Towards Multi-level Aware Model Transformations

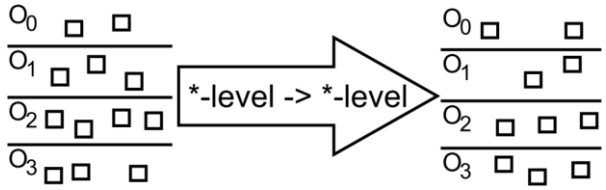
Colin Atkinson, Ralph Gerbig and Christian Tunjic

Agenda

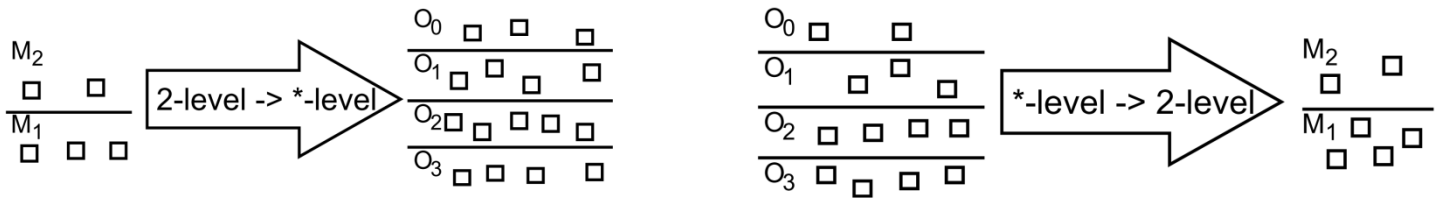
- **Motivation & Target**
- Differences between Multi-level and OMG/UML Infrastructure (2-level Modeling) based modeling
- Multi-level Aware Transformations
- Case Study

Motivation

- Model transformations are a central topic in model-driven development
  - e.g. exchange of models between tools, refactoring



- Interoperability between Multi-level modeling tools and OMG/UML infrastructure based tools



Target

- Support model transformations for multi-level modeling
- Gain interoperability with existing model-driven tools
- Do not reinvent the wheel
  - Keep learning curve for new users low

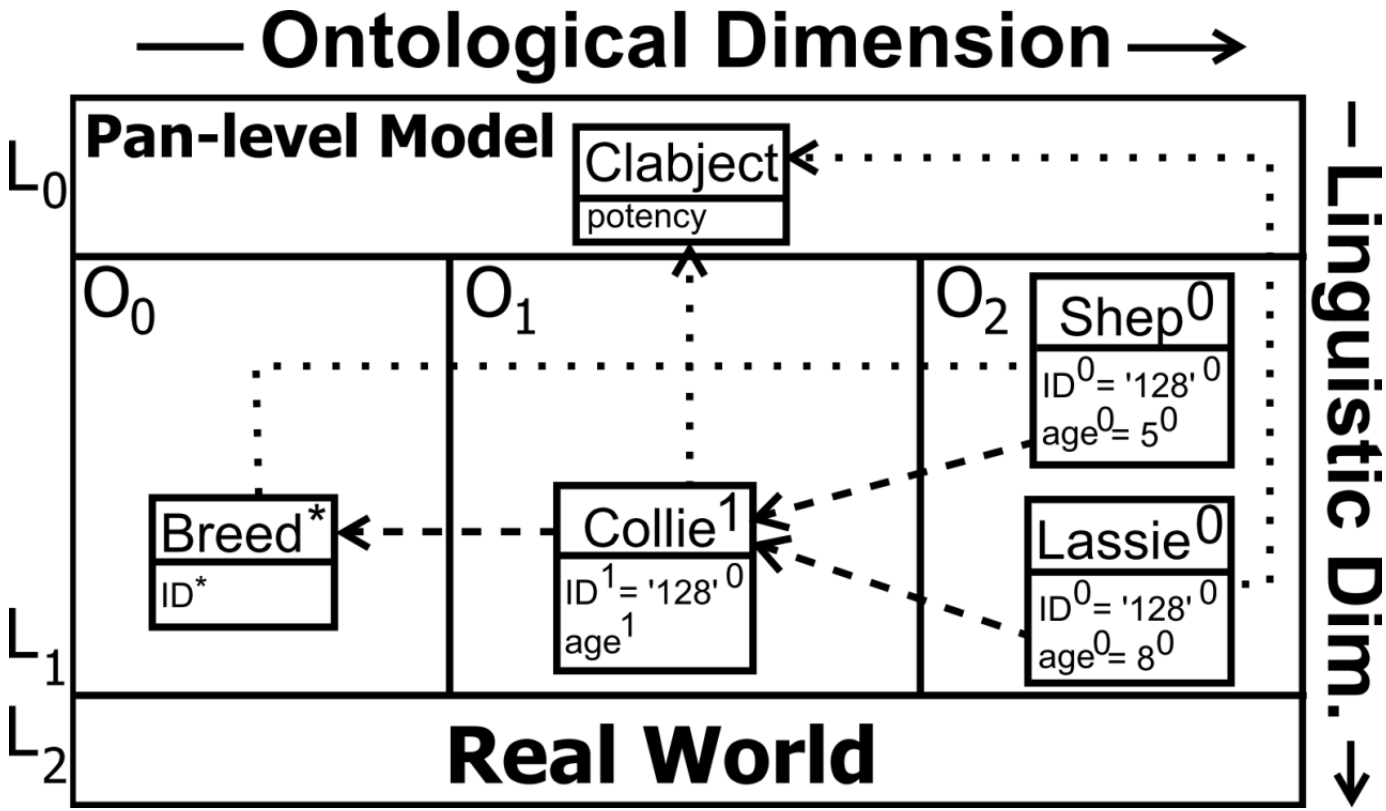
➔Extend an existing transformation technology with support for multi-level modeling

