Dresden OCL2 in MOFLON

TECHNISCHE
UNIVERSITÄT
DARMSTADT

10 Jahre Dresden-OCL – Workshop

ES Real-Time Systems Lab

Prof. Dr. rer. nat. Andy Schürr

Dept. of Electrical Engineering and Information Technology
Dept. of Computer Science (adjunct Professor)

www.es.tu-darmstadt.de

Felix Klar

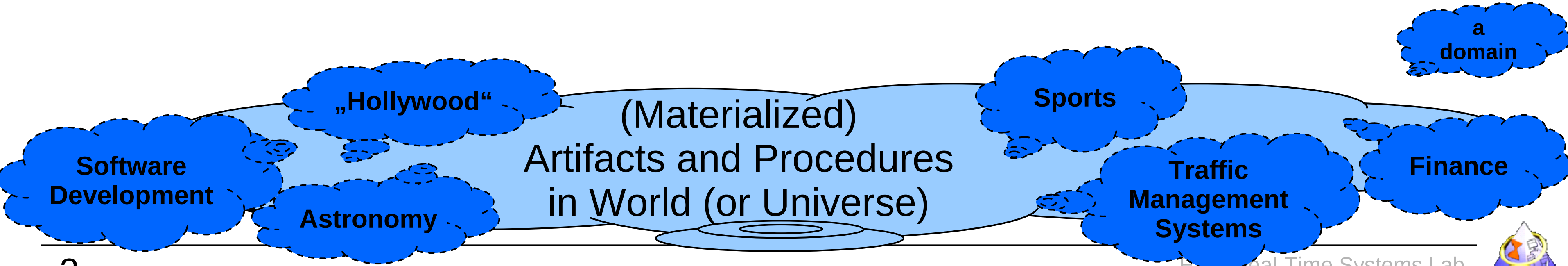
Felix.Klar@es.tu-darmstadt.de

15.10.2009

Outline

- Metamodels are Languages – A Motivation
- Metamodeling – Goals
- MOFLON – OCL
- MOFLON – Architecture
- MOFLON – Scenarios
- Demo (Integration Scenario – TiE-CDDS)
- Future Activities

Metamodeling – Overview and Motivation



„Hollywood“

Sports

a domain

Software Development

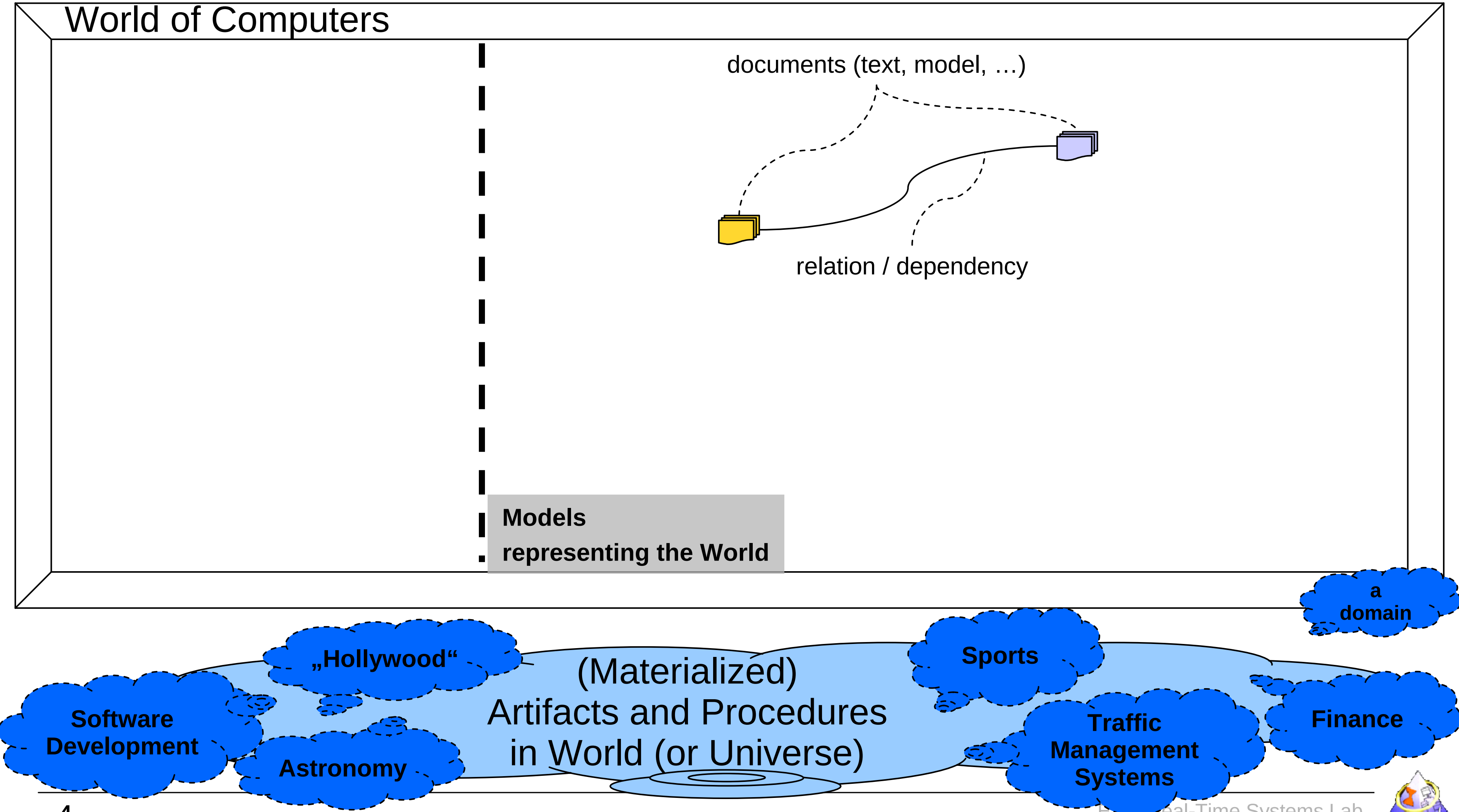
Astronomy

(Materialized) Artifacts and Procedures in World (or Universe)

Traffic Management Systems

Finance

Metamodeling – Overview and Motivation



World of Computers

documents (text, model, …)

relation / dependency

Models
representing the World

„Hollywood“

Software Development

Astronomy

(Materialized)
Artifacts and Procedures
in World (or Universe)

