Creating and Reconciling Diagrams after Executing Model Transformations

Torbjörn Lundkvist

Perlen-Seminar 23/07/2008

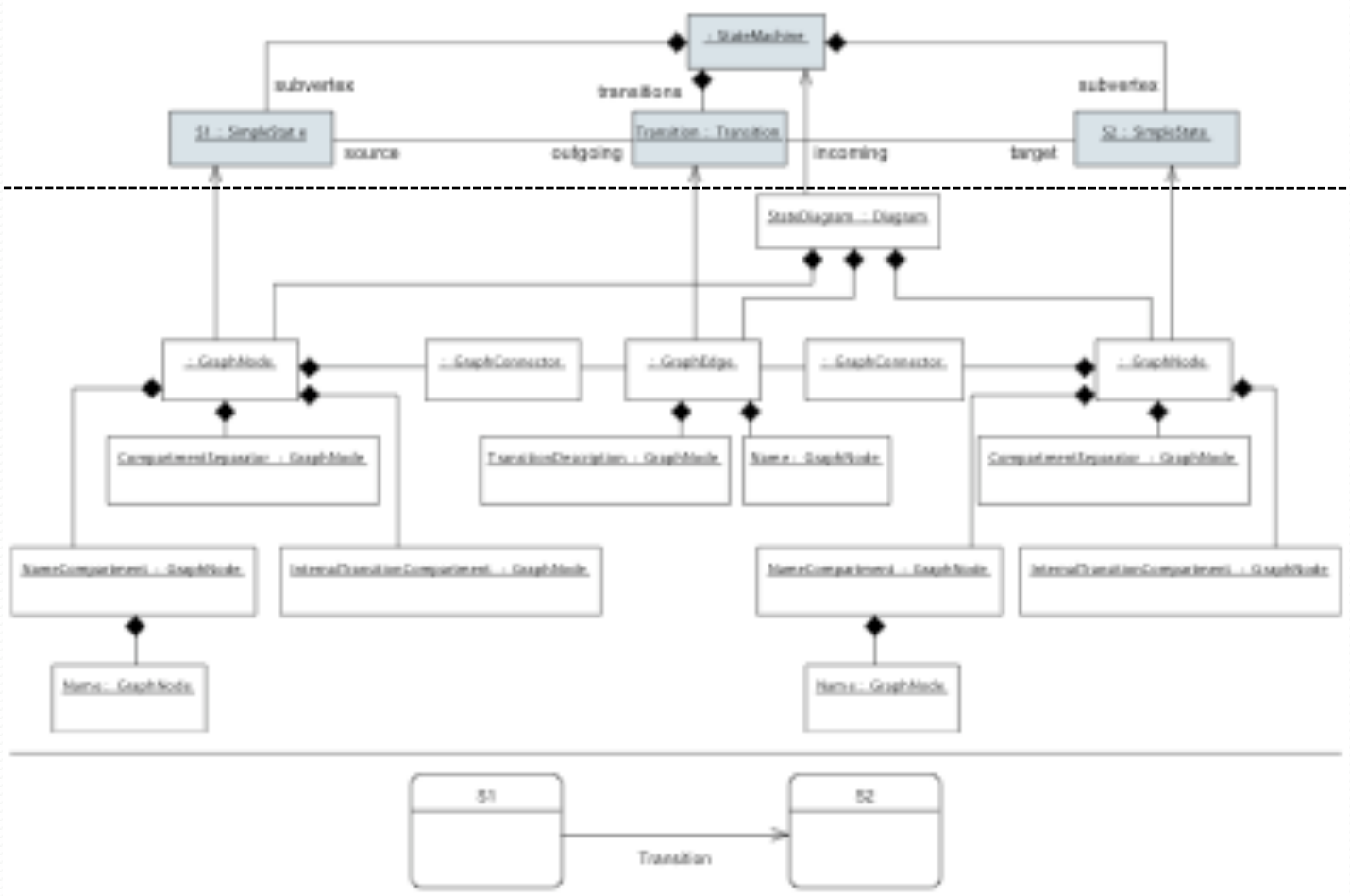
Introduction

- Software modeling languages
  - Formal (abstract) syntax defined using a metamodel
  - Visual notations (concrete syntax) to help humans understand the system
- Both the abstract and concrete syntax is important for manipulating and creating views of models
- To fully realize the possibility of having multiple views, the abstract and concrete syntax should be separated

Introduction (cont.)