Agenda

- Motivation & Target
- **Differences between Multi-level and OMG/UML Infrastructure (2-level) based modeling**
- Multi-level Aware Transformations
- Case Study

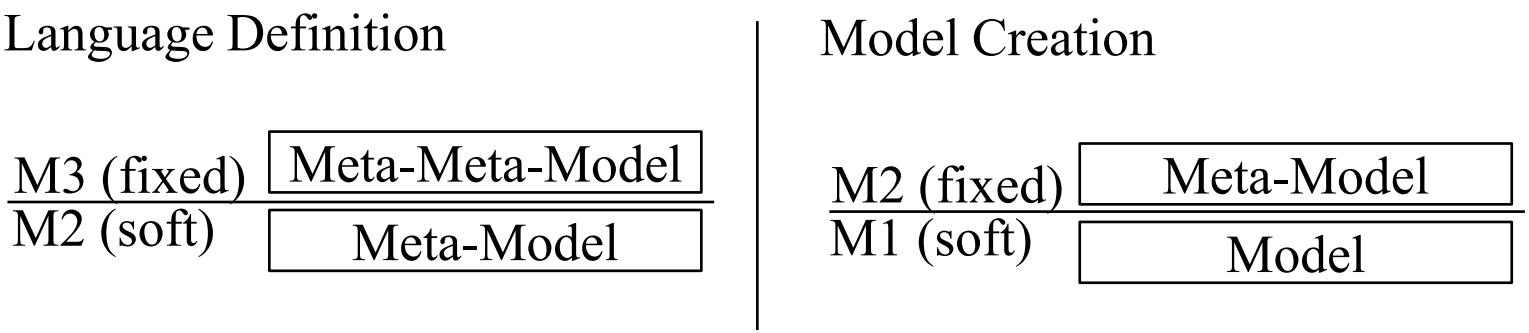
Multi-level Modeling

- Arbitrary number of classification levels
- Ontological and linguistic classification
  - Traits and Attributes
- Clabject
- Potency

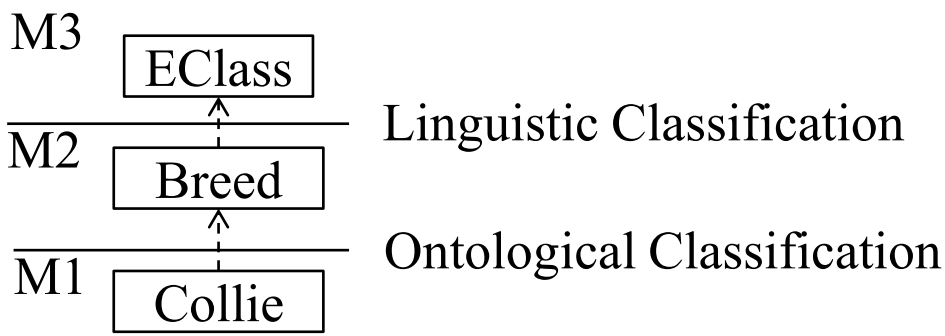


OMG/UML Infrastructure(2-level Modeling)

- Limited to two levels available at one time (thus here called 2-level models)



- One classification dimension



Impact of differences

- Ontological and linguistic classification vs. 1 classification dimension

➔ Extend language to support navigation between ontological and linguistic classification