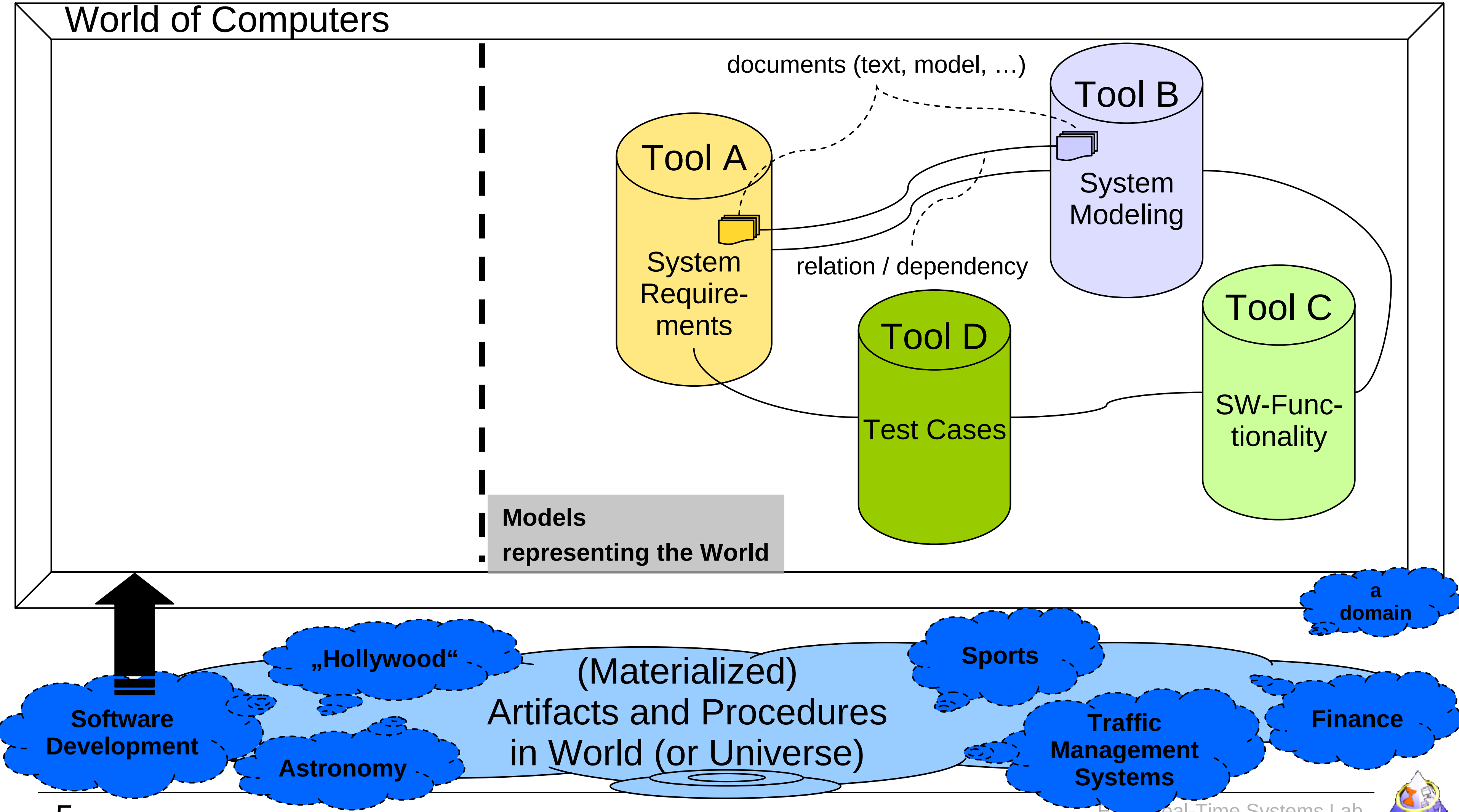
Sports

Traffic Management Systems

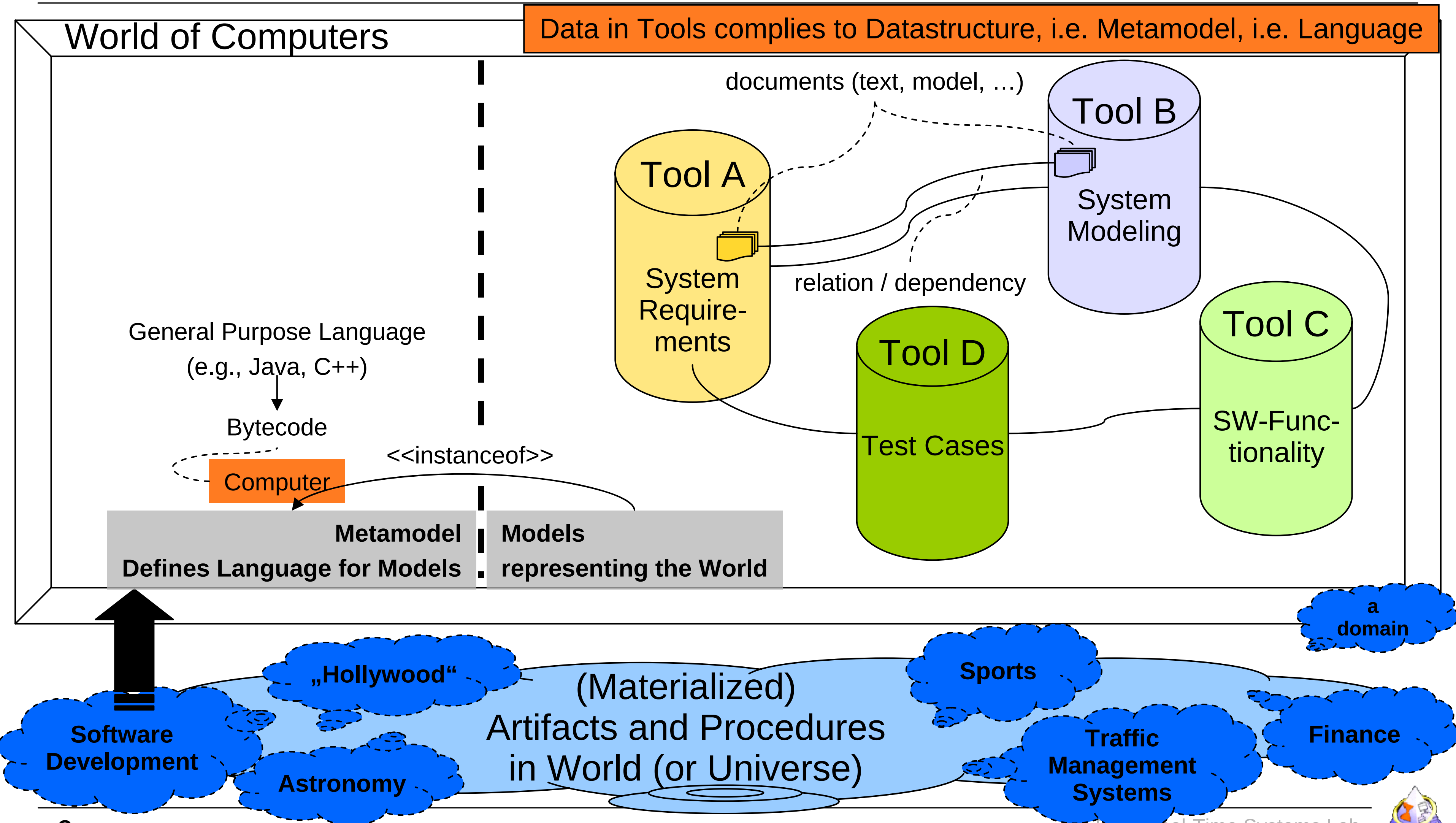
Finance

a domain

Metamodeling – Overview and Motivation

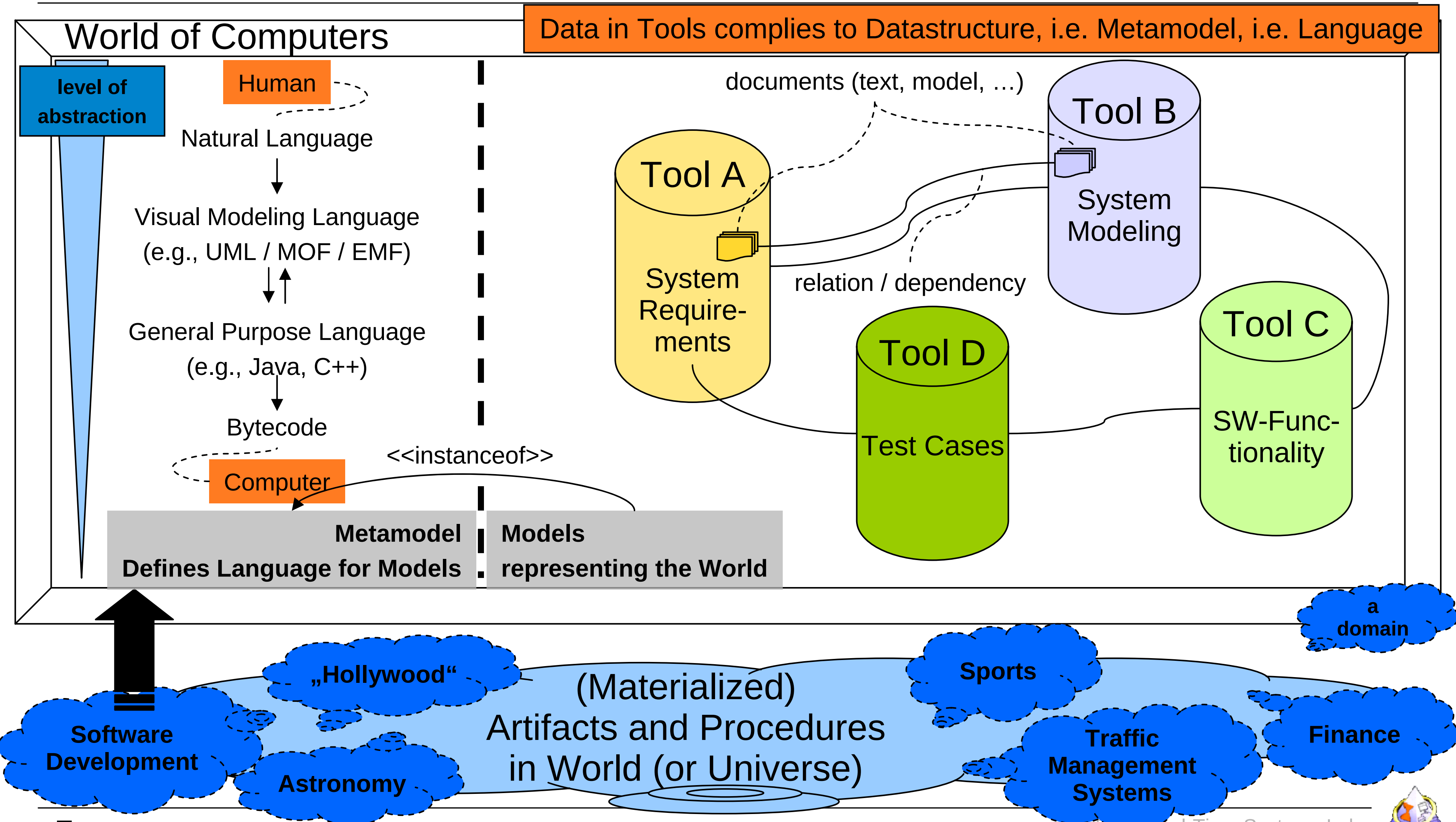


Metamodeling – Overview and Motivation



Data in Tools complies to Datastructure, i.e. Metamodel, i.e. Language

Metamodeling – Overview and Motivation



Data in Tools complies to Datastructure, i.e. Metamodel, i.e. Language

World of Computers

level of abstraction

Human

Natural Language

Visual Modeling Language (e.g., UML / MOF / EMF)

General Purpose Language (e.g., Java, C++)

Bytecode

Computer

<<instanceof>>

Metamodel
Defines Language for Models

Models
representing the World

documents (text, model, …)

relation / dependency

Tool A – System Requirements

Tool B – System Modeling

Tool C – SW-Functionality

Tool D – Test Cases

(Materialized) Artifacts and Procedures in World (or Universe)

a domain

„Hollywood“

Sports

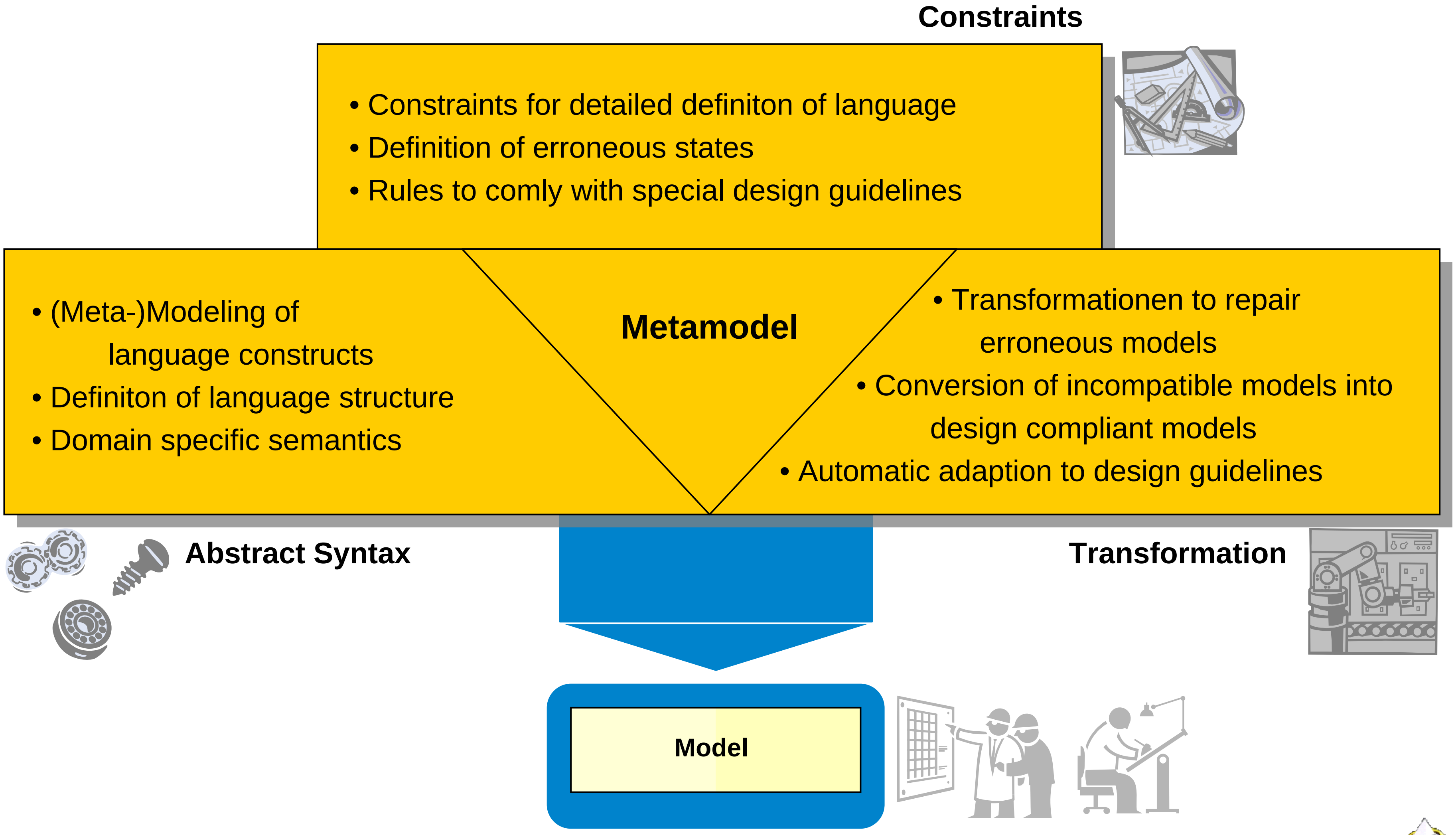
Software Development

Astronomy

Traffic Management Systems

Finance

Metamodeling – Goals



Constraints

- Constraints for detailed definiton of language
- Definition of erroneous states
- Rules to comly with special design guidelines

- (Meta-)Modeling of language constructs
- Definiton of language structure
- Domain specific semantics

Metamodel

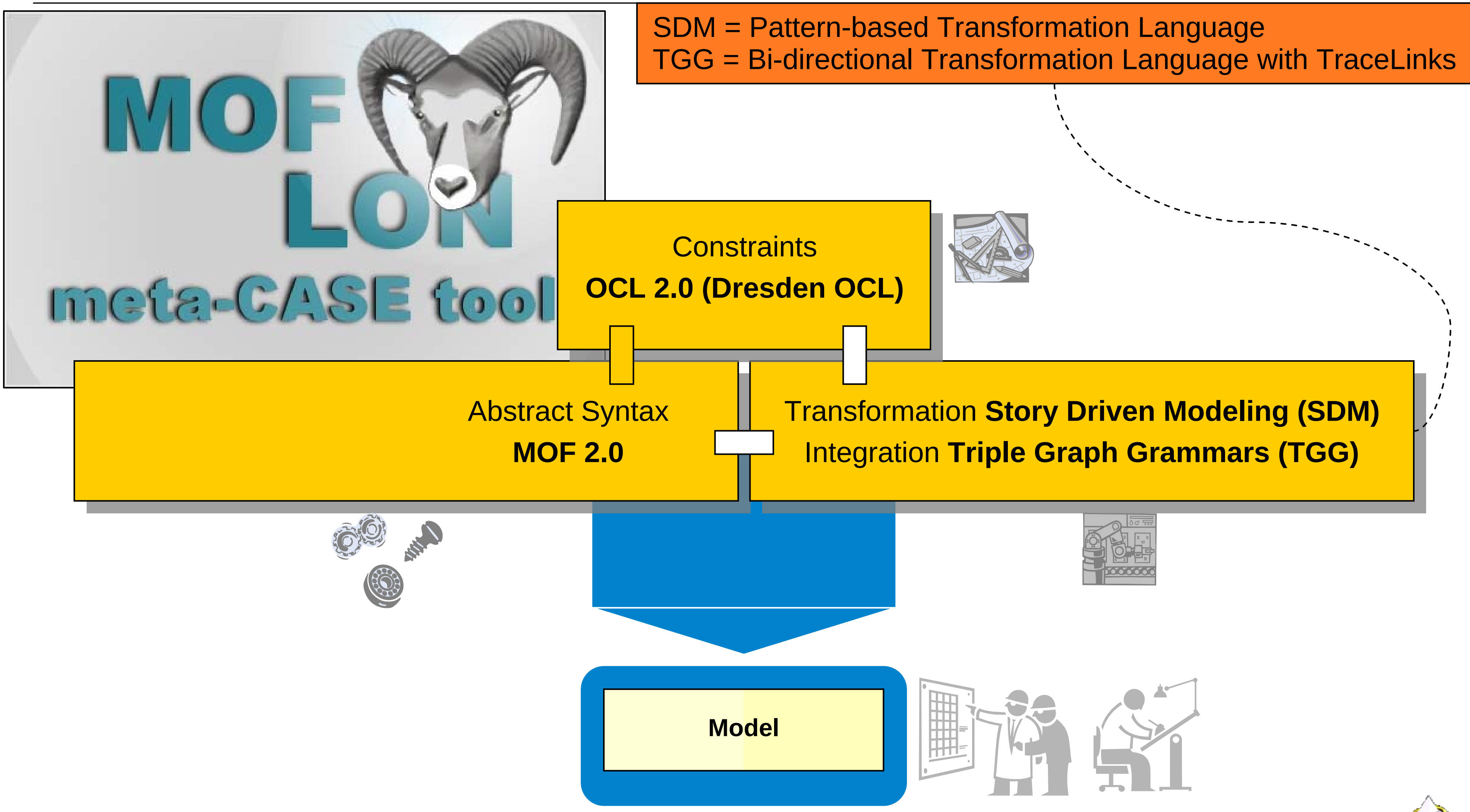
- Transformationen to repair erroneous models
- Conversion of incompatible models into design compliant models
- Automatic adaption to design guidelines

