From Class Diagrams to Zope Products with the Meta-Modelling Tool $AToM^3$

Andriy Levytskyy,

Eugene J. H. Kerckhoffs

TU Delft / ITS / Mediamatica / KBS
Mekelweg 4, 2628 CD Delft,
The Netherlands

a.levytskyy@cs.tudelft.nl

Presentation Overview

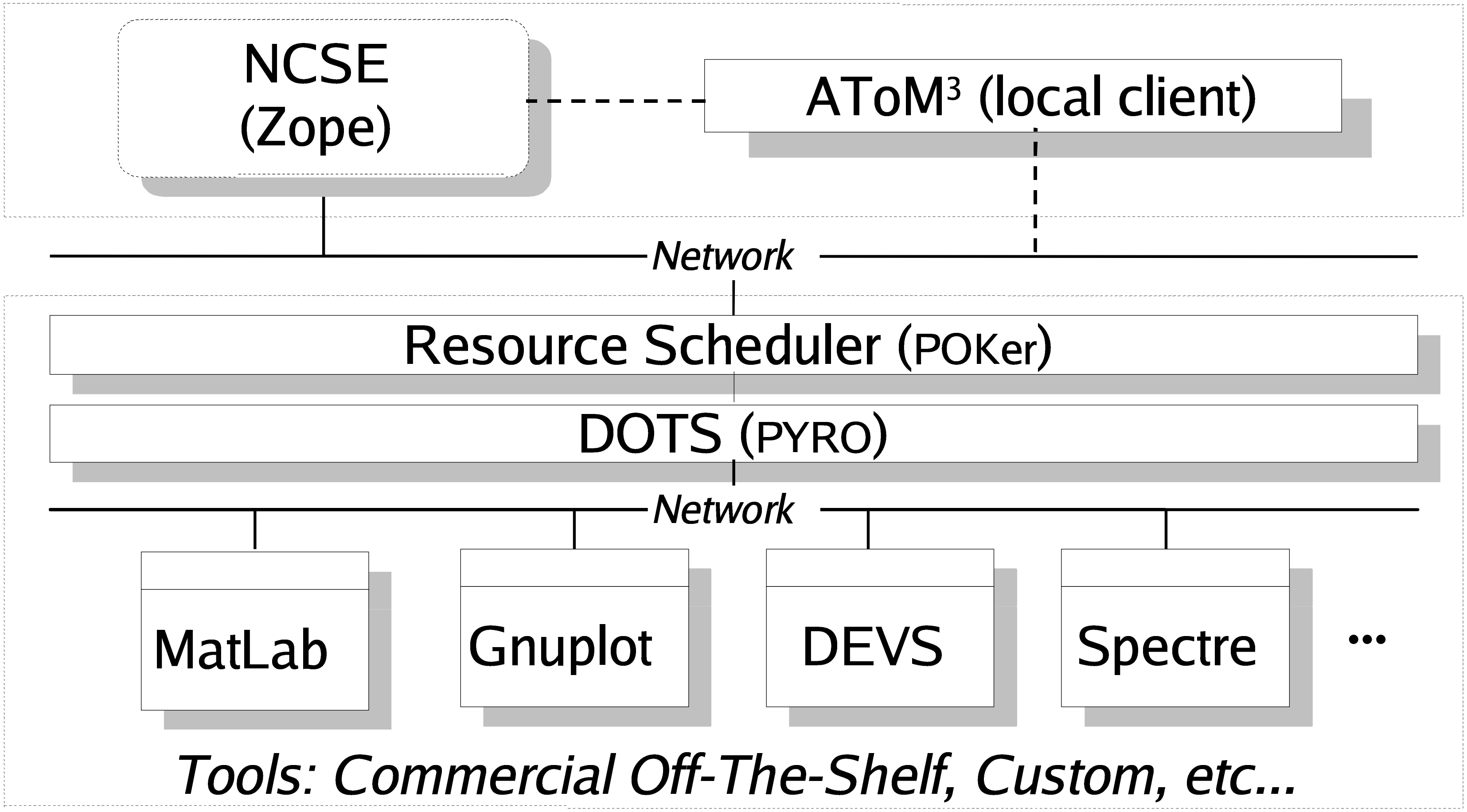
- Environment Overview
- Extending Zope
- Simplified Class Diagrams
- Meta-Modelling and Transforming
- Code Generation Example
- Conclusions