- Arbitrary number of levels vs. fixed number of levels

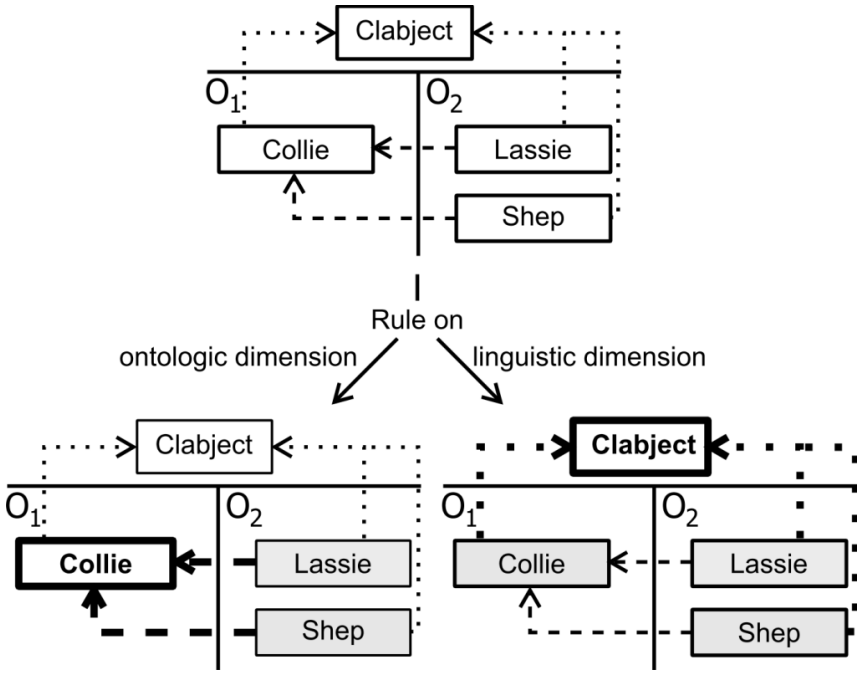
➔Handle an arbitrary number of levels during definition and execution of a transformation

Agenda

- Motivation & Target
- Differences between Multi-level and OMG/UML Infrastructure (2-level) based modeling
- **Multi-level Aware Transformations**
- Case Study

Handling Ont. and Ling. Classification

- Definition of rules on linguistic and ontological dimension

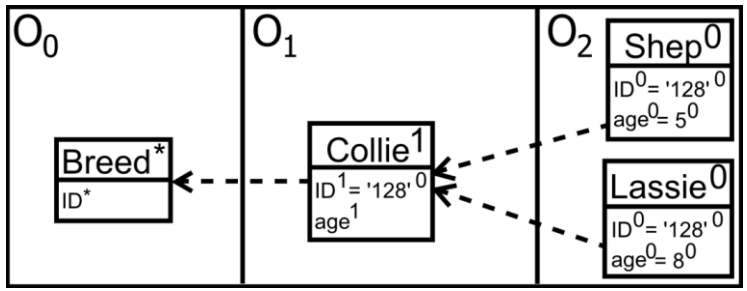



```
rule Collie2Class {
  from s : PLM!"O1::Collie"
  to t : UML!Class
    (name <- s._l_.name )
}
```

```
rule Clabject2Class {
  from s : PLM!Clabject
  to t : UML!Class
    (name <- s._l_.name )
}
```

Handling Multiple Model Levels

- Multi-level awareness
  - What to transform when defining rule on Breed?



  - Currently the instances at the deepest level are translated
  - "Potency" for rules needed? ➔ Further research

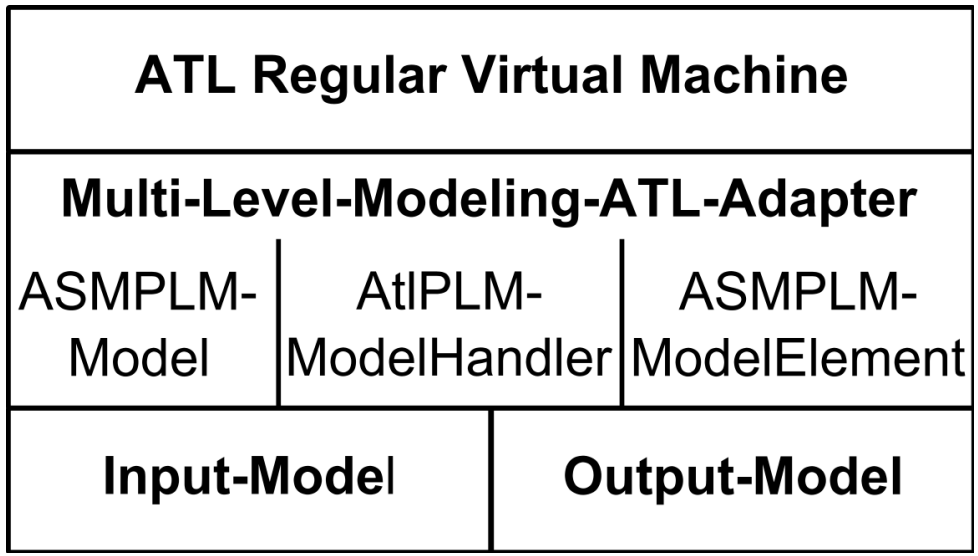
```
rule ComponentClass2Class {
  from s : PLM!"O1::Breed*"
  to t : UML!Class
    (name <- s._l_.name )
}
```