- In the OMG standards, the abstract and concrete syntax are two separate but related artifacts
  - MOF, UML 2.0 Infra to define modeling languages
  - UML 2.0 Diagram Interchange (DI) to represent diagrams containing only graphical information
- Both models stored alongside in an XMI document
- DI model links the visual representation to the abstract model using unidirectional links
- But, OMG did not specify how diagrams relate to abstract models, or how DI diagrams are constructed
- This information is needed to construct new and maintain existing diagrams in modeling tools

Example Model with DI

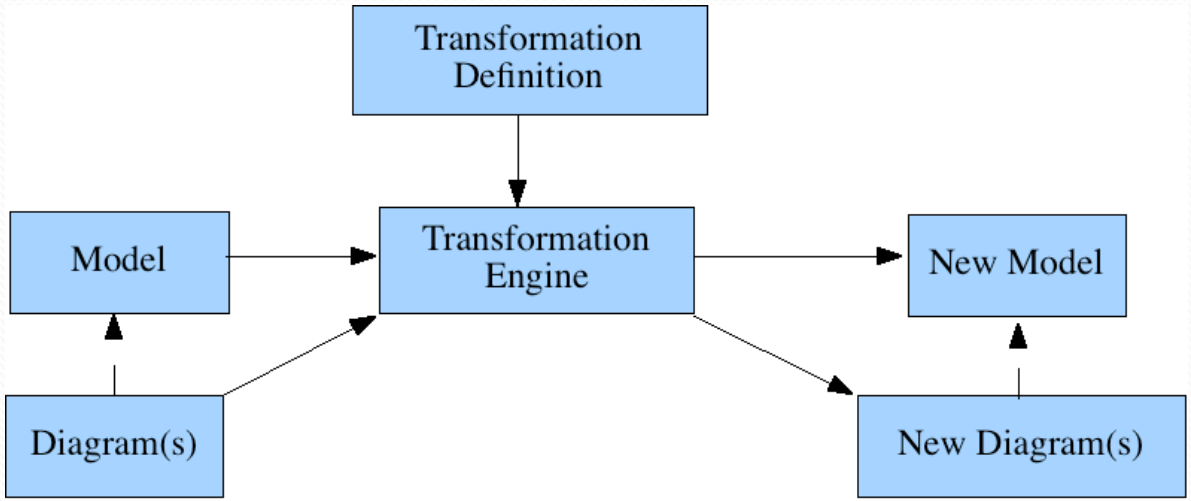


Model Transformation

- A model transformation component reads a model M and transforms it to a (new) model M' according to a transformation definition
  - Model transformation languages (e.g. QVT, Viatra...)
  - MT using programming languages
  - We can also consider editors in a modeling tool to be seen as a MT component
- Often deals with abstract syntax alone, and hence related diagrams are not preserved