Environment Overview

- Online virtual laboratory:
  - Registration and Discovery of scientific Models and Tools
  - Access to experiments
- Treats models and tools as limited Internet Resources
- Controls access to Resources
- Extensibility of Resources
  - Little integration requirements for tools
  - Support for new model families through metamodels with AToM^{3*}

*J. de Lara and H. Vangheluwe, *AToM3: A Tool for Multi-Formalism Modelling and Meta-Modelling.*

Environment Overview (2)



NCSE (Zope)

AToM^3 (local client)

Network

Resource Scheduler (POKer)

DOTS (PYRO)

Network

MatLab

Gnuplot

DEVS

Spectre

...

Tools: Commercial Off-The-Shelf, Custom, etc...

TU Delft

Delft University of Technology

Extending Zope

Products provide a way to extend Zope with custom types of objects tailored to needs of a specific application.

Current *Product Construction* is based on on the *mxm Easy product* * and features:

- Atomic or Container Objects
- Predefined properties (`_properties`) and methods
- Children control (`_allowed_meta_types`)
- Views and permissions for ZMI (`__ac_permissions__`)
- Product has the following file structure:

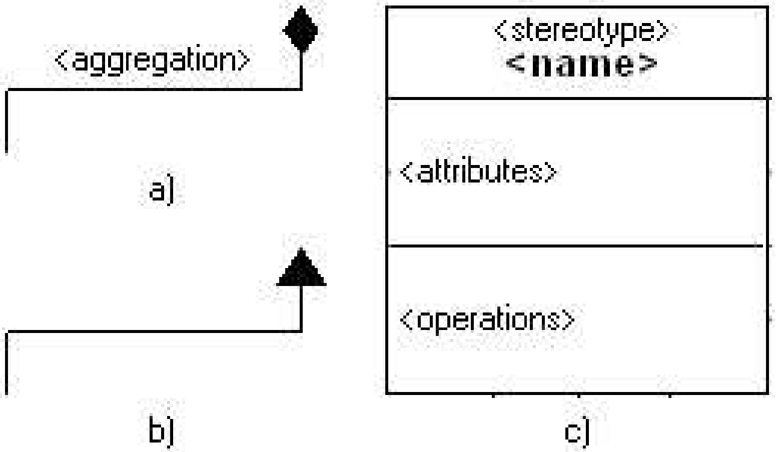
```
myProduct/
    myProduct.py
    __init__.py
```

* M. Max, *An easier way to write products.*

Simplified Class Diagrams

Simplified Class Diagrams (SCD) is a custom engineering method based on, and fewer and simpler in features then UML class diagrams.

Appearance:



Concepts:

a) Association
b) Generalization
c) Class

Well-Formedness Rules:

- based on the originals from the UML metamodel
- Only selected rules that are meaningful in the simpler SCD context, are left.