Multi-level aware ATL

- Motivations for using ATL
  - Good extensibility through adapter concept
  - Widely spread thus low learning curve for new useres
  - We have a tool making heavy use of ATL
    - Target stepwise migration from 2-level to multi-level models
- Features of ATL Adapter implementation
  - Syntax extension to support ontological and linguistic classification
  - Support for all three modes of transformations
    - 2-level to *-level, *-level to 2-level, *-level to *-level
  - Limitation(?): Instances at the lowest ontological-level are translated at the moment only
- Available at http://eclipselabs.org/p/melanie

Implementing Multi-level aware ATL

- ASMPLMModel, AtlPLMModelHandler and ASMPLMModelElement are extensions of the respective EMF-Adapter classes
- ASMPLMModel: Definition of rules on linguistic and ontological model elements
- ASMPLMModelElement: Access to linguistic and ontological attributes of model elements
- AtlPLMModelHandler: Loading of AMSPLMModels



Agenda

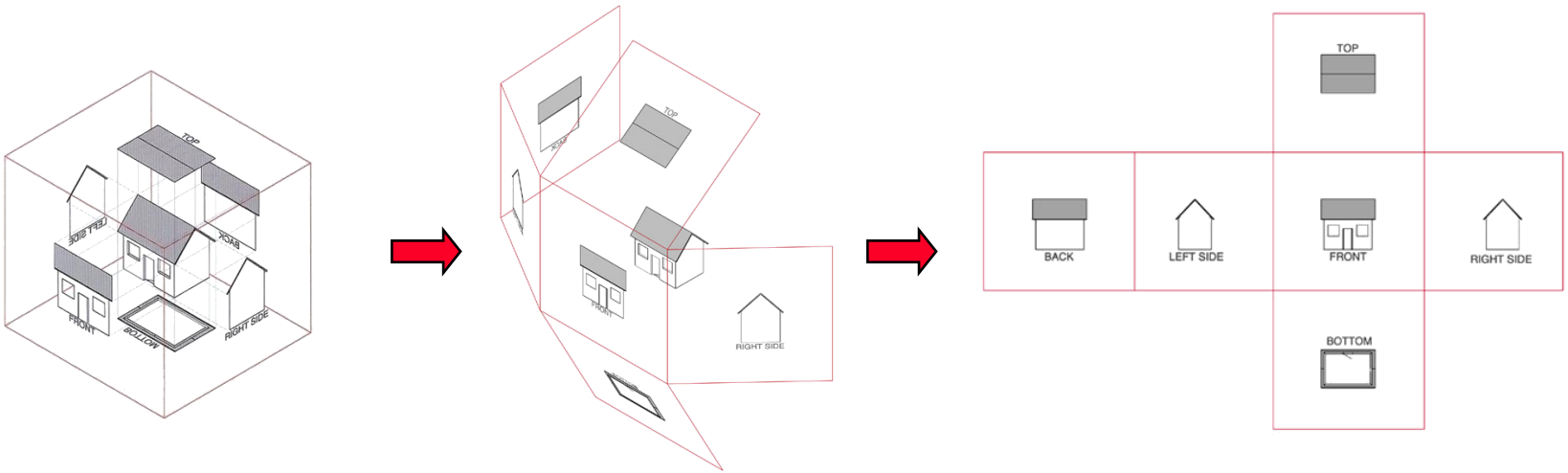
- Motivation & Target
- Differences between Multi-level and OMG/UML Infrastructure (2-level) based modeling
- Multi-level Aware Transformations
- **Case Study**

Case Study: Making Naomi Multi-level

- Eclipse plug-in (Naomi) to support Orthographic Software Modeling with the KobrA approach
- Developed by Christian Tunjic (tunjic@informatik.uni-mannheim.de)
- Available at http://eclipselabs.org/p/naomi
- Licensed under Eclipse Public License (EPL) version 1.0
- Heavy use of “2-level to *-level” and “*-level to 2-level” transformations
- Transformations available at http://svn.codespot.com/a/eclipselabs.org/naomi