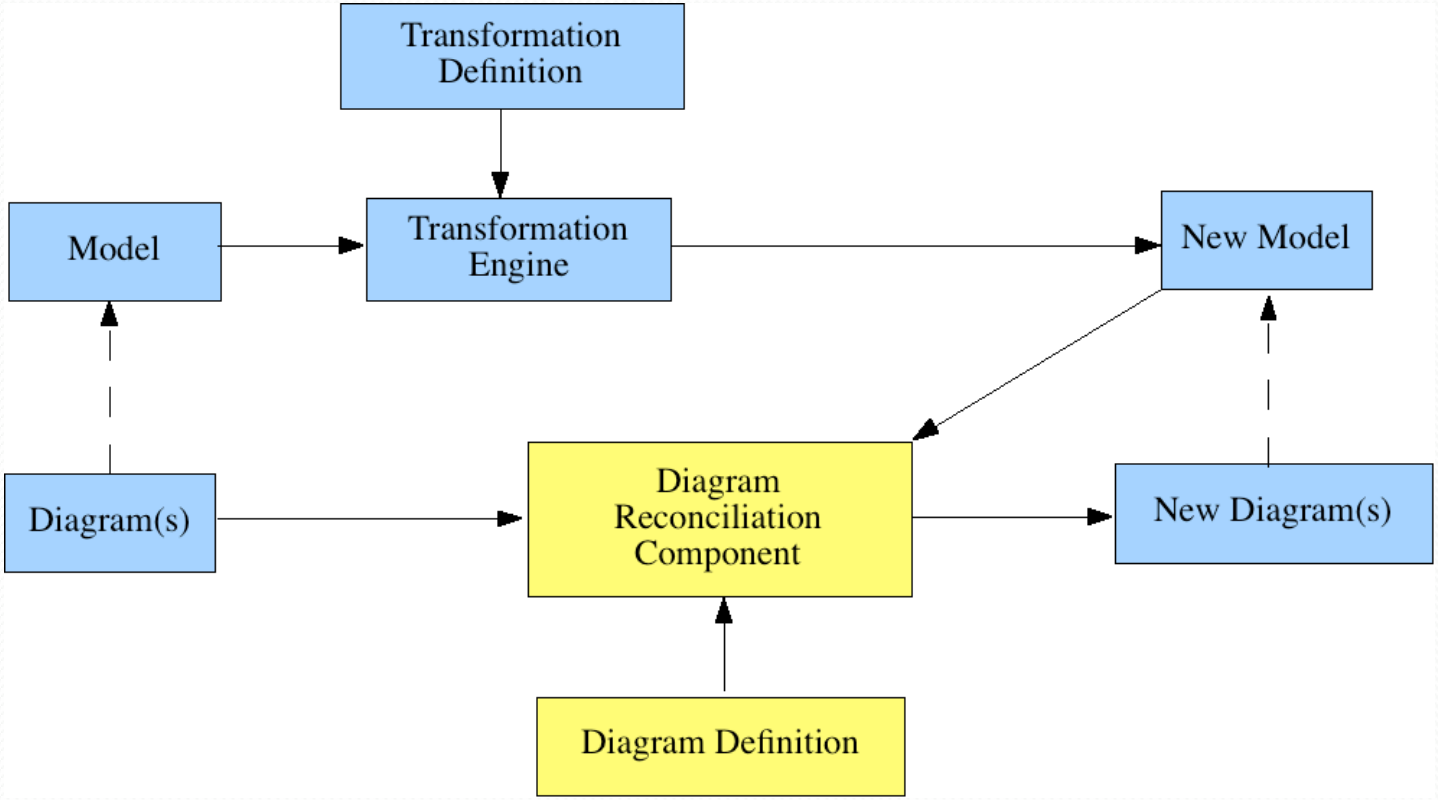
One solution?

- The transformation could maintain the diagrams but leads to very complex and error prone transformations
- Often the case in VL tool frameworks
- What if there are several transformation engines?



The Proposed Solution

- Using a separate diagram reconciliation component to maintain consistency between models and diagrams