TU Delft
Delft University of Technology

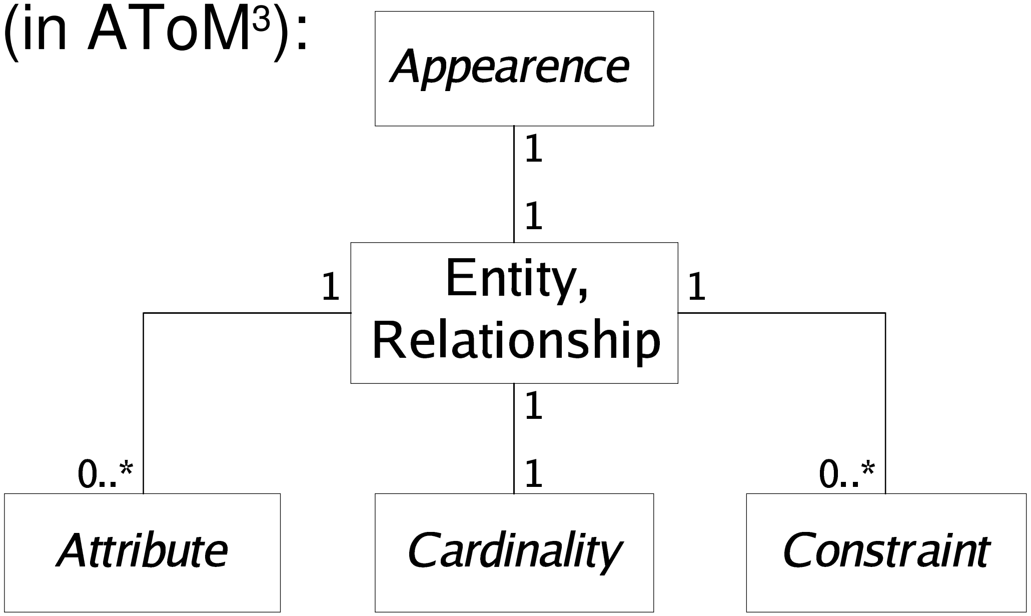
Modelling a Metamodel

Metamodel specifies the syntax aspect of a formalism by defining the language constructs and how they are built-up in terms of other constructs.

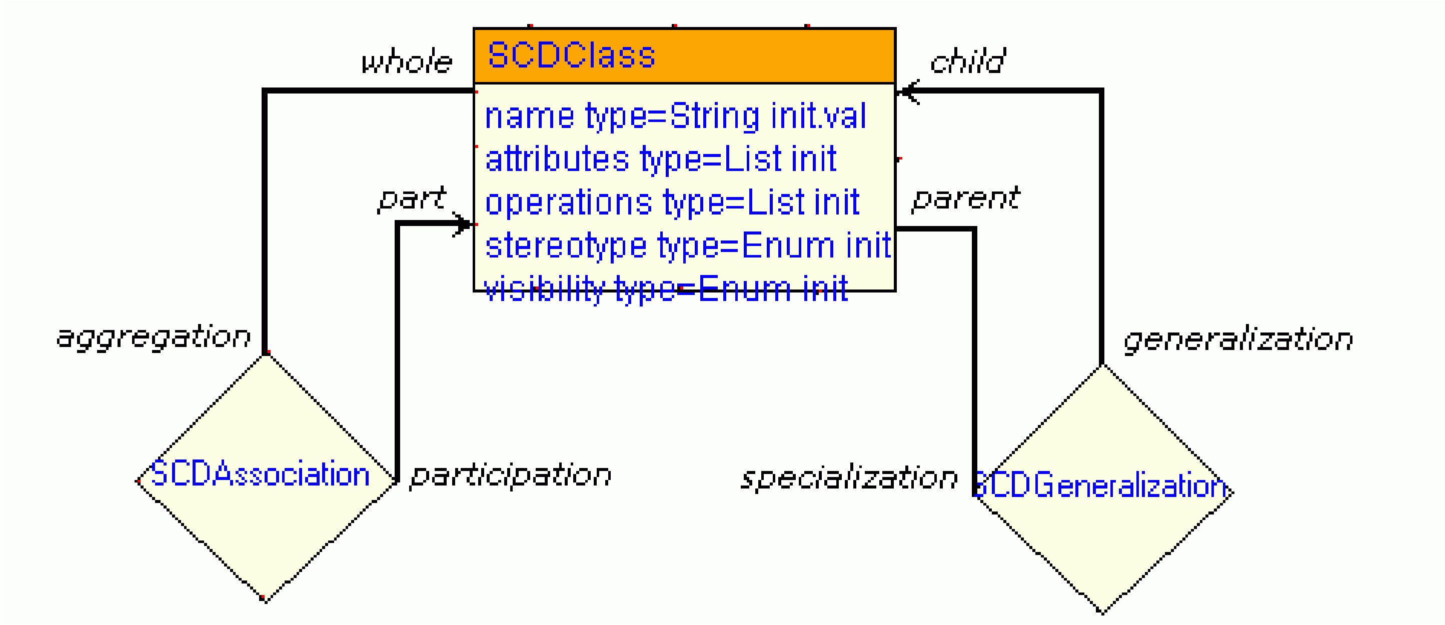
Modelling Formalism (in AToM^3):