Abstract Syntax

Transformation

Model

A Solution



SDM = Pattern-based Transformation Language
TGG = Bi-directional Transformation Language with TraceLinks

Constraints
OCL 2.0 (Dresden OCL)

Abstract Syntax
MOF 2.0

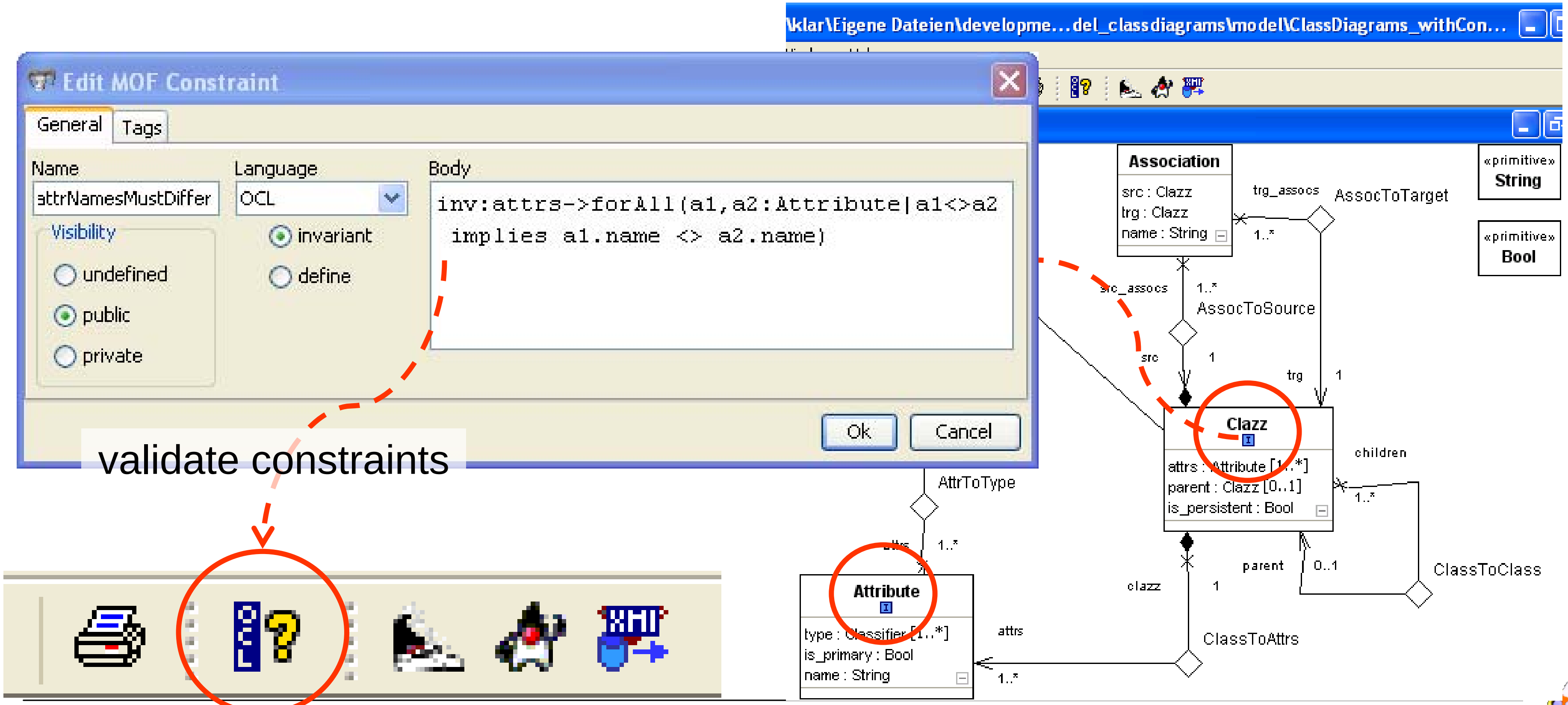
Transformation **Story Driven Modeling (SDM)**
Integration **Triple Graph Grammars (TGG)**

Model

(OCL) Constraints in MOFLON – MOF Editor

- MOF allows to add constraints to every MOF element
- MOFLON has an underlying MOF metamodel repository

→ MOFLON MOF editor may add constraints to elements



Edit MOF Constraint

General | Tags

Name: attrNamesMustDiffer

Language: OCL

Visibility: ○ undefined ● public ○ private

● invariant ○ define

Body:

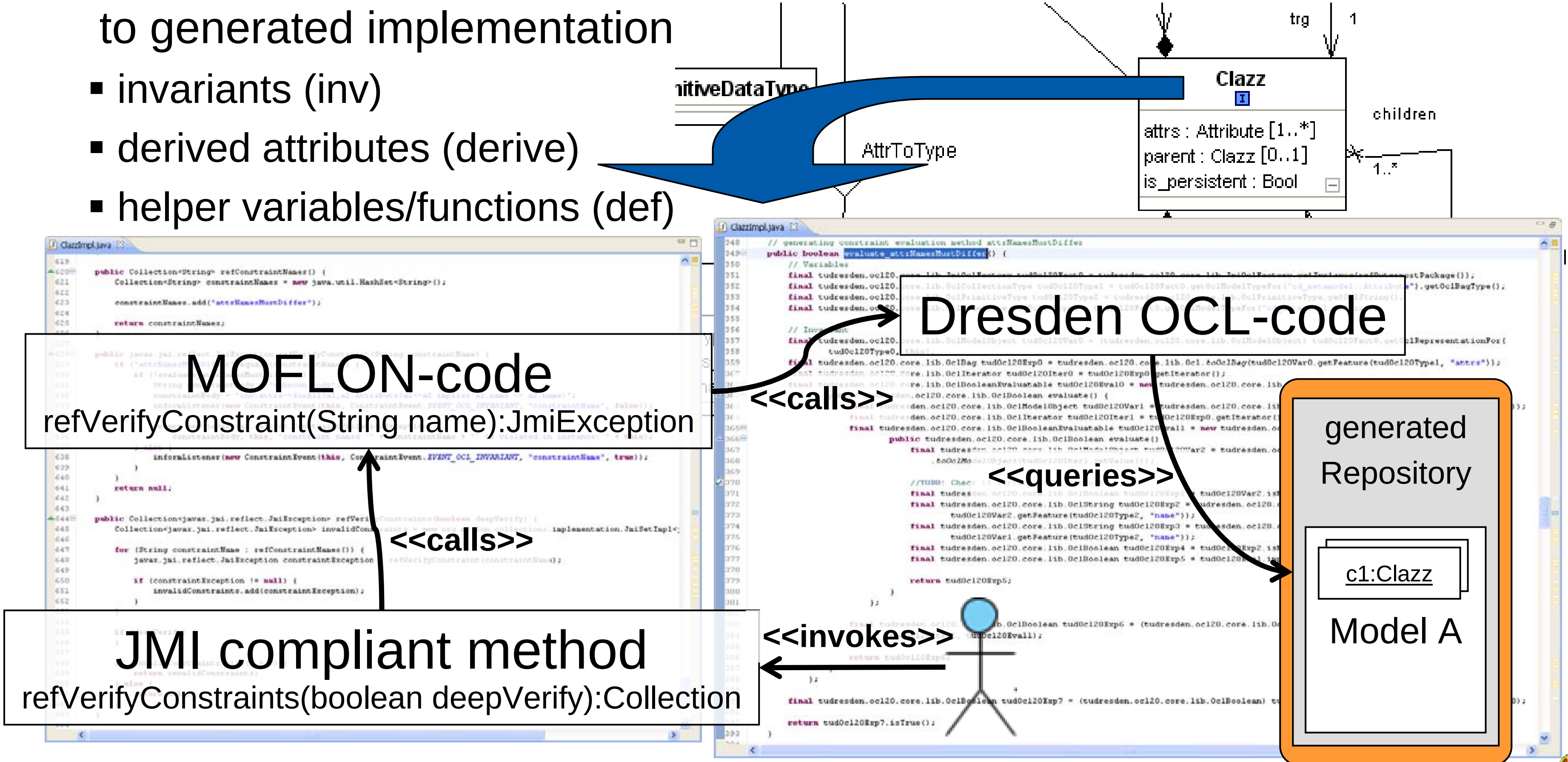
```
inv:attrs->forAll(a1,a2:Attribute|a1<>a2
 implies a1.name <> a2.name)
```

Ok | Cancel

validate constraints

(OCL) Constraints in MOFLON – Generated Implementations

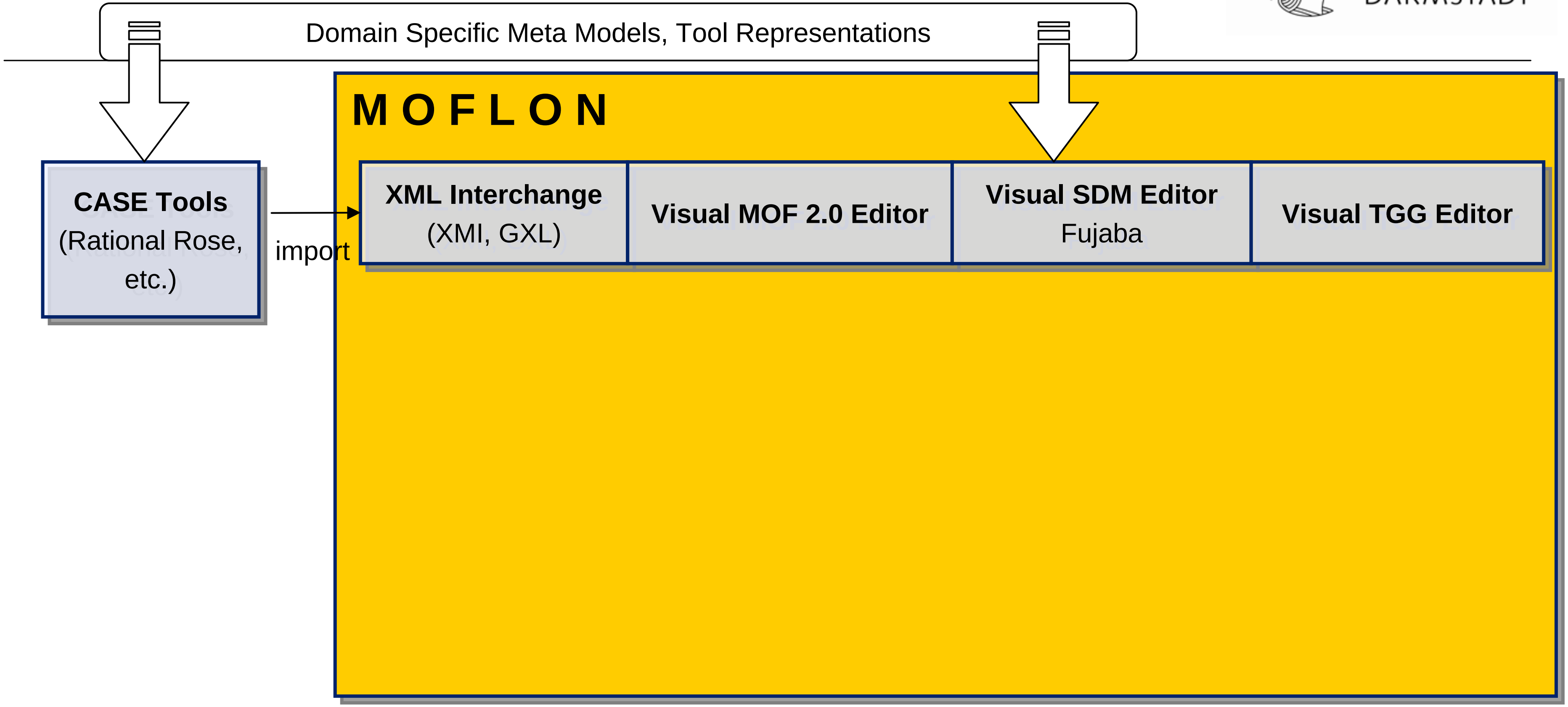
- MOFLON allows to generate metamodel implementations (Java/JMI)
- MOFLON uses Dresden OCL to add constraint code to generated implementation
  - invariants (inv)
  - derived attributes (derive)
  - helper variables/functions (def)



