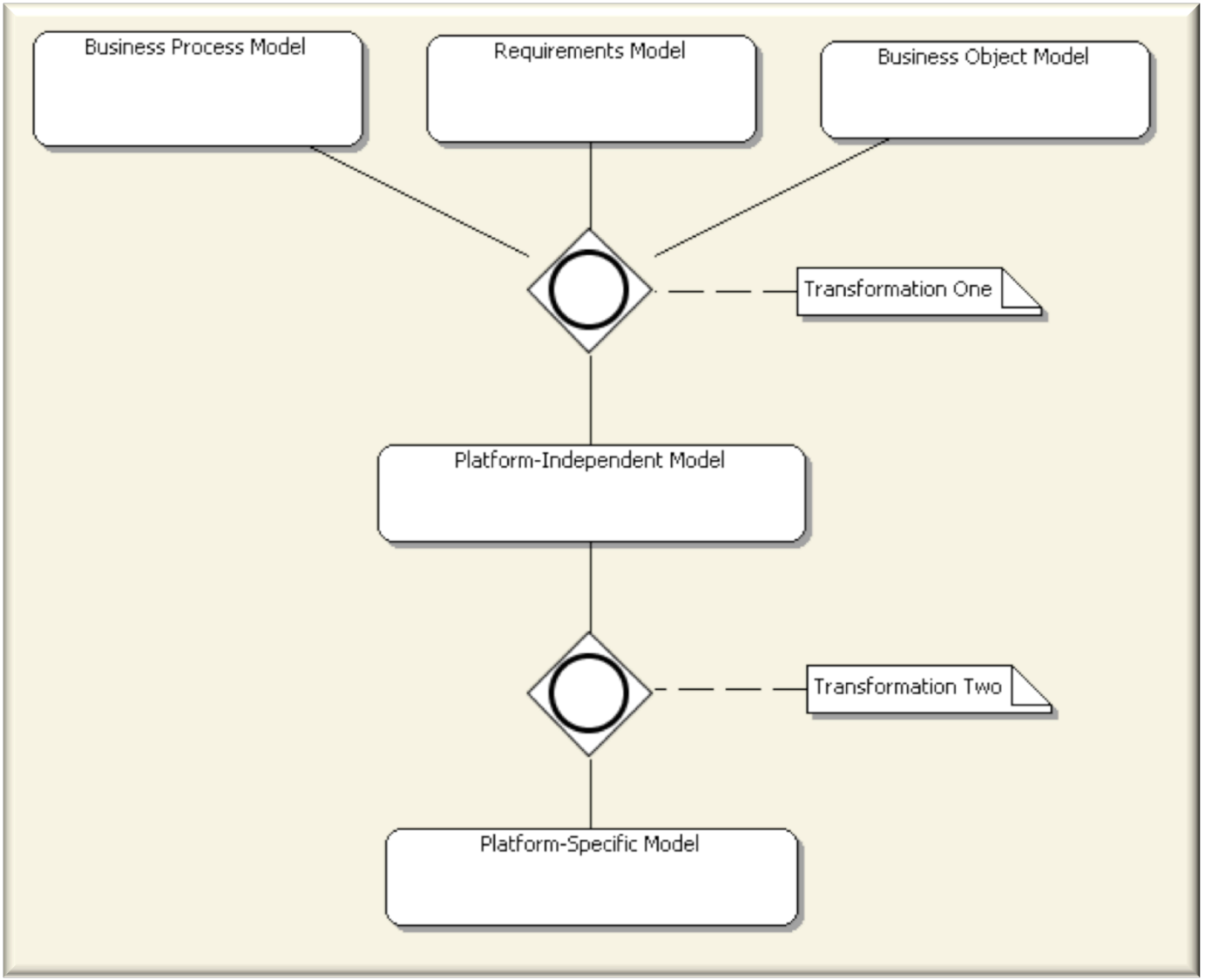
- Marks are specific to a transformation
  - Can be a stereotype in a profile, for example