ER + constraints

Constructs:



SCD Metamodel


SCDClass
name type=String init.val
attributes type=List init
operations type=List init
stereotype type=Enum init
visibility type=Enum init
whole
child
part
parent
aggregation
generalization
SCDAssociation
participation
specialization
SCDGeneralization

Global Properties:

(*name*, *title*, *subject*, *description*, *author*, *version*, *attributes*†, *constraints*†)

Some Well-Formedness Rules

SCDAssociation :: *EDIT*(…), *CONNECT*(…)

post: *len*(*SCDAssociation.allConnections*) >= 3 **and**
SCDAssociation.aggregation = #none

SCDGeneralization :: *CONNECT*(…)

post: *self.child* –> *forAll*(*c* | **not** *c.isRoot*)

SCDClass :: *CONNECT*(…)

post: *self.attributes* –> *forAll*(*a1*, *a2* |
a1.name = *a2.name* **implies** *a1* = *a2*
)

Modelling a Transformation

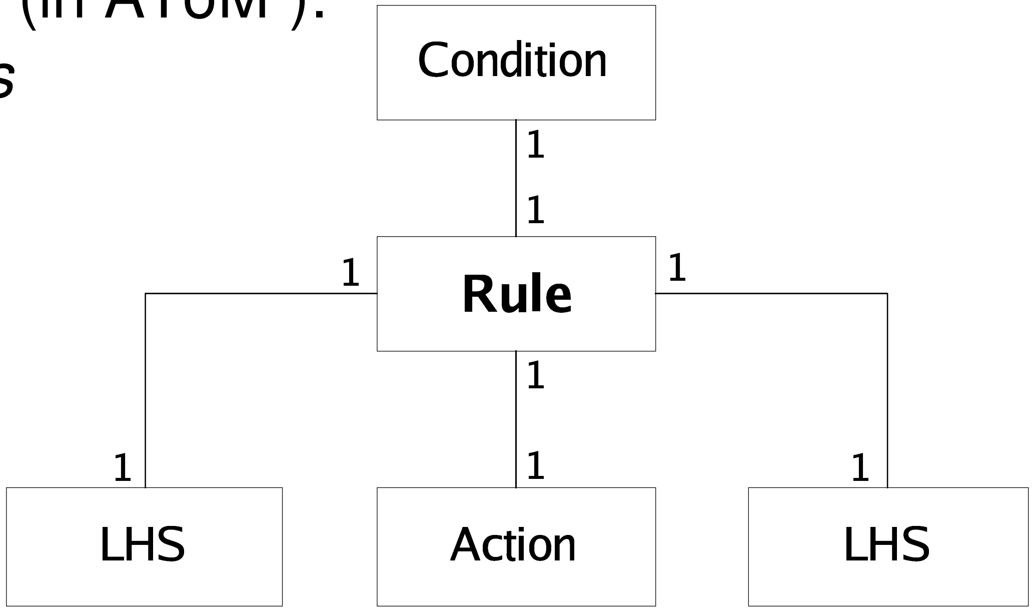
Model transformation is related to dynamic semantics of a formalism, which defines the meaning of well-formed constructs. A model can be transformed into another model of the same or different formalism.