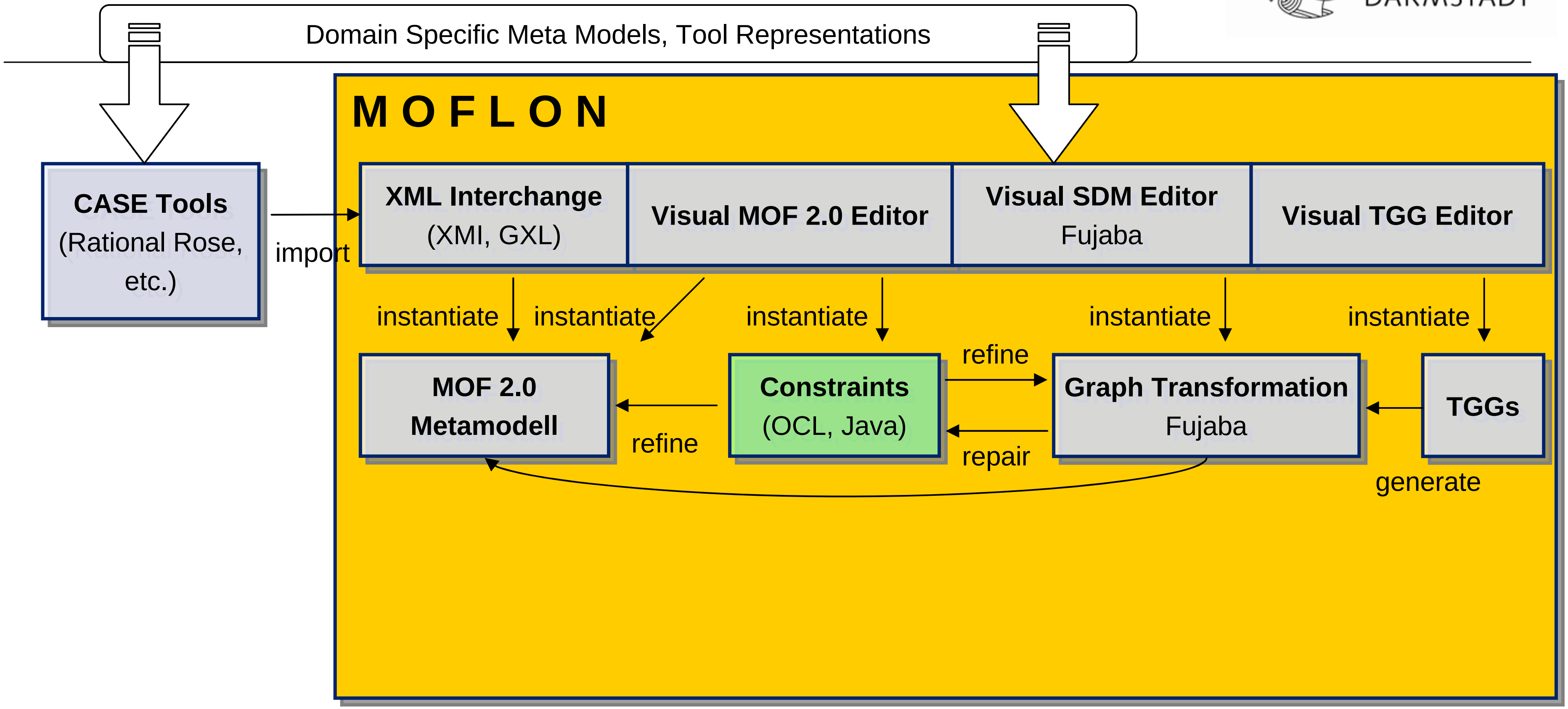
```
    public Collection<String> refConstraintNames() {
        Collection<String> constraintNames = new java.util.HashSet<String>();

        constraintNames.add("attrNamesMustDiffer");

        return constraintNames;
```

```
    // generating constraint evaluation method attrNamesMustDiffer
    public boolean evaluate_attrNamesMustDiffer() {
        // Variables
```

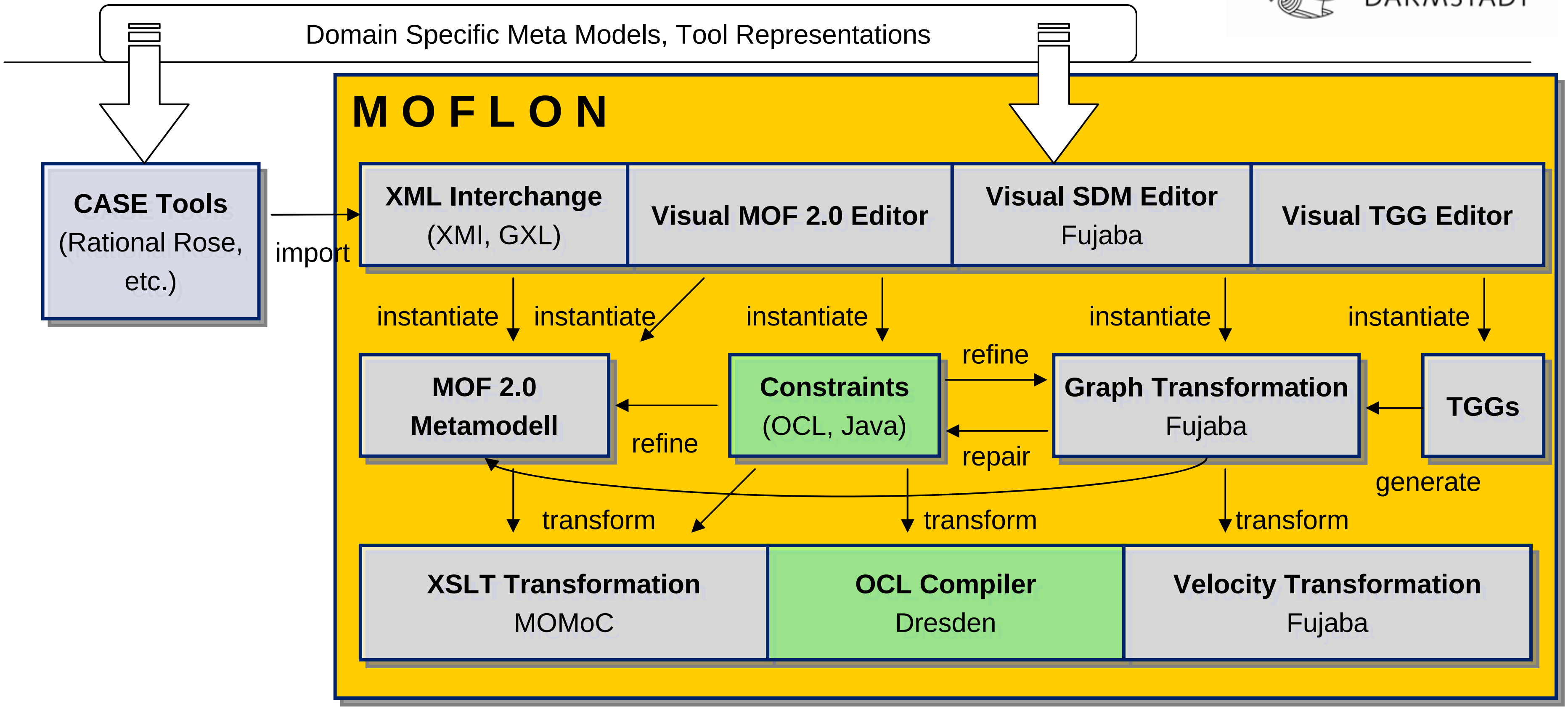
MOFLON – Architecture



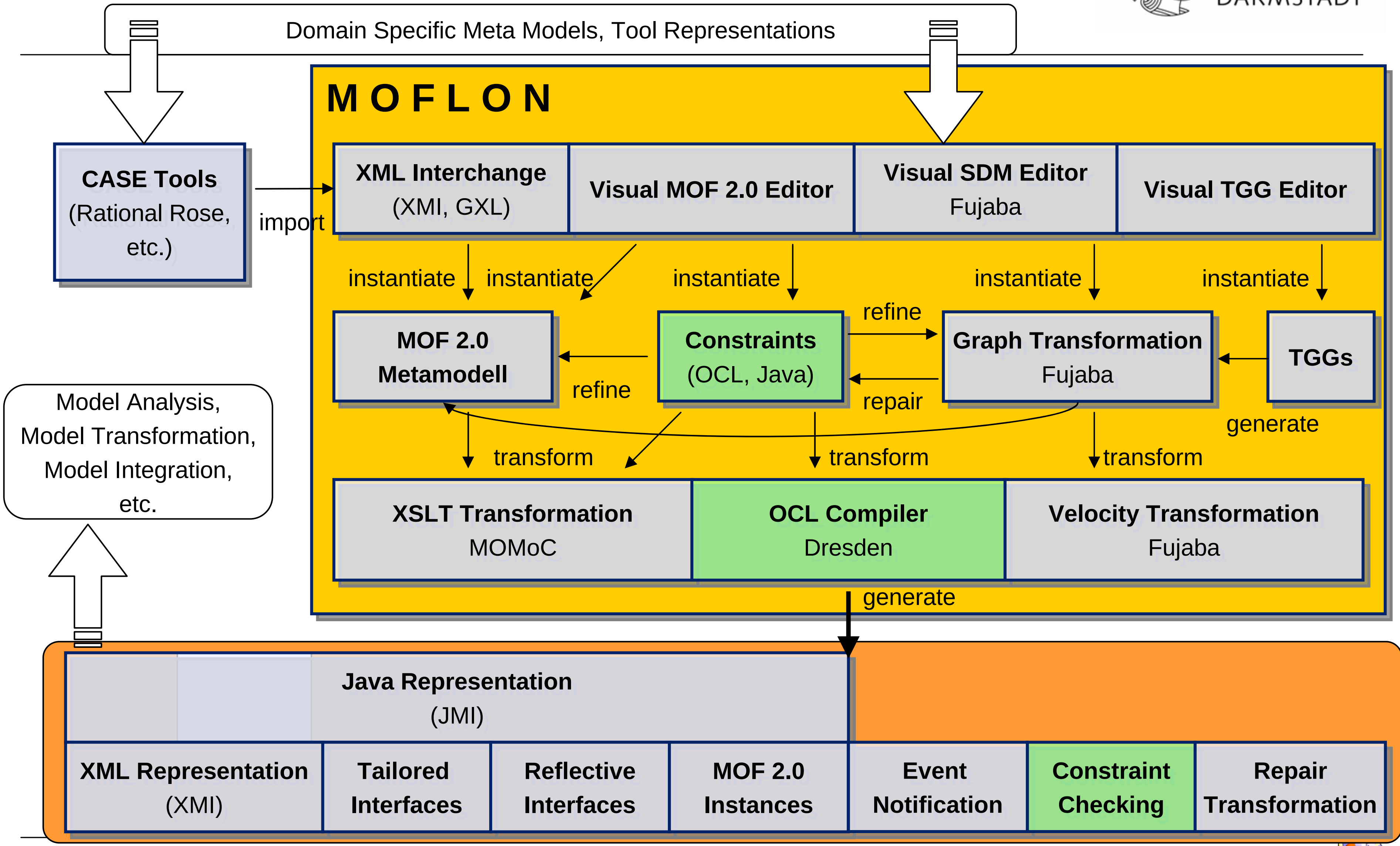
MOFLON – Architecture



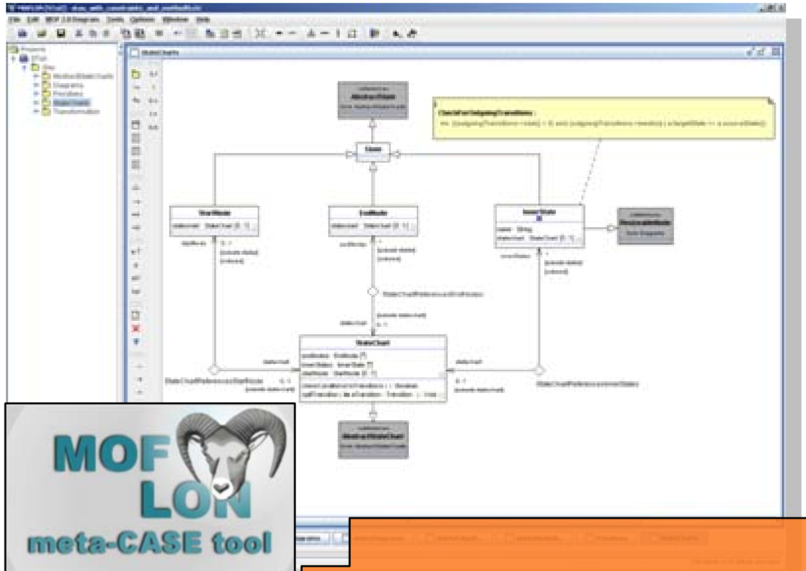
MOFLON – Architecture



MOFLON – Architecture



Case Study – Statechart Editor (STaX)