Query-View-Transformation

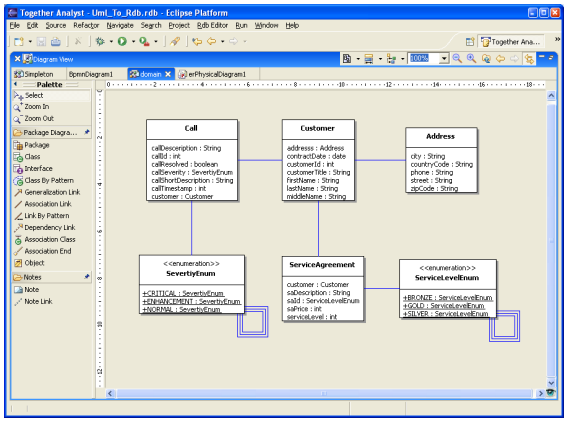
- QVT specification is the heart of Model Driven approaches
- Queries take a model as input and select specific elements from that model
- Views are models that are derived from other models
- Transformations take a model as input and update it or create a new model

Example Transformation

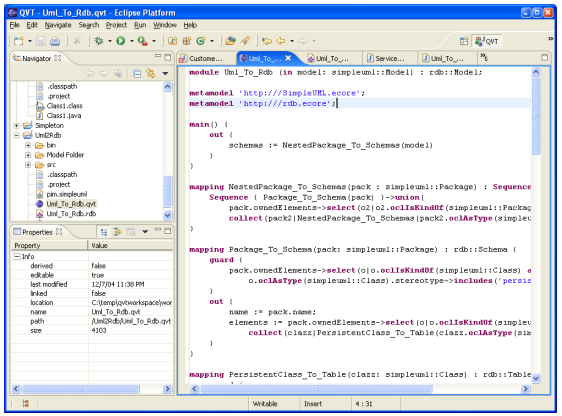


UML to RDB Example

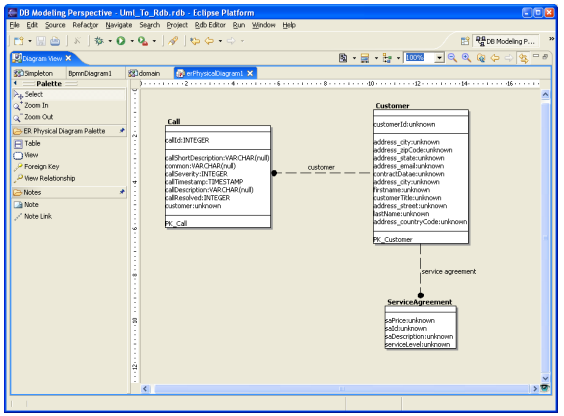
- UML Class model → Relational Data Model