Modelling Formalism (in AToM^3):

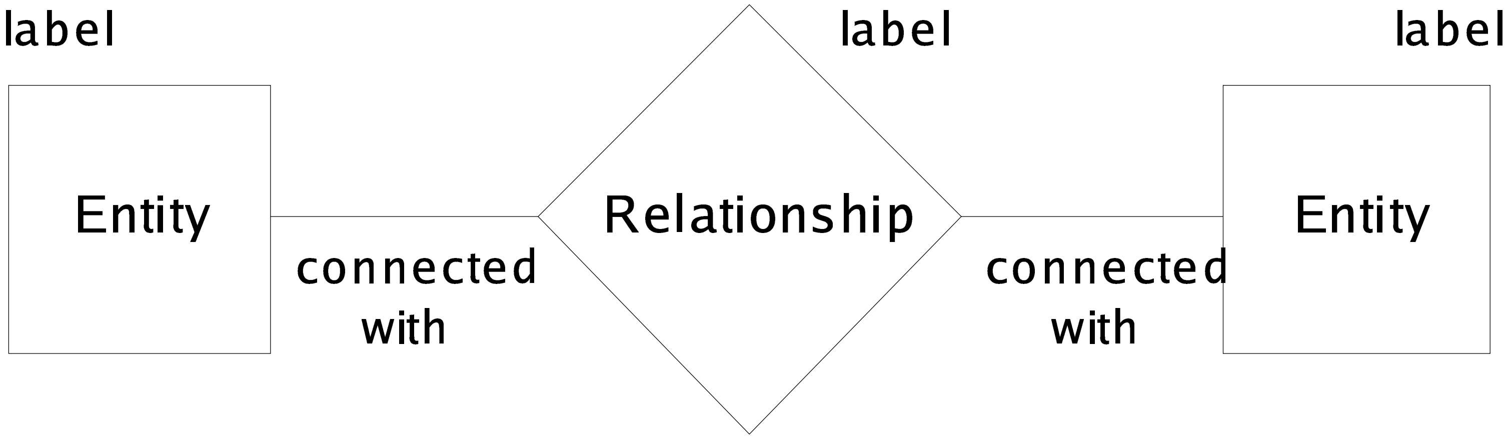
Graph Grammars

Constructs:

- *Initial Action*
- *Rules*
- *Final Action*



LHS and RHS pattern



- The rightmost entity can be omitted.
- Elements are labeled with successive numbers.

SCD-to-ZProduct Transformation

INITIAL ACTION create list `‘body’` to store a signatures,

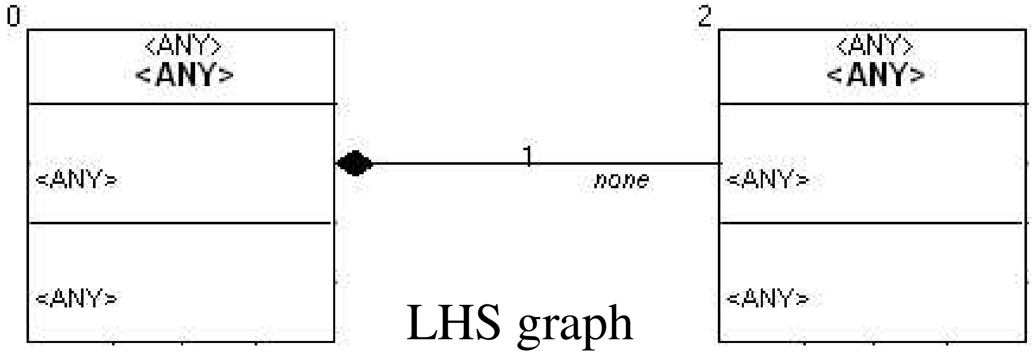
- **ruleAllowedMetaTypes** (*priority 1*) creates list `‘parts’` of allowed meta-types based on the associations of the CC.
- **ruleLocateImmidiateParent** (*priority 2*) selects an immediate parent of the CC and saves this information.
- **ruleMakeClassSignature** (*priority 3*) makes a signature of the current class and adds it to `‘body’`.
- **ruleChooseNewCurrent** (*priority 4*) picks up a class among possible candidates and makes it current (CC).

FINAL ACTION converts information in `‘body’` to a ZProduct structure and code.