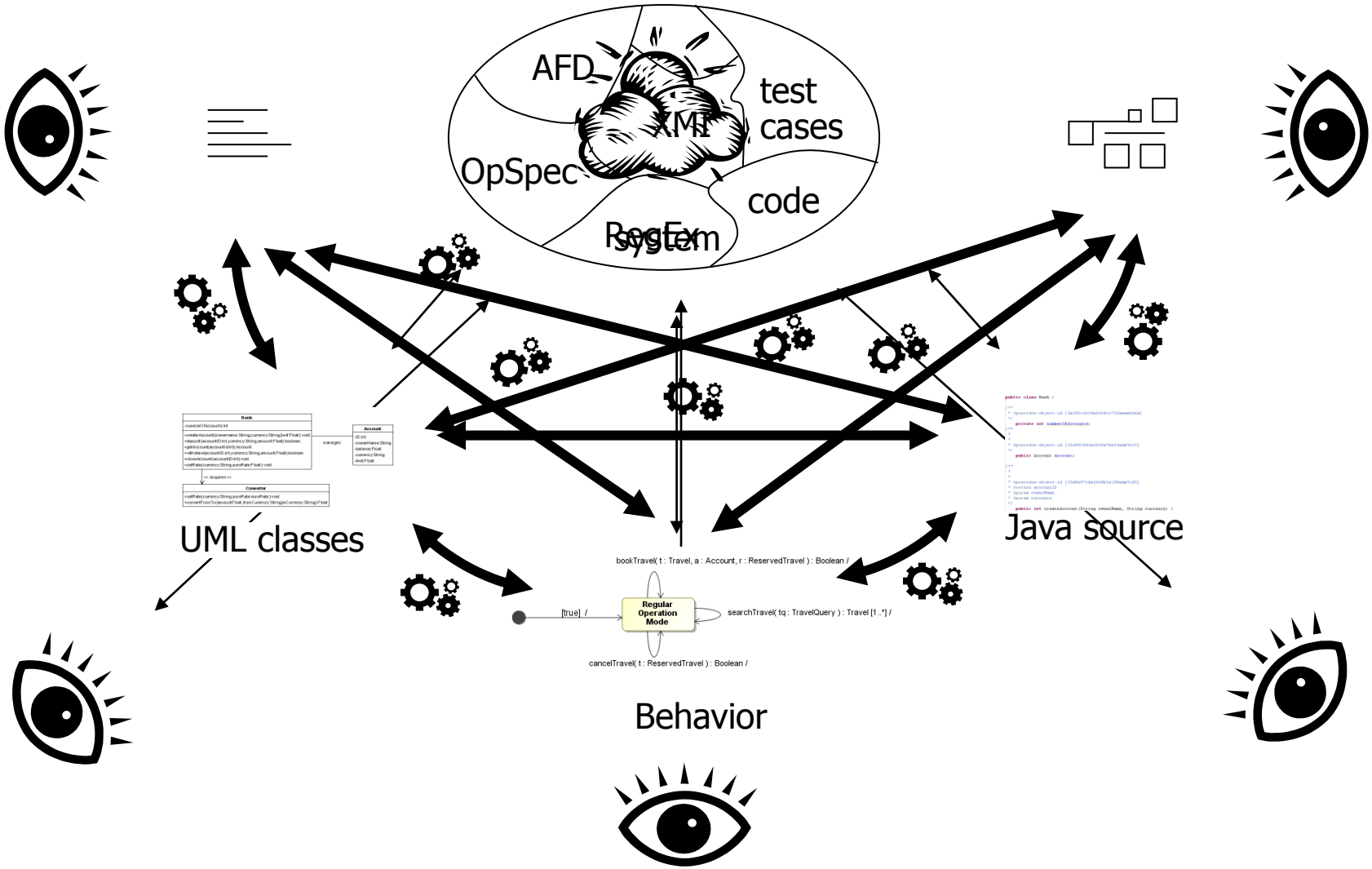
A Simple View-Based "Modeling" Metaphor

- Other engineering disciplines have a long and successful tradition of technical drawing
  - Orthographic projection