Model



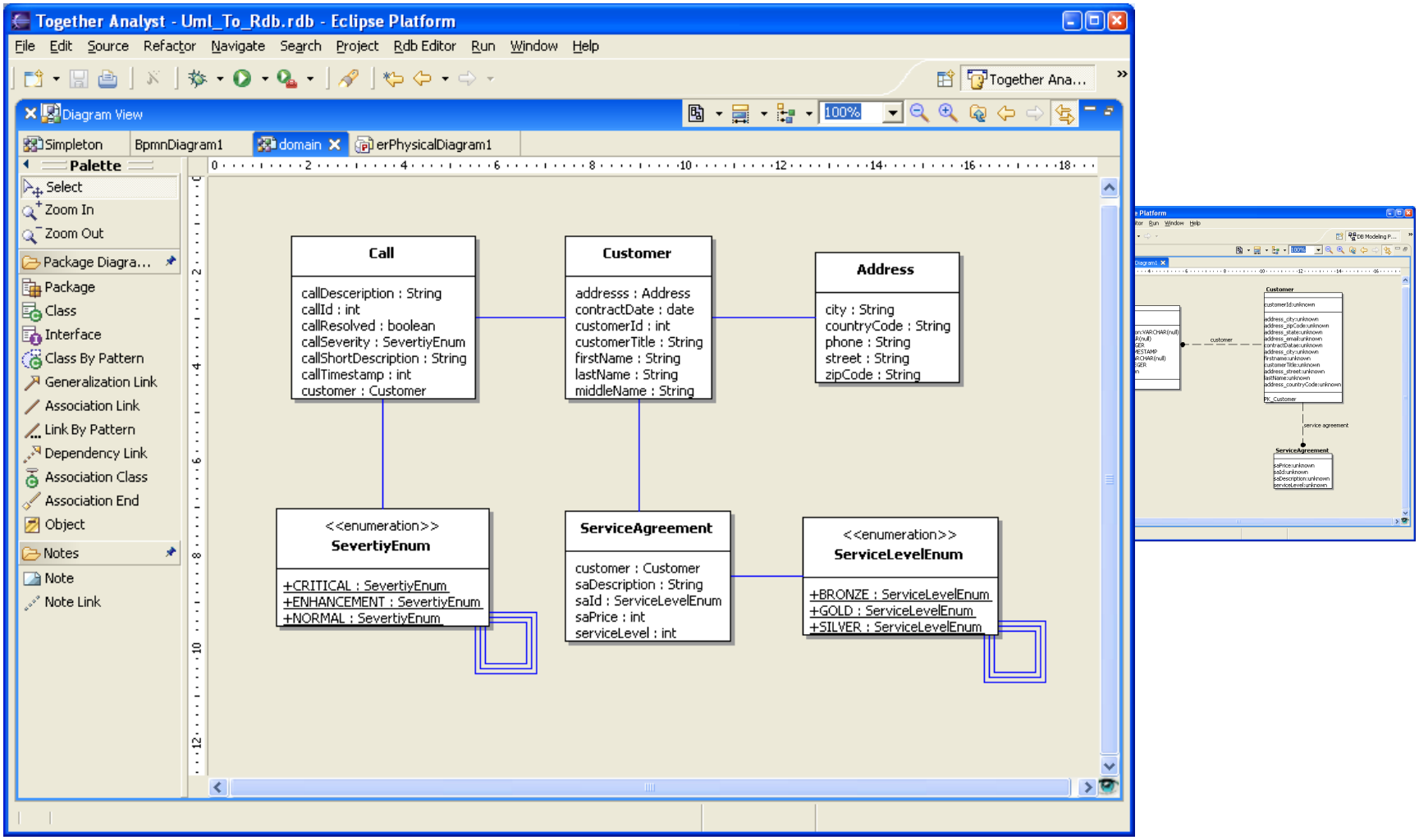
Query



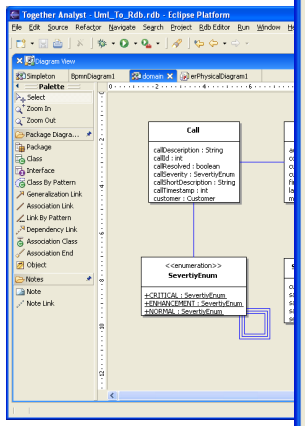
View

transformation

UML to RDB Example: Model



UML to RDB Example: Query



QVT - Uml_To_Rdb.qvt - Eclipse Platform

File Edit Navigate Search Project Run Window Help

Navigator

.classpath
.project
Class1.class
Class1.java
Simpleton
Uml2Rdb
bin
Model Folder
src
.classpath
.project
pim.simpleuml
Uml_To_Rdb.qvt
Uml_To_Rdb.rdb

Properties

Property	Value
Info	
derived	false
editable	true
last modified	12/7/04 11:38 PM
linked	false
location	C:\temp\qvtworkspace\wor
name	Uml_To_Rdb.qvt
path	/Uml2Rdb/Uml_To_Rdb.qvt
size	4103