TU Delft
Delft University of Technology

Example of a Rule

ruleAllowedMetaTypes locates classes associated with CC, copies the LHS to the RHS and stores references to non-abstract participants at CC's attribute `'parts'...`



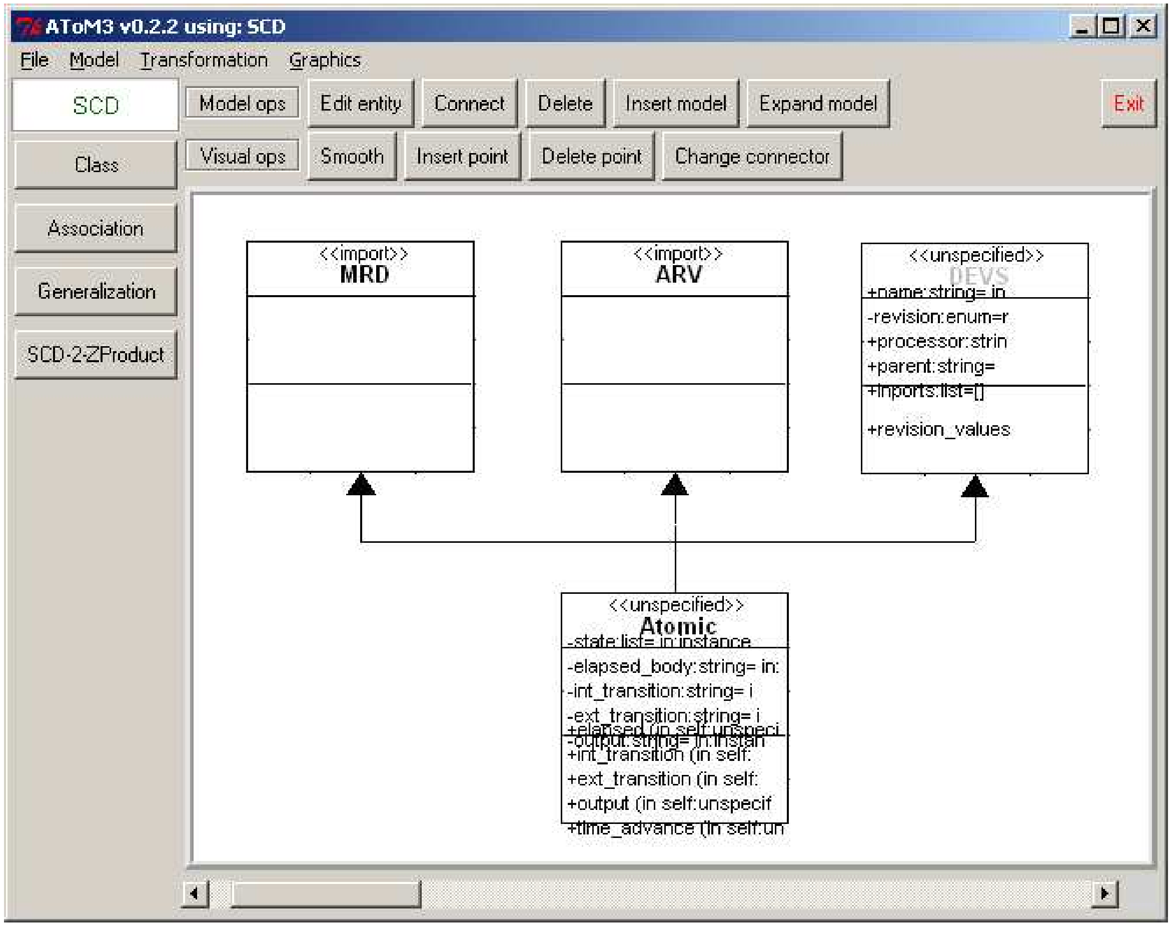
LHS graph

<u>Action</u>

pre: *LHS.element0.isCurrent* = 1
and *LHS.element1.aggregation* = #none
and *LHS.element2.isAssociated* = 0