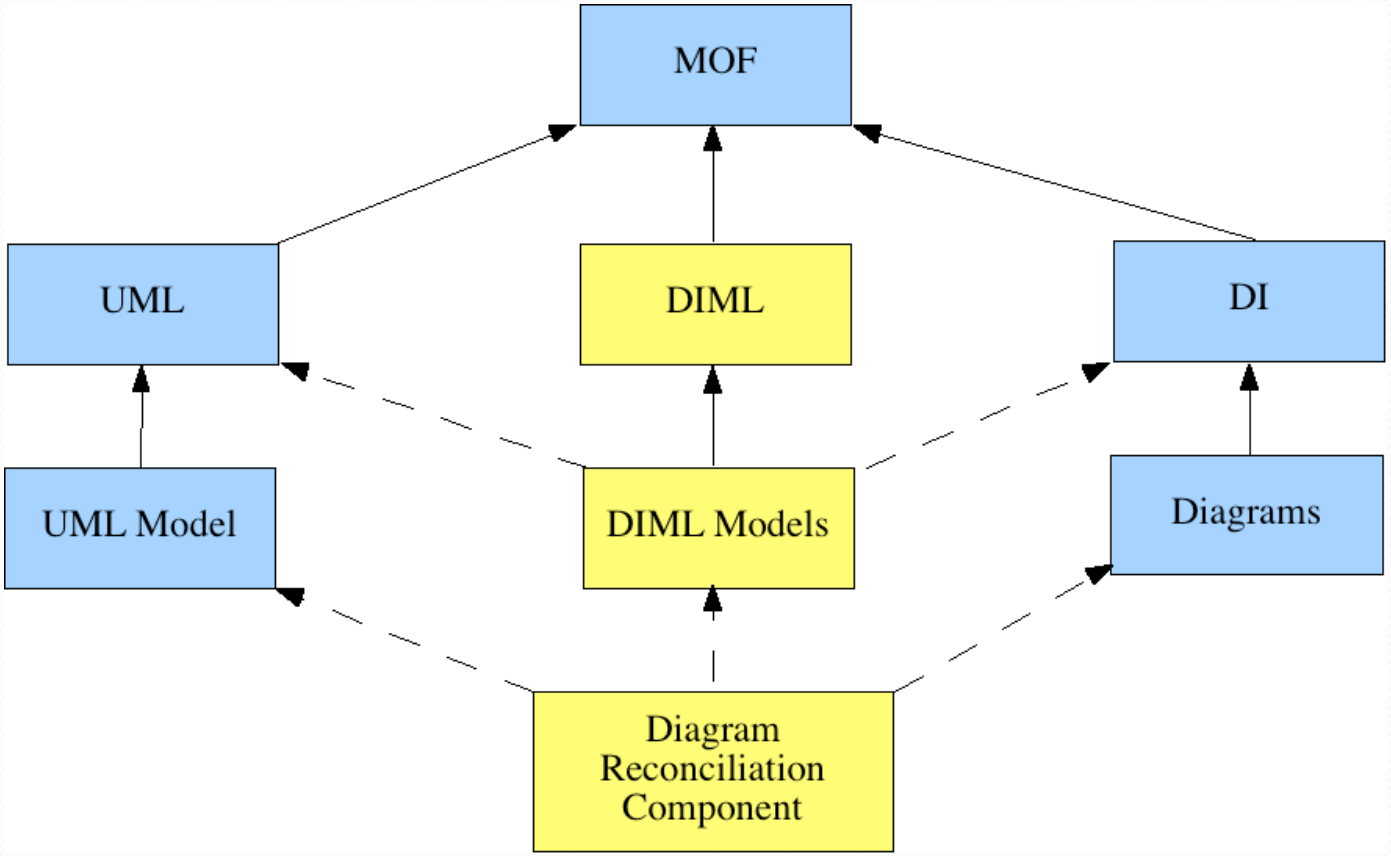
The Proposed Solution

- Based on the idea that a diagram can be defined based on the relation or *mapping* between a model and a diagram
- This mapping can then be interpreted and applied to an abstract model (AM) to
  - Create new diagrams out of pure AM data
  - Reconcile existing diagrams when changes occur in AM while preserving as much layout information from the diagrams as possible