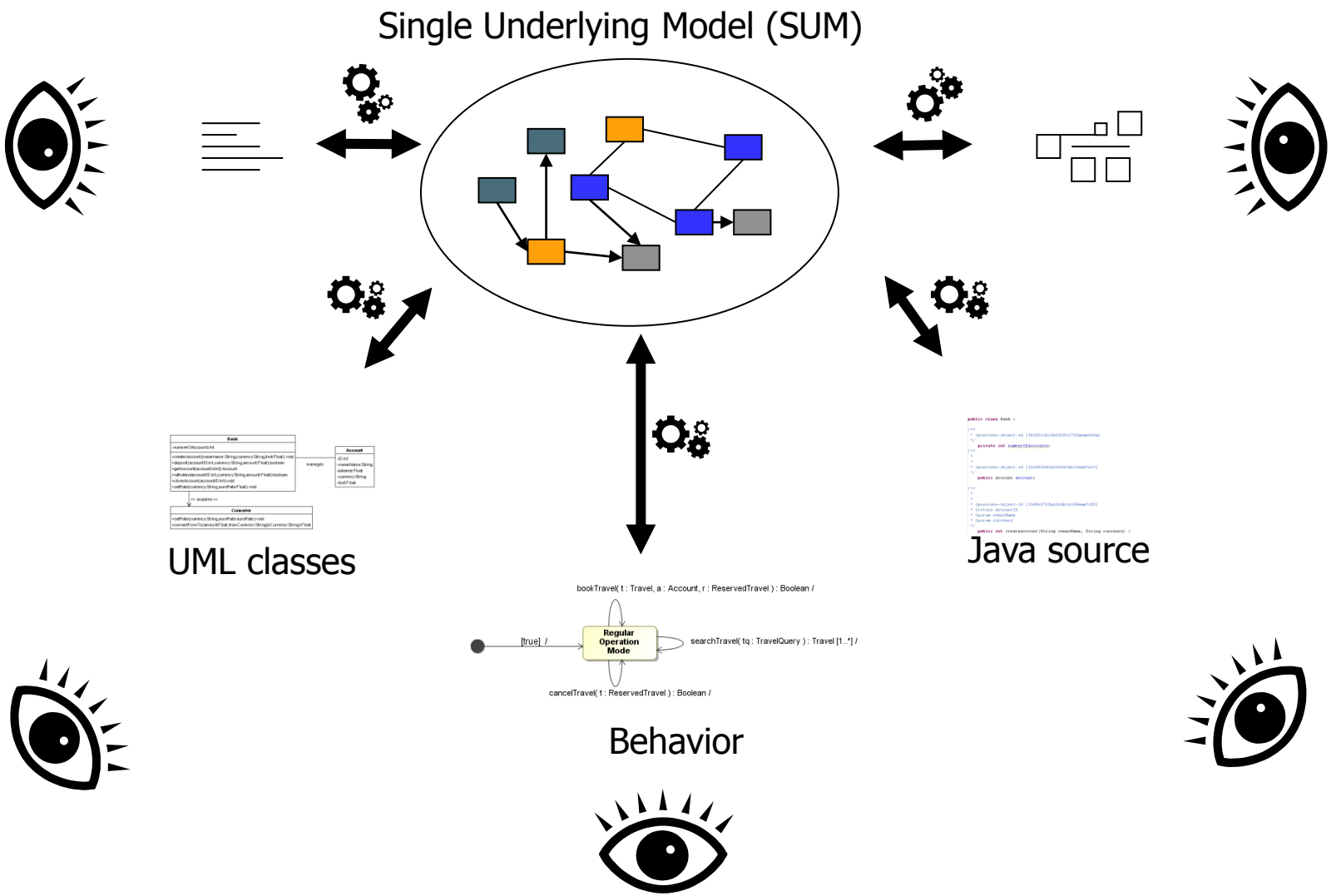


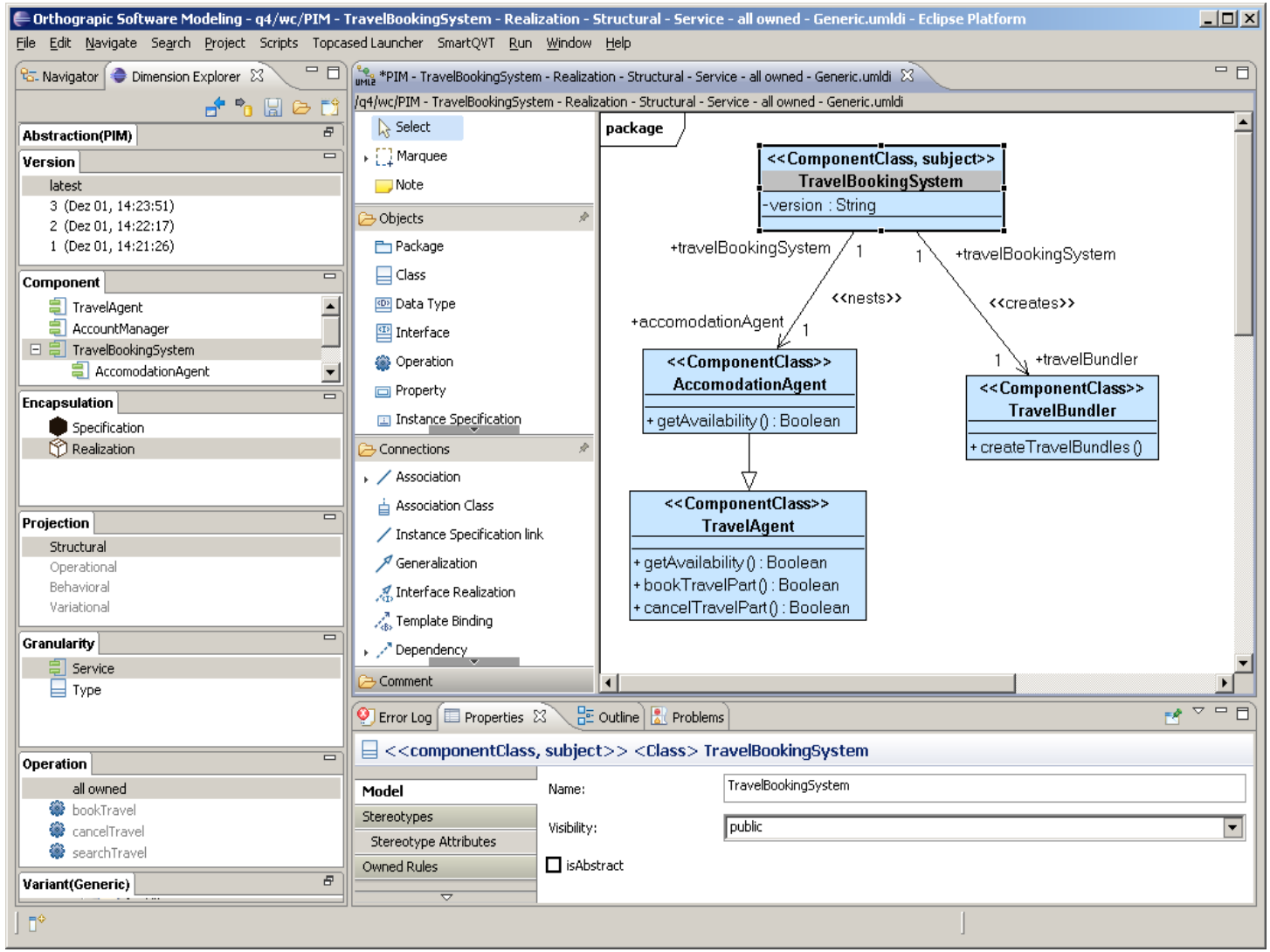
- So why don't we do this in software engineering?