post: *RHS.element2.isAssociated* = 1

SCD Tool and Example Model



TU Delft

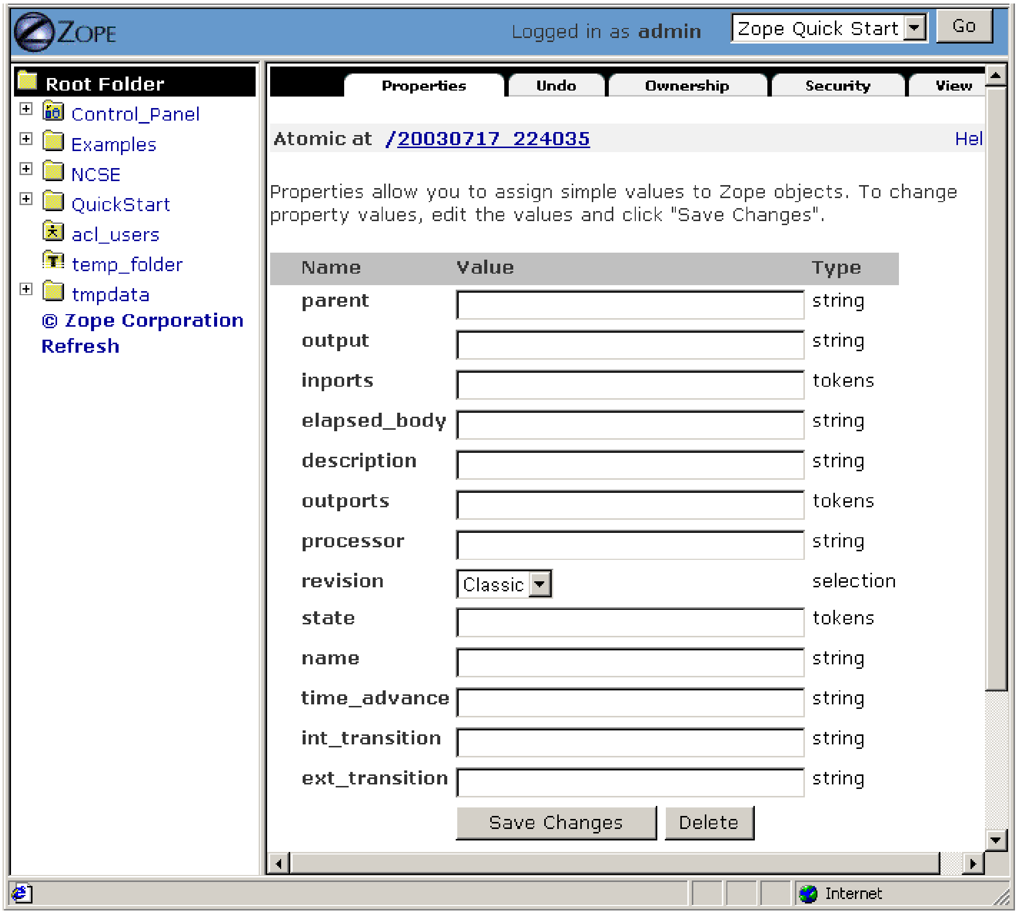
Delft University of Technology

Generated Code

```
class Atomic(mxmSimpleItem, MRD, ARV):
        """Atomic DEVS Component."""
        meta_type = ‘Atomic'

        _properties = ( {'type': 'string',    'id': 'name'},
                        {'type': 'string',    'id': ‘output_body'},
                          …
                         {'type': 'selection', 'id': 'revision', \
                          'select_variable': 'revision_values',},
                        ) + MRD._properties + ARV._properties

        def revision_values (self):
                """Return list of DEVS revisions."""
                return ['Classic', 'Parallel']

        def output (self):
                return self._getProperty(‘output_body’)
```

Delft University of Technology

Result in Zope



Atomic at /20030717_224035

Properties allow you to assign simple values to Zope objects. To change property values, edit the values and click "Save Changes".