Diagram Interchange Mapping Language

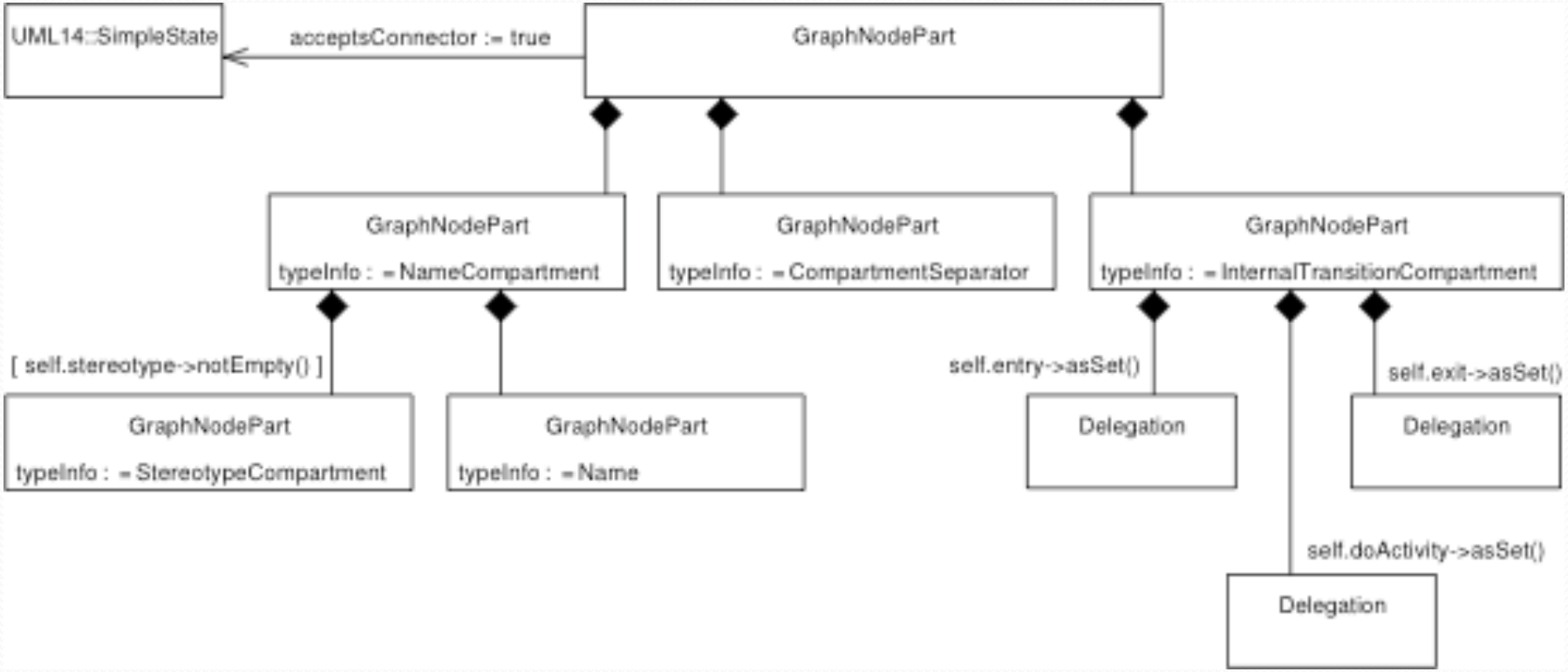
- Defines the relation between concepts in a modeling language to concepts in the DI language
- Uses the fact that diagrams can be created top-down
- A DIML mapping is a unidirectional map between a metaclass and a parameterized DI skeleton
- The parameterization is done by using OCL expressions to control the creation of diagrams
- DIML mappings declare how diagrams should look like and do not impose a particular algorithm of application
- Details of this language in journal paper