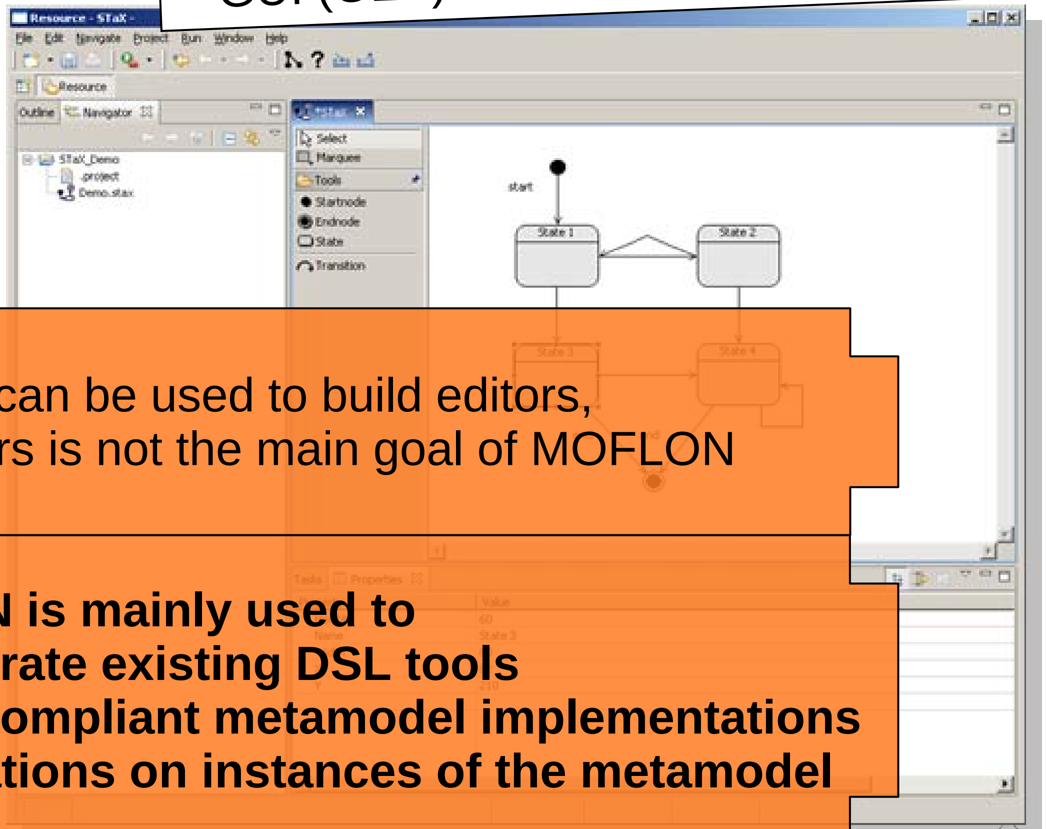


MOFLON meta-CASE tool

+

eclipse GEF

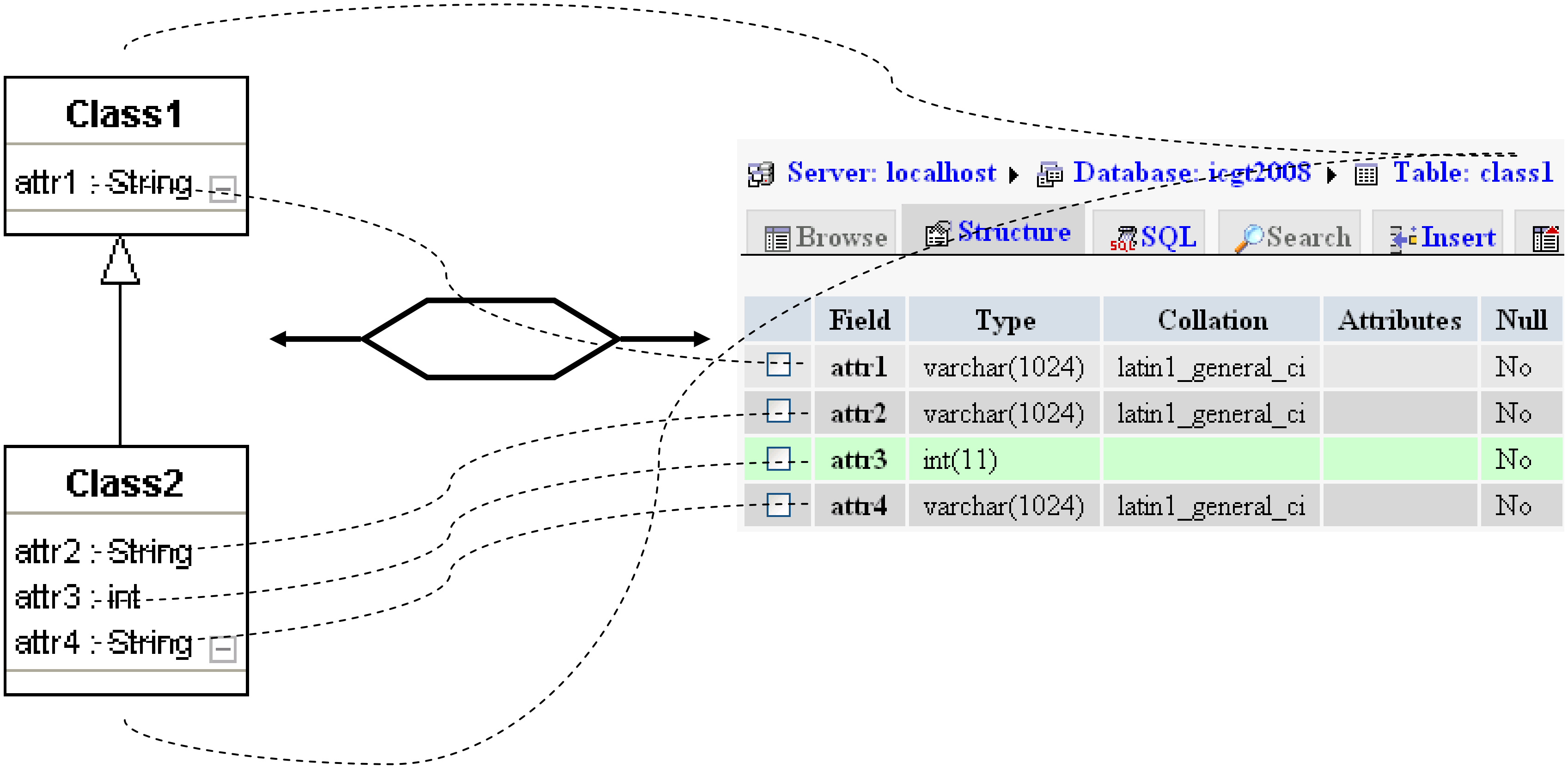
Editor:
- data structure (MOFLON repository)
- GUI (GEF)



MOFLON can be used to build editors,
but building editors is not the main goal of MOFLON

MOFLON is mainly used to
- **integrate existing DSL tools**
- **generate standard compliant metamodel implementations**
- **specify transformations on instances of the metamodel**

Integration Example –
Class diagrams / database schemata



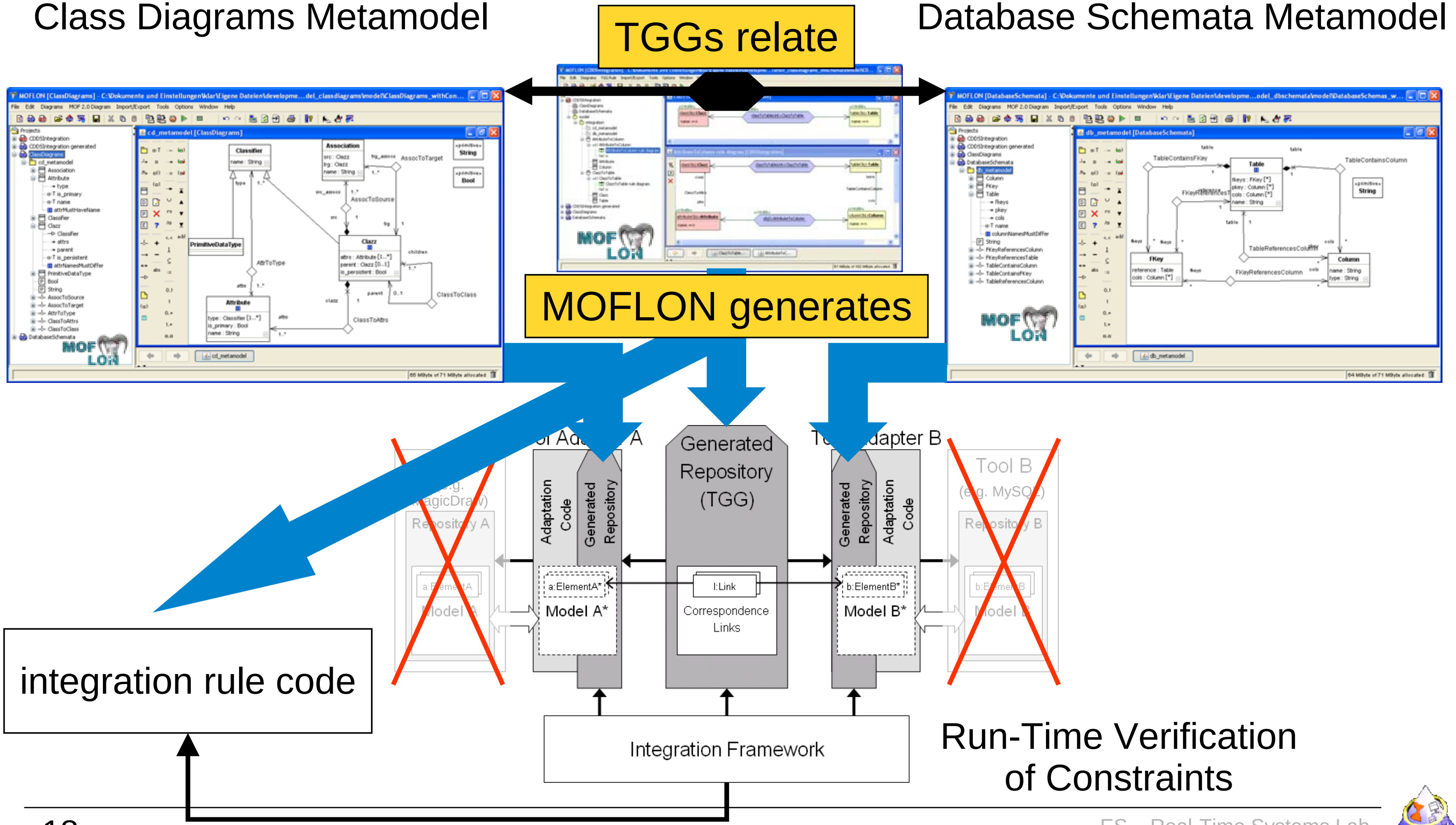
Server: localhost ▸ Database: icgt2008 ▸ Table: class1

	Field	Type	Collation	Attributes	Null
⊟	attr1	varchar(1024)	latin1_general_ci		No
⊟	attr2	varchar(1024)	latin1_general_ci		No
⊟	attr3	int(11)			No
⊟	attr4	varchar(1024)	latin1_general_ci		No

domain specific language,
e.g. Class Diagrams

domain specific language,
e.g. Database Schemata

Tool Integration Scenario (CD / DS)