Name	Value	Type
parent		string
output		string
inports		tokens
elapsed_body		string
description		string
outports		tokens
processor		string
revision	Classic	selection
state		tokens
name		string
time_advance		string
int_transition		string
ext_transition		string

TU Delft

Delft University of Technology

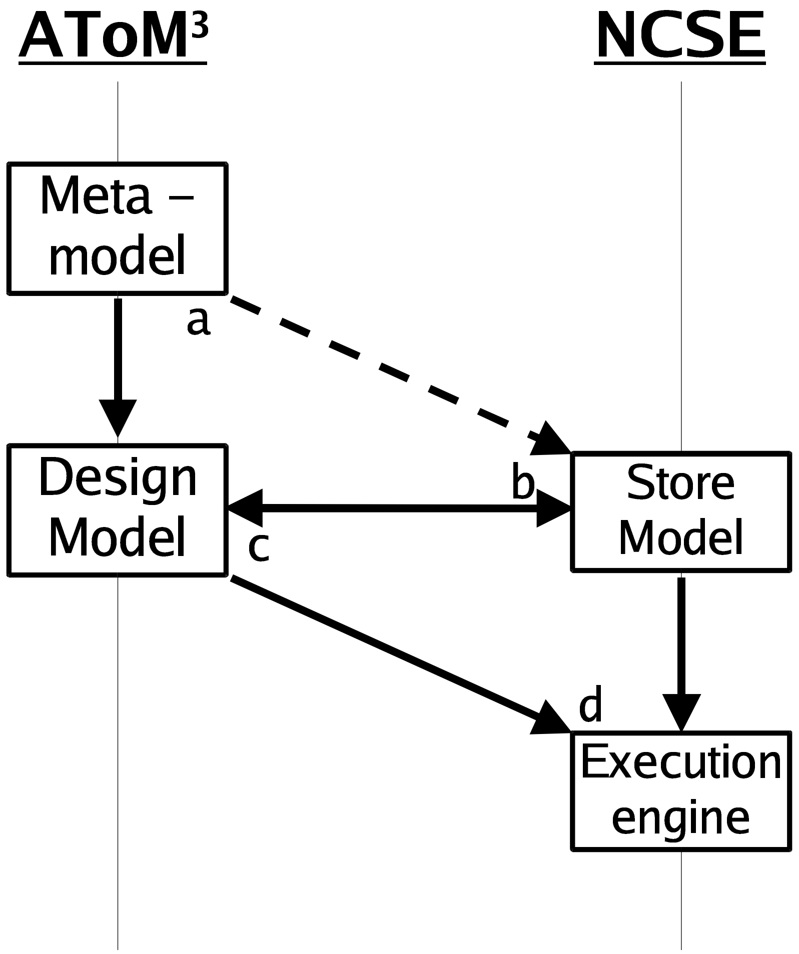
Conclusions

- A modeling environment for our custom design method was meta-modeled.
- Product generation was modeled.
- The resulting CASE tool is:
  - Domain-specific
  - Compatible with the existing technologies
  - Flexible, easy to control and use
- Future work will focus on extending the current design method with sequence diagrams in order to specify behavior.

NCSE and AToM³ Clients

- Create FM
- Save FM

- Load FM as MM
- Make models in F
- Generate code from models



- (Un-)register models
- Discover models
- View models
- Limited edit
- Set-up experiment

- Simulate models

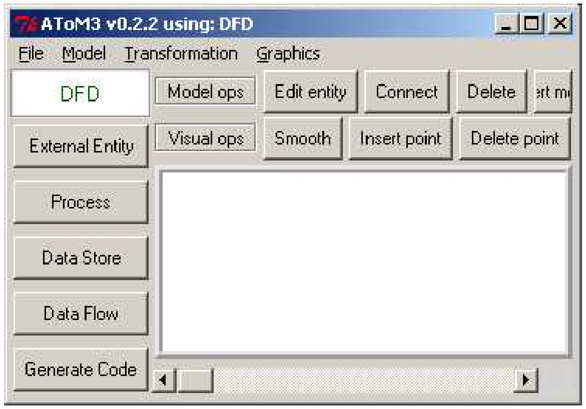
TU Delft
Delft University of Technology

Run Time View

SvOutPlaceObject

Generating a Tool

SvOutPlaceObject



TU Delft
Delft University of Technology

Model Transforming