Overview of DIML wrt. OMG Standards



DIML Example

- Shows the mapping of a UML SimpleState



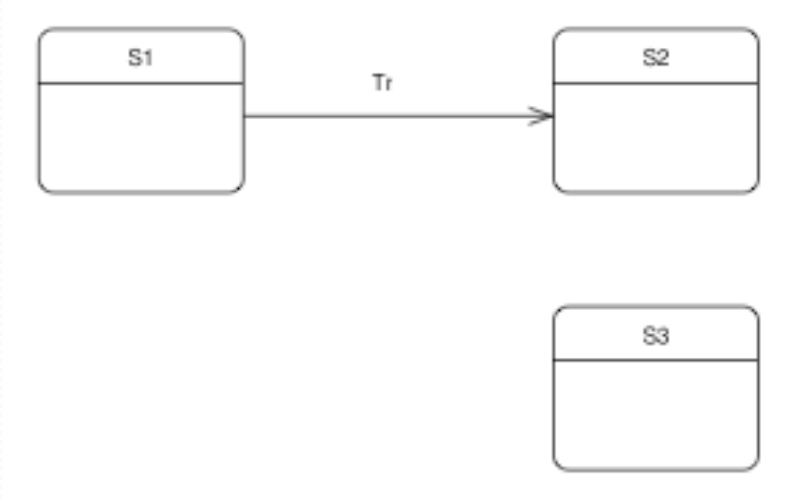
Creating New Diagrams

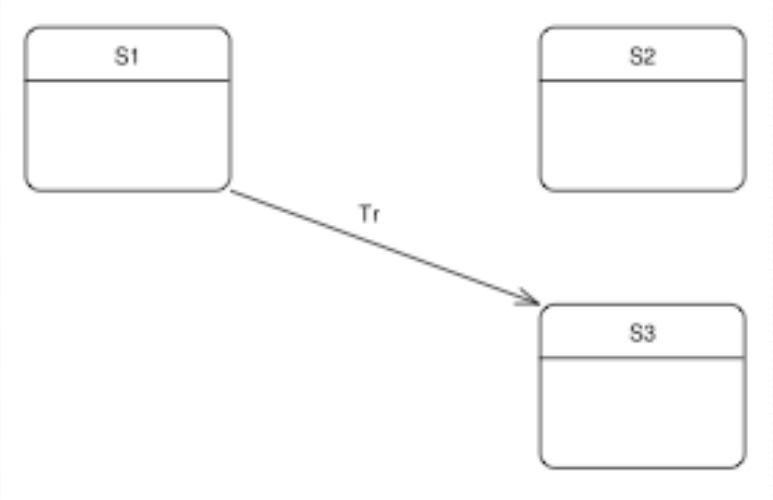
- The mappings can be used to generate new diagrams purely out of abstract model data
- This can be useful when e.g. new models have been created as a result of a conversion to another language
- Can be handled using a general depth-first algorithm traversing the DIML mappings and querying the abstract models at the same time
- Not however usable when a model is only slightly modified, as layouting will not be preserved

Diagram Reconciliation

- From the DIML mappings it is at all times possible to calculate how a diagram should look like given a model
- Hence, by having access to the obsolete diagram, the old model and a change description, it is possible to bring a diagram up-to-date using minimal changes
- The necessary changes are calculated using generic optimized reconciliation algorithms
- However, DIML itself does not enforce a particular reconciliation algorithm

Reconciliation Example





Execution of command “Tr.target = S3”
Instead of recreating the transition in the diagram, the connector is simply moved to the new class

Validation

- Since the mappings are declarative, there are many possible applications of the DIML language
- The diagram reconciliation component based on DIML provides one single source of diagram management in a modeling tool
- Reconciliation is truly independent of how a transformation has been executed on a model
- The principles of diagram reconciliation are possible to apply on other diagramming languages as well
- Satisfies key points in the OMG RFP *Model view to diagram*, an effort standardize diagram definition

Conclusion

- We have decoupled diagram updates from transformations on the abstract models
- This is based on a mapping language between a modeling language and diagrams
- Implemented in the Coral Modeling Framework and is used as the only source of updating a diagram
- More specific details in the journal publication: **M. Alanen, T. Lundkvist, I. Porres**: *Creating and reconciling diagrams after executing model transformations.* Science of Computer Programming 68(2007) p. 155-178