Class Diagrams Metamodel

TGGs relate

Database Schemata Metamodel

MOFLON generates

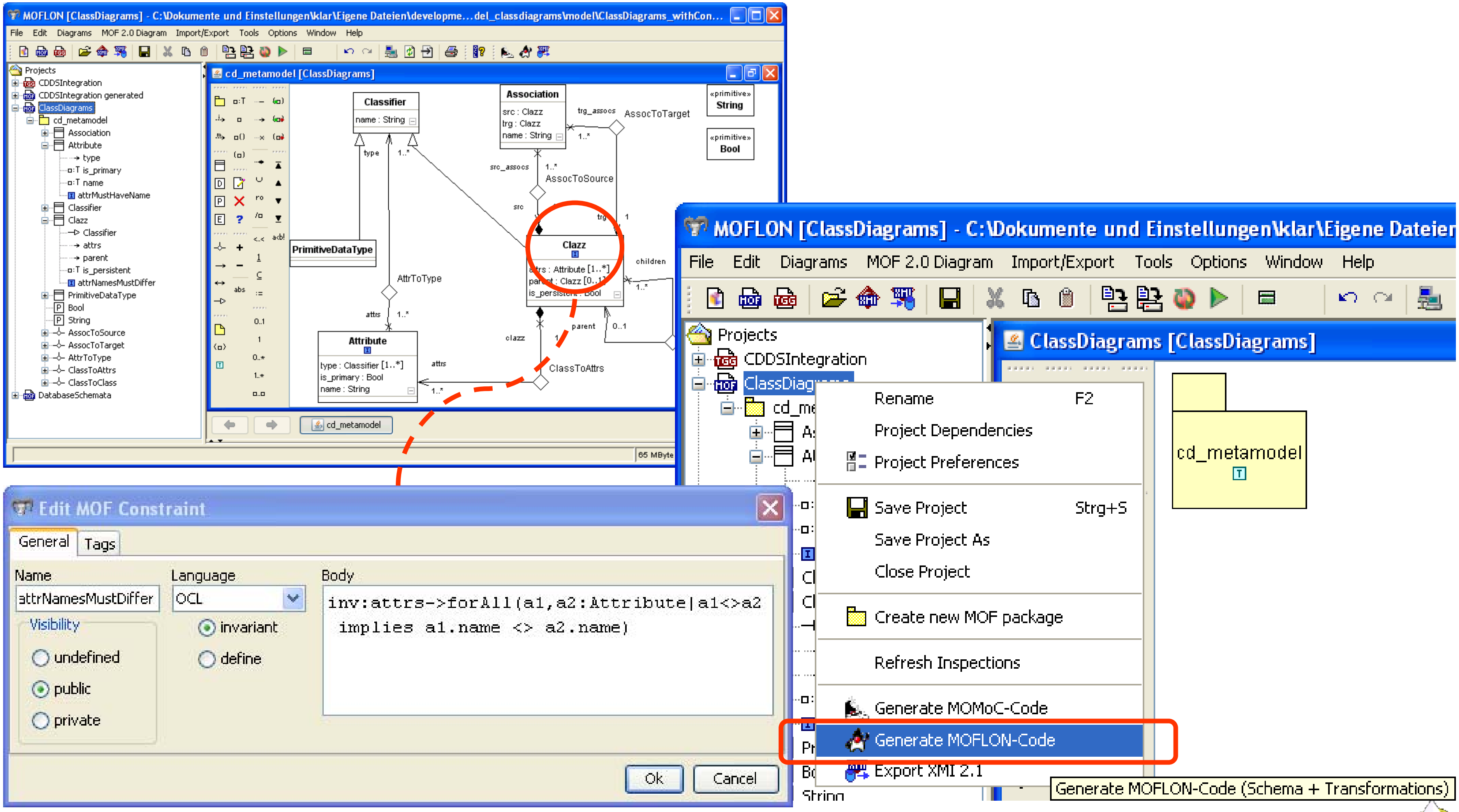
integration rule code

Run-Time Verification of Constraints

Outline

- Metamodels are Languages – A Motivation
- Metamodeling – Goals
- MOFLON – OCL
- MOFLON – Architecture
- MOFLON – Scenarios
- **Demo (Integration Scenario – TiE-CDDS)**
- Future Activities

TiE-CDDS – Focus on Constraints in CD (1)
Generate Code from MOF model (CD metamodel)



Edit MOF Constraint

Name: attrNamesMustDiffer

Language: OCL

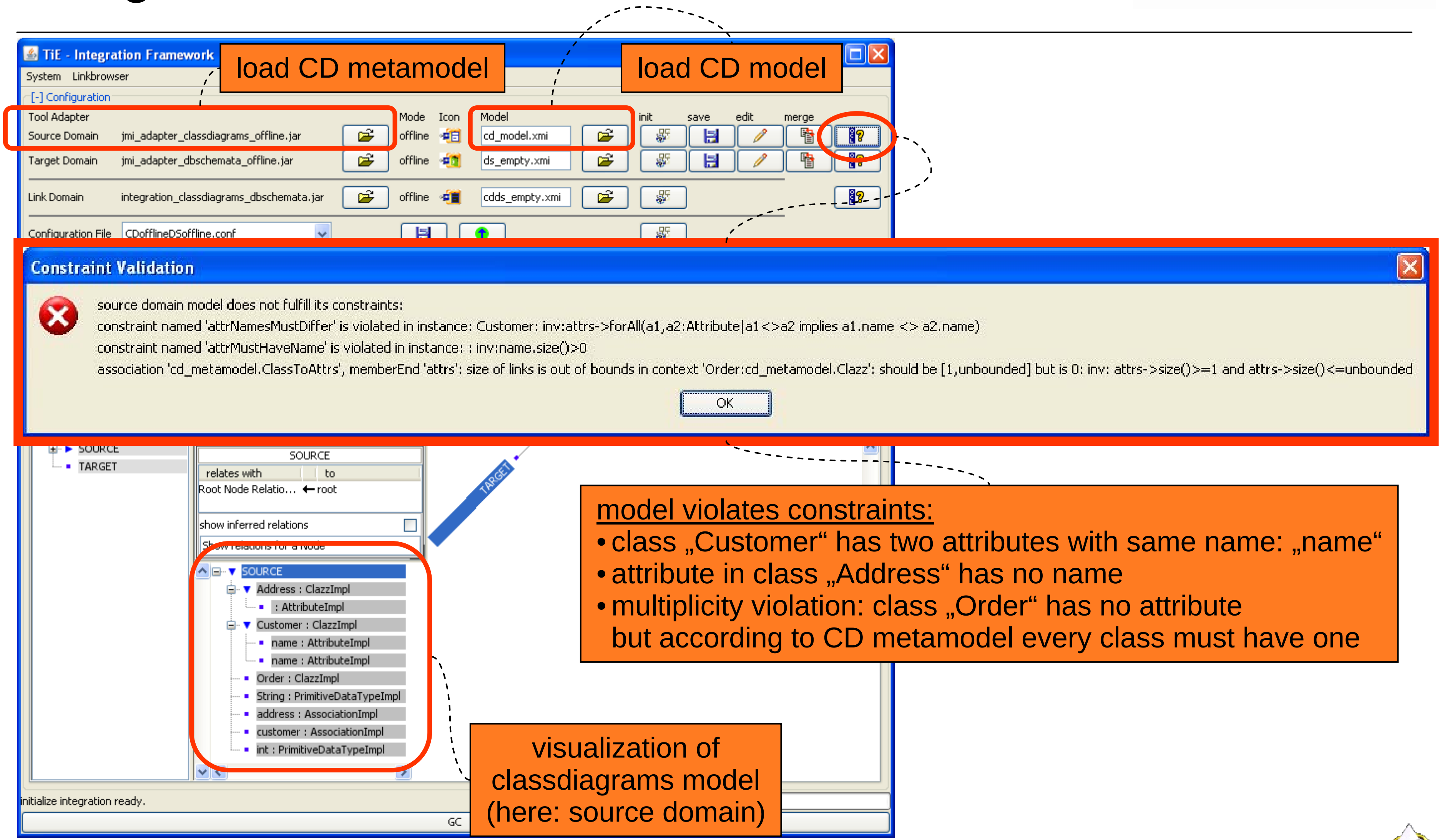
Visibility: ○ undefined ● public ○ private

● invariant ○ define

Body:

```
inv:attrs->forAll(a1,a2:Attribute|a1<>a2
 implies a1.name <> a2.name)
```

TiE-CDDS – Focus on Constraints in CD (2) Integration Framework



Tool Adapter

Source Domain jmi_adapter_classdiagrams_offline.jar offline cd_model.xmi

Target Domain jmi_adapter_dbschemata_offline.jar offline ds_empty.xmi

Link Domain integration_classdiagrams_dbschemata.jar offline cdds_empty.xmi

Configuration File CDofflineDSoffline.conf

Root Node Relatio… ← root

- SOURCE
  - Address : ClazzImpl
    - : AttributeImpl
  - Customer : ClazzImpl
    - name : AttributeImpl
    - name : AttributeImpl
  - Order : ClazzImpl
  - String : PrimitiveDataTypeImpl
  - address : AssociationImpl
  - customer : AssociationImpl
  - int : PrimitiveDataTypeImpl

initialize integration ready.

Constraint Validation

```
source domain model does not fulfill its constraints:
constraint named 'attrNamesMustDiffer' is violated in instance: Customer: inv:attrs->forAll(a1,a2:Attribute|a1<>a2 implies a1.name <> a2.name)
constraint named 'attrMustHaveName' is violated in instance: : inv:name.size()>0
association 'cd_metamodel.ClassToAttrs', memberEnd 'attrs': size of links is out of bounds in context 'Order:cd_metamodel.Clazz': should be [1,unbounded] but is 0: inv: attrs->size()>=1 and attrs->size()<=unbounded
```

OK

load CD metamodel

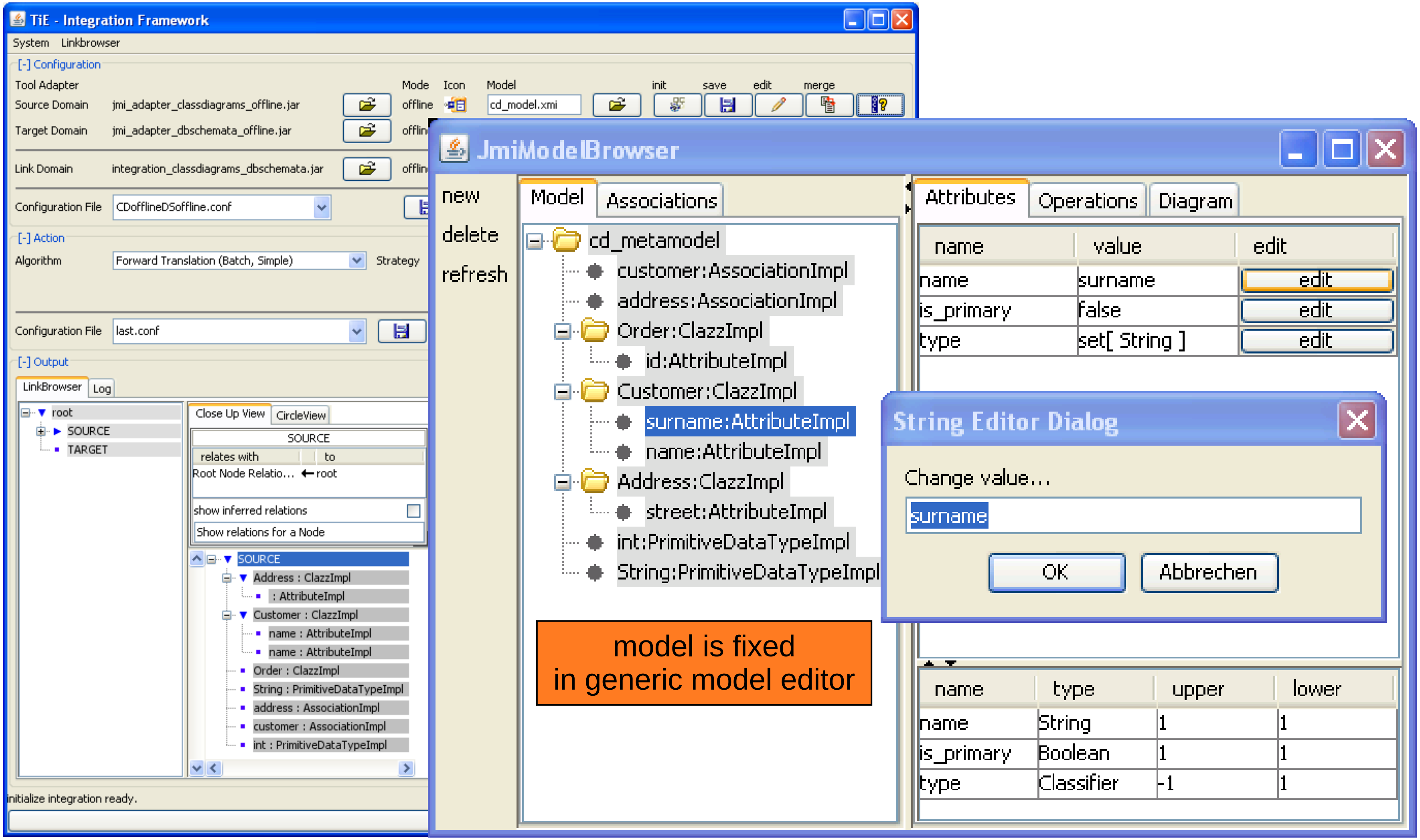
load CD model

model violates constraints:

- class „Customer“ has two attributes with same name: „name“
- attribute in class „Address“ has no name
- multiplicity violation: class „Order“ has no attribute but according to CD metamodel every class must have one

visualization of classdiagrams model (here: source domain)

TiE-CDDS – Focus on Constraints in CD (3)
Model Browser



[-] Configuration

Tool Adapter		Mode	Icon	Model
Source Domain	jmi_adapter_classdiagrams_offline.jar	offline		cd_model.xmi
Target Domain	jmi_adapter_dbschemata_offline.jar	offlin		
Link Domain	integration_classdiagrams_dbschemata.jar	offlin		

Configuration File: CDofflineDSoffline.conf

[-] Action

Algorithm: Forward Translation (Batch, Simple) Strategy

Configuration File: last.conf

[-] Output

LinkBrowser | Log

- root
  - SOURCE
  - TARGET

Close Up View | CircleView

SOURCE

relates with	to
Root Node Relatio…	← root

show inferred relations □

Show relations for a Node

- SOURCE
  - Address : ClazzImpl
    - : AttributeImpl
  - Customer : ClazzImpl
    - name : AttributeImpl
    - name : AttributeImpl
  - Order : ClazzImpl
  - String : PrimitiveDataTypeImpl
  - address : AssociationImpl
  - customer : AssociationImpl
  - int : PrimitiveDataTypeImpl

initialize integration ready.

- cd_metamodel
  - customer:AssociationImpl
  - address:AssociationImpl
  - Order:ClazzImpl
    - id:AttributeImpl
  - Customer:ClazzImpl
    - surname:AttributeImpl
    - name:AttributeImpl
  - Address:ClazzImpl
    - street:AttributeImpl
  - int:PrimitiveDataTypeImpl
  - String:PrimitiveDataTypeImpl

name	value	edit
name	surname	edit
is_primary	false	edit
type	set[String]	edit

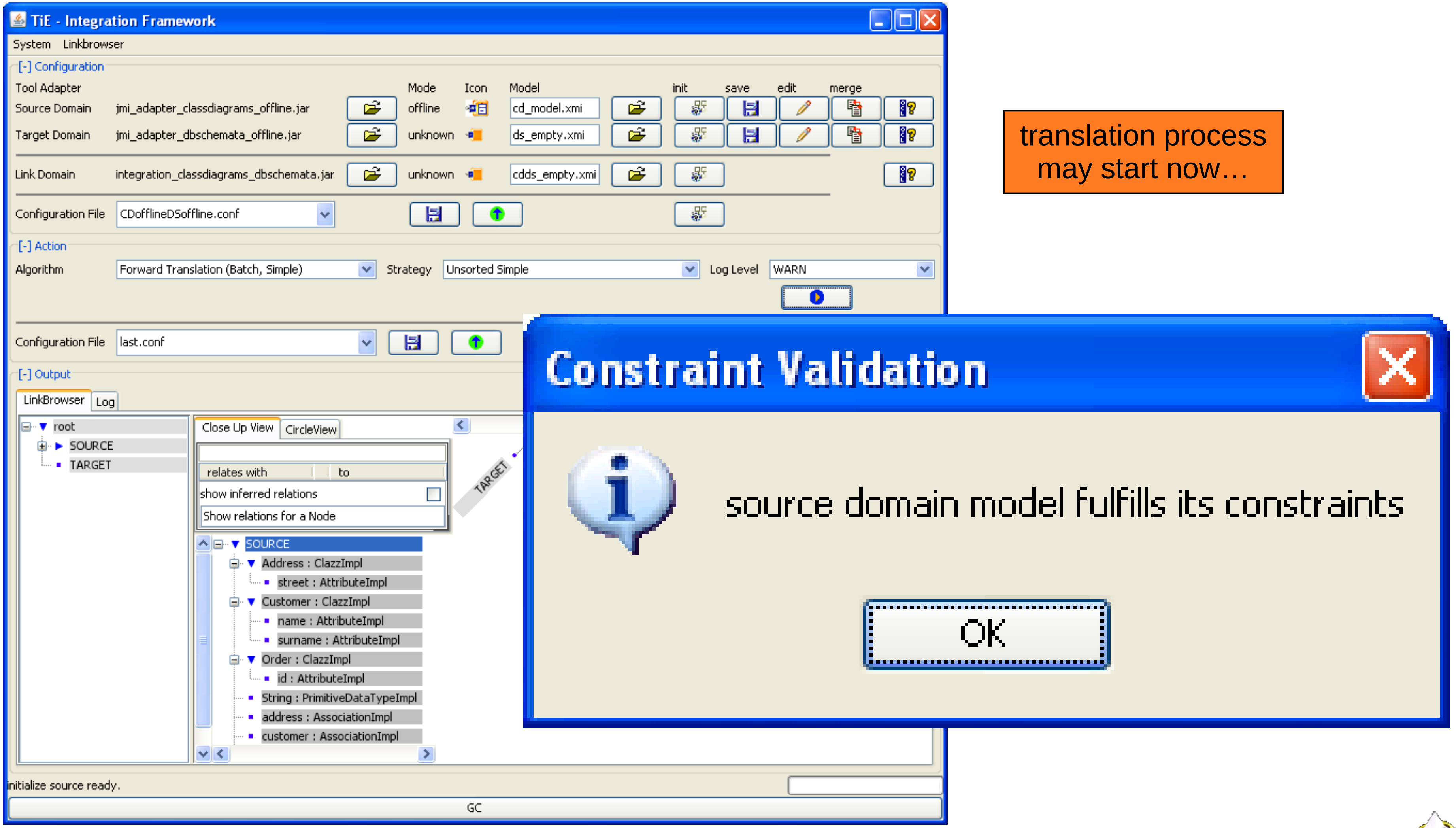
Change value…

surname

name	type	upper	lower
name	String	1	1
is_primary	Boolean	1	1
type	Classifier	-1	1

model is fixed
in generic model editor

TiE-CDDS – Focus on Constraints in CD (4) Integration Framework



TiE - Integration Framework

System Linkbrowser

[-] Configuration

Tool Adapter		Mode	Icon	Model
Source Domain	jmi_adapter_classdiagrams_offline.jar	offline		cd_model.xmi
Target Domain	jmi_adapter_dbschemata_offline.jar	unknown		ds_empty.xmi
Link Domain	integration_classdiagrams_dbschemata.jar	unknown		cdds_empty.xmi

Configuration File: CDofflineDSoffline.conf

[-] Action

Algorithm: Forward Translation (Batch, Simple)

Strategy: Unsorted Simple

Log Level: WARN

Configuration File: last.conf

[-] Output

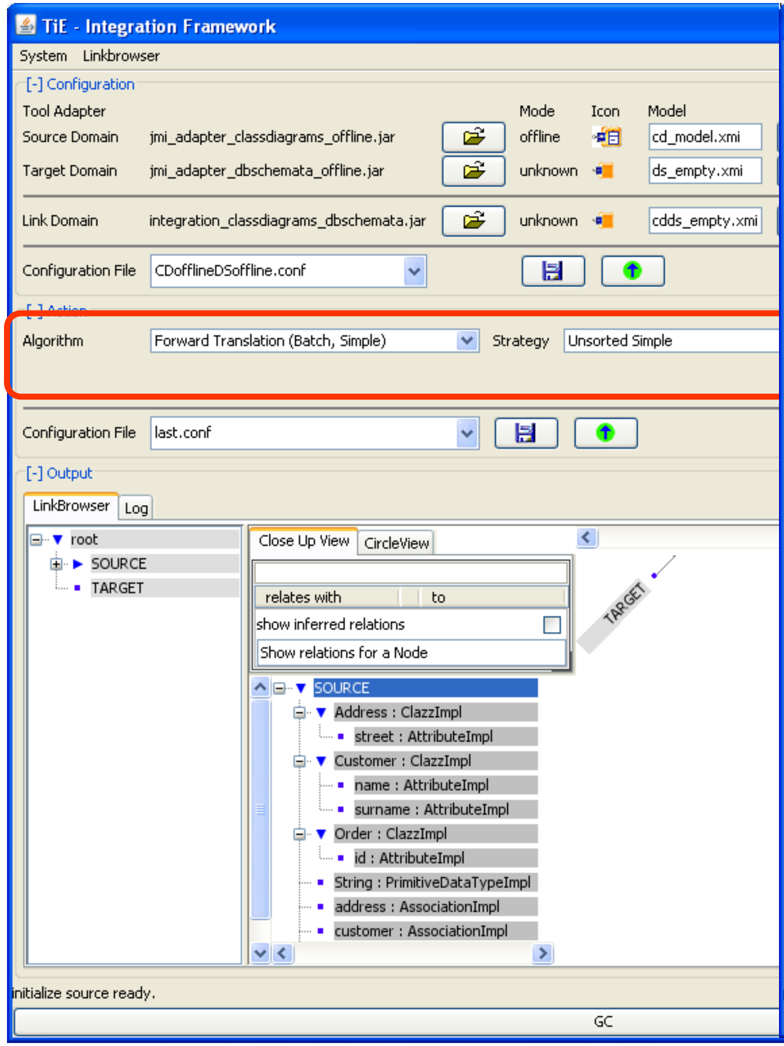
initialize source ready.

Constraint Validation

source domain model fulfills its constraints

translation process may start now…

TiE-CDDS – Focus on Constraints in CD (5)
Forward Translation to DB representation



Tool Adapter		Mode	Icon	Model
Source Domain	jmi_adapter_classdiagrams_offline.jar	offline		cd_model.xmi
Target Domain	jmi_adapter_dbschemata_offline.jar	unknown		ds_empty.xmi
Link Domain	integration_classdiagrams_dbschemata.jar	unknown		cdds_empty.xmi

Configuration File CDofflineDSoffline.conf

Algorithm Forward Translation (Batch, Simple) Strategy Unsorted Simple

Configuration File last.conf

- root
  - SOURCE
  - TARGET

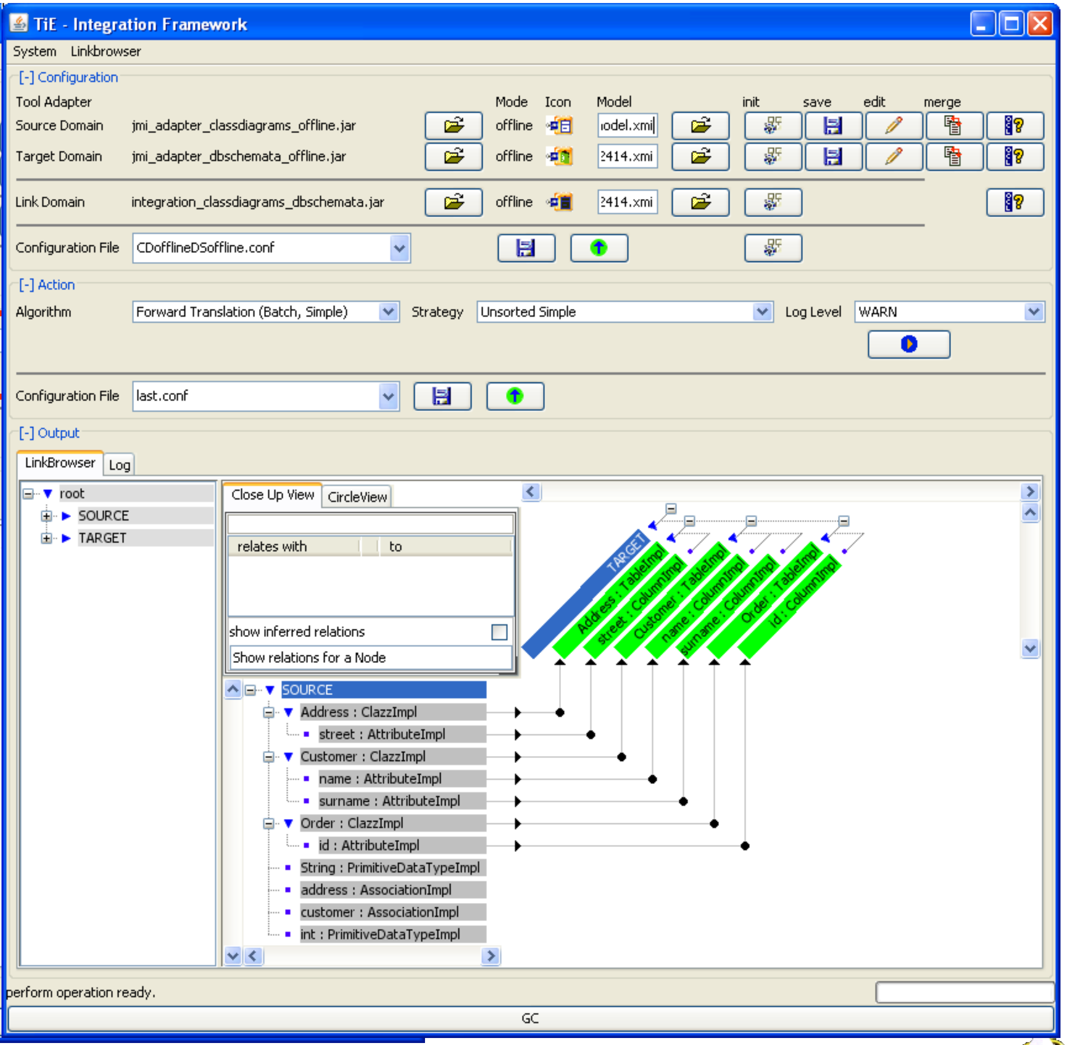
relates with	to

show inferred relations ☐

- SOURCE
  - Address : ClazzImpl
    - street : AttributeImpl
  - Customer : ClazzImpl
    - name : AttributeImpl
    - surname : AttributeImpl
  - Order : ClazzImpl
    - id : AttributeImpl
  - String : PrimitiveDataTypeImpl
  - address : AssociationImpl
  - customer : AssociationImpl

TARGET

initialize source ready.