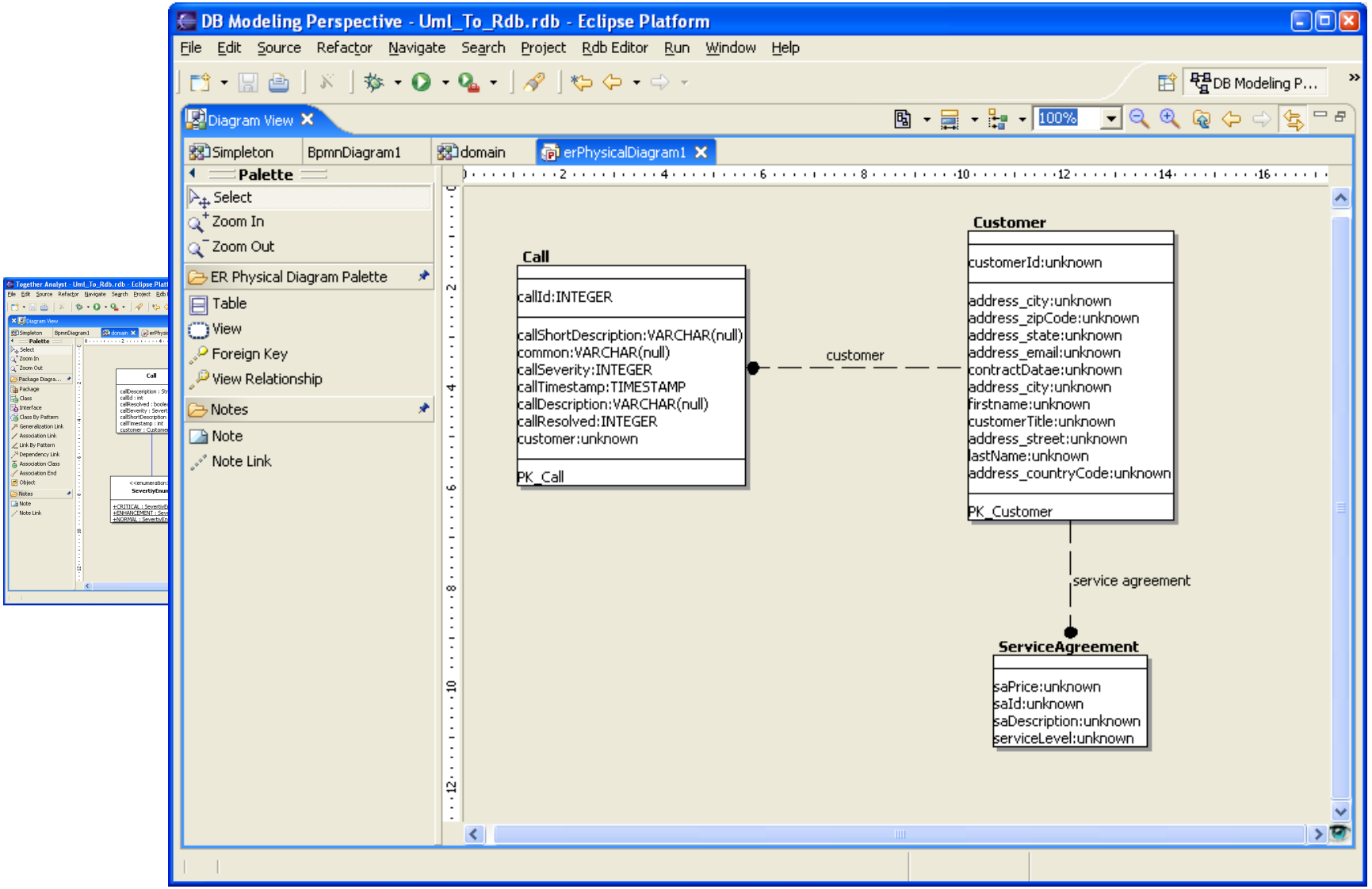
Custome... | Uml_To_... | Uml_To_... | Service... | Uml_To_...

```
module Uml_To_Rdb (in model: simpleuml::Model) : rdb::Model;

metamodel 'http:///SimpleUML.ecore';
metamodel 'http:///rdb.ecore';

main() {
    out {
        schemas := NestedPackage_To_Schemas(model)
    }
}

mapping NestedPackage_To_Schemas(pack : simpleuml::Package) : Sequence
    Sequence { Package_To_Schema(pack) }->union(
        pack.ownedElements->select(o2|o2.oclIsKindOf(simpleuml::Packag
        collect(pack2|NestedPackage_To_Schemas(pack2.oclAsType(simpleu
}

mapping Package_To_Schema(pack: simpleuml::Package) : rdb::Schema {
    guard {
        pack.ownedElements->select(o|o.oclIsKindOf(simpleuml::Class) a
            o.oclAsType(simpleuml::Class).stereotype->includes('persis
    }
    out {
        name := pack.name;
        elements := pack.ownedElements->select(o|o.oclIsKindOf(simpleu
            collect(clazz|PersistentClass_To_Table(clazz.oclAsType(sim
    }
}

mapping PersistentClass_To_Table(clazz: simpleuml::Class) : rdb::Table
```

Writable | Insert | 4 : 31

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

UML to RDB Example: View



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Homework and Milestone Reminders

- Read Chapter 6 on Metamodeling
- Let's talk tomorrow on your project