Traditional View-based Environment

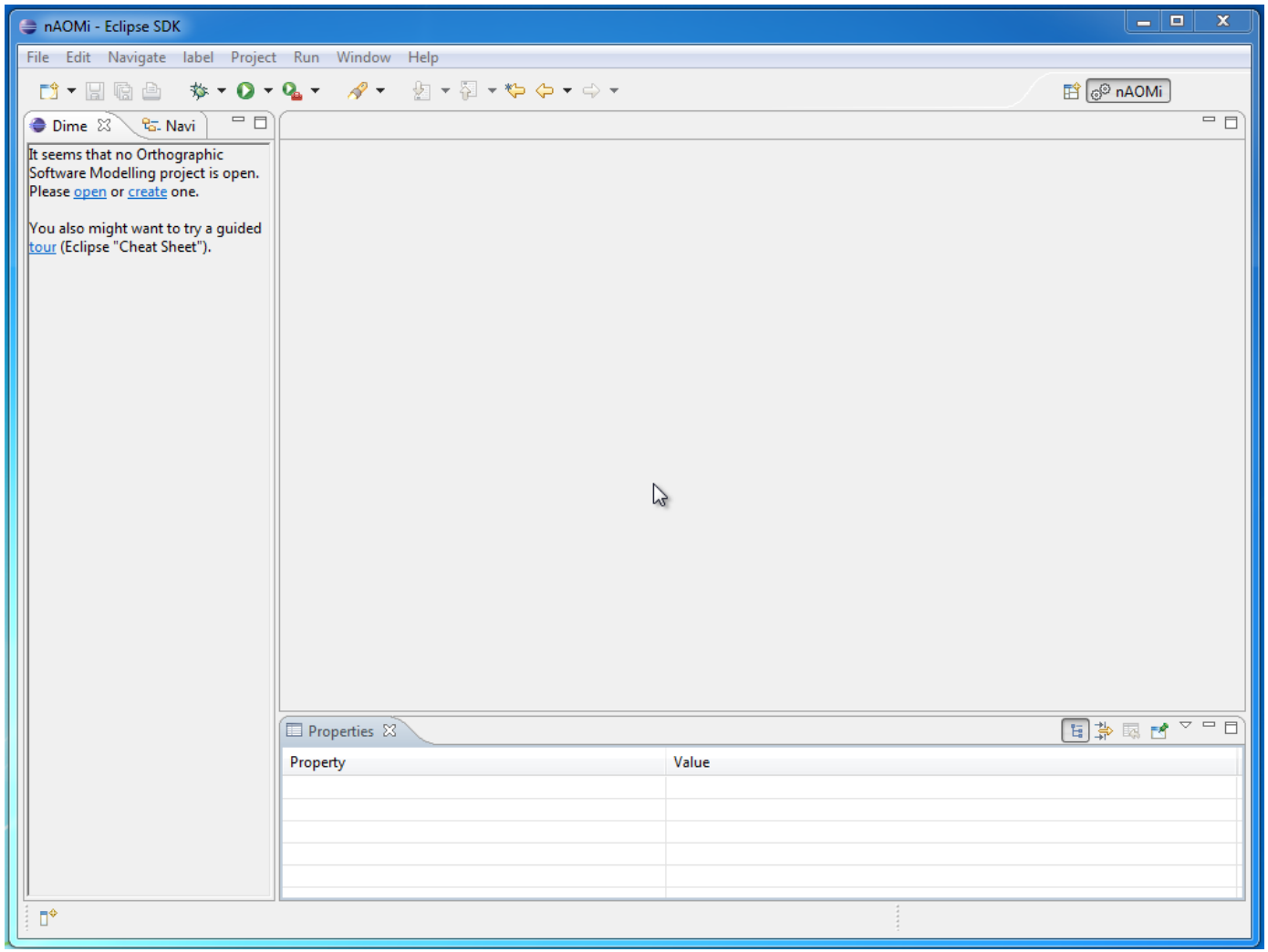


On-Demand View Generation




Orthograpic Software Modeling - q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi - Eclipse Platform
File Edit Navigate Search Project Scripts Topcased Launcher SmartQVT Run Window Help
Navigator
Dimension Explorer
*PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi
/q4/wc/PIM - TravelBookingSystem - Realization - Structural - Service - all owned - Generic.umldi
Abstraction(PIM)
Version
latest
3 (Dez 01, 14:23:51)
2 (Dez 01, 14:22:17)
1 (Dez 01, 14:21:26)
Component
TravelAgent
AccountManager
TravelBookingSystem
AccomodationAgent
Encapsulation
Specification
Realization
Projection
Structural
Operational
Behavioral
Variational
Granularity
Service
Type
Operation
all owned
bookTravel
cancelTravel
searchTravel
Variant(Generic)
Select
Marquee
Note
Objects
Package
Class
Data Type
Interface
Operation
Property
Instance Specification
Connections
Association
Association Class
Instance Specification link
Generalization
Interface Realization
Template Binding
Dependency
Comment
package
<<ComponentClass, subject>>
TravelBookingSystem
-version : String
+travelBookingSystem
1
1
+travelBookingSystem
<<nests>>
<<creates>>
+accomodationAgent
1
1
+travelBundler
<<ComponentClass>>
AccomodationAgent
+ getAvailability() : Boolean
<<ComponentClass>>
TravelBundler
+ createTravelBundles ()
<<ComponentClass>>
TravelAgent
+ getAvailability() : Boolean
+ bookTravelPart() : Boolean
+ cancelTravelPart() : Boolean
Error Log
Properties
Outline
Problems
<<componentClass, subject>> <Class> TravelBookingSystem
Model
Stereotypes
Stereotype Attributes
Owned Rules
Name:
TravelBookingSystem
Visibility:
public
isAbstract

Naomi – Video Demo



OSM – Example Transformation

- Transformation (excerpt) from View to Single Underlying Model (SUM):

```
create OUT : UML from IN : PLM;

helper context PLM!"O0::Acquires" def : createName:String =
    'From' + self._o_.source._l_.name + 'To' +
    self._o_.target._l_.name;

rule ComponentClass2Class {
  from s : PLM!"O0::ComponentClass"
  to t : UML!Class (name <- s._l_.name )
  do {
      thisModule.umlModel.packagedElement <-
    thisModule.umlModel.packagedElement->append(t);
   }
}
```

Conclusions

- Multi-level transformations are an important part for making multi-level modeling more usable
- First steps..
  - Linguistic vs. Ontological classification
  - Ontology aware naming scheme
  - Running prototype
- Natural extensions are ..
  - to specify Potency of rules
  - Extension of ATL's multi-level aware OCL part
  - Extension to the ATL editor to support editing of mulit-level transformations
- Ultimate goal is a language that does..
  - Transformations, Rules, Enquiries, Actions, and Constraints
  - TREACLE

Thank You!