Tool Adapter		Mode	Icon	Model
Source Domain	jmi_adapter_classdiagrams_offline.jar	offline		ıodel.xmi
Target Domain	jmi_adapter_dbschemata_offline.jar	offline		2414.xmi
Link Domain	integration_classdiagrams_dbschemata.jar	offline		2414.xmi

Configuration File CDofflineDSoffline.conf

Algorithm Forward Translation (Batch, Simple) Strategy Unsorted Simple Log Level WARN

Configuration File last.conf

- root
  - SOURCE
  - TARGET

relates with	to

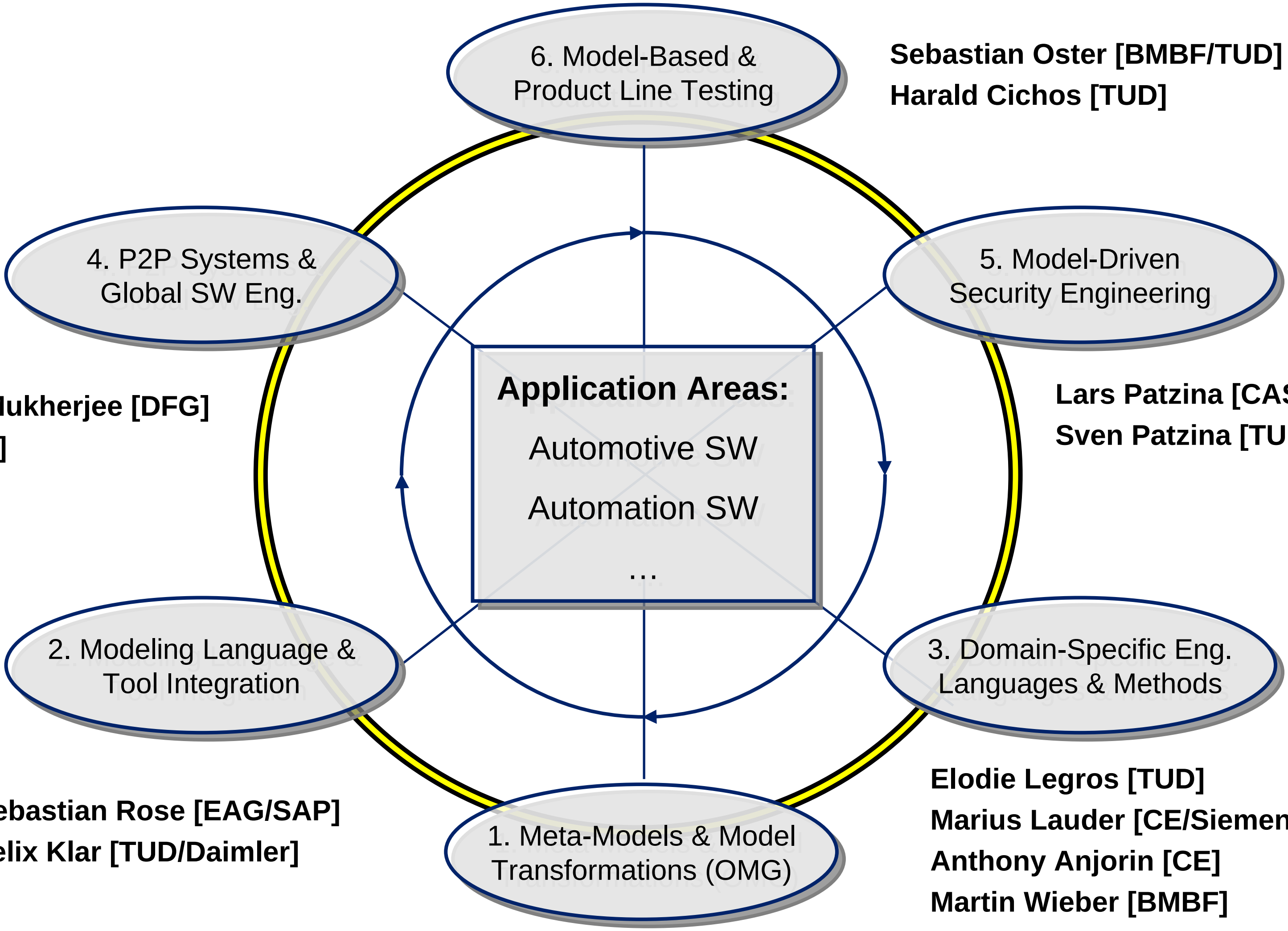
show inferred relations ☐

- SOURCE
  - Address : ClazzImpl
    - street : AttributeImpl
  - Customer : ClazzImpl
    - name : AttributeImpl
    - surname : AttributeImpl
  - Order : ClazzImpl
    - id : AttributeImpl
  - String : PrimitiveDataTypeImpl
  - address : AssociationImpl
  - customer : AssociationImpl
  - int : PrimitiveDataTypeImpl

perform operation ready.

Model-Driven Software Development at Real-Time Systems Lab

TECHNISCHE UNIVERSITÄT DARMSTADT



6. Model-Based & Product Line Testing
4. P2P Systems & Global SW Eng.
5. Model-Driven Security Engineering
Application Areas:
Automotive SW
Automation SW
...
2. Modeling Language & Tool Integration
3. Domain-Specific Eng. Languages & Methods
1. Meta-Models & Model Transformations (OMG)

Sebastian Oster [BMBF/TUD]
Harald Cichos [TUD]

Patrick Mukherjee [DFG]
NN [DFG]

Lars Patzina [CASED]
Sven Patzina [TUD]

Sebastian Rose [EAG/SAP]
Felix Klar [TUD/Daimler]

Elodie Legros [TUD]
Marius Lauder [CE/Siemens]
Anthony Anjorin [CE]
Martin Wieber [BMBF]

Future Work – OCL

- Activate more features of Dresden OCL in MOFLON
  - MOF editor
    - User friendly OCL syntax checking
    - OCL expression completion
  - MOFLON code generator
    - Initial Values (init)
  - Queries?
  - …

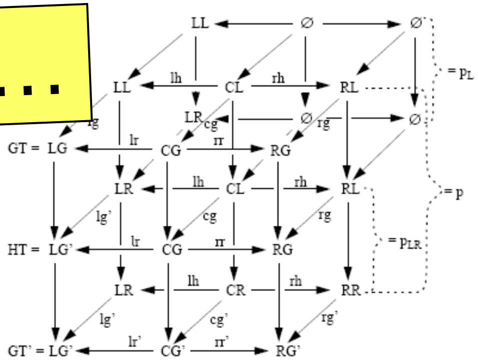
- We bootstrap our MOFLON MOF Metamodel periodically
  - Add more OCL constraints to our MOF Metamodel
  - Regenerate MOFLON MOF implementation
  - Activate constraint checking in MOFLON
    → Model Verification

Further reading

- **A. Königs, A. Schürr:** "Tool Integration with Triple Graph Grammars - A Survey"**, in: R. Heckel (ed.), Proceedings of the SegraVis School on Foundations of Visual Modelling Techniques, Amsterdam: Elsevier Science Publ., 2006; Electronic Notes in Theoretical Computer Science, Vol. 148, 113-150.**
- **F. Klar, S. Rose, A. Schürr:** "TiE - A Tool Integration Environment"**, Proceedings of the 5th ECMDA Traceability Workshop, 2009; CTIT Workshop Proceedings, Vol. WP09-09, 39-48**
- **F. Klar, S. Rose, A. Schürr:** "A Meta-Model-Driven Tool Integration Development Process"**, Proceedings of the 2nd International United Information Systems Conference, 2008; Lecture Notes in Business Information Processing, 201-212.**
- **C. Amelunxen, A. Königs, T. Rötschke, A. Schürr:** "MOFLON: A Standard-Compliant Metamodeling Framework with Graph Transformations"**, in: A. Rensink, J. Warmer (eds.), Model Driven Architecture - Foundations and Applications: Second European Conference, Heidelberg: Springer Verlag, 2006; Lecture Notes in Computer Science (LNCS), Vol. 4066, Springer Verlag, 361-375.**
- **A. Königs:** "Model Integration and Transformation - A Triple Graph Grammar-based QVT Implementation"**, Technische Universität Darmstadt, Phd Thesis, 2009.**

Time for questions and discussion

Thank you for your attention…

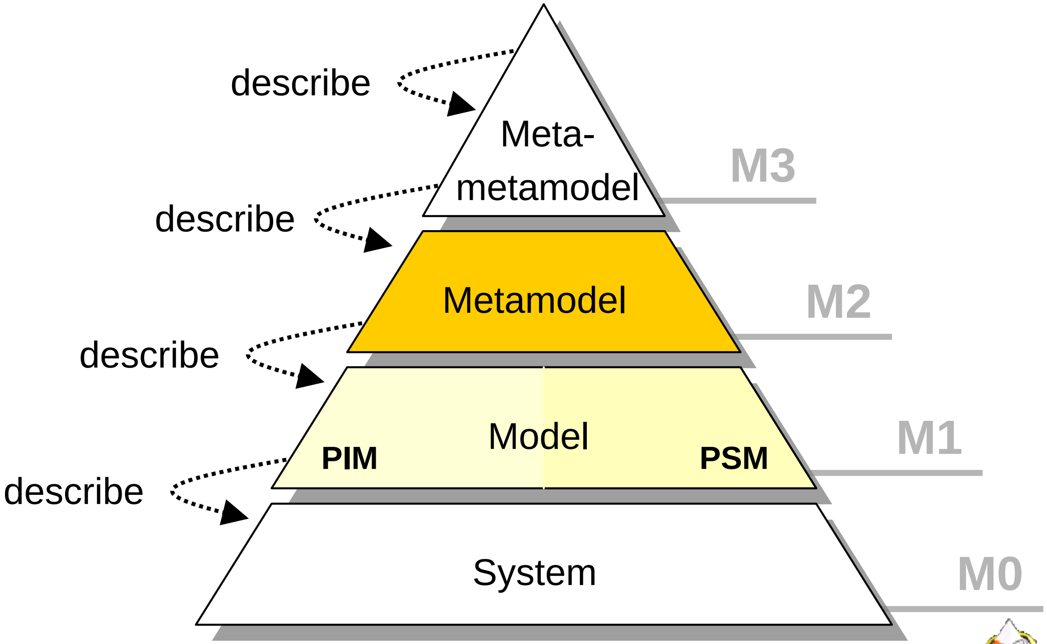


http://www.moflon.org

Backup Slides

Motivation

- Models are widely used in engineering disciplines
- Need for tool support that enables model-editing
- Domain experts want domain specific languages (DSL)
  → domain specific models
- do not build model editors from scratch each time
  → reuse functionality
  → use meta-information



MOFLON – Main Features

- MOF2.0 editor (draw metamodels that comply to MOF2.0 standard)
  → build Domain Specific Languages (DSLs)
- based on the CASE-tool framework Fujaba
- possibility to extend MOFLON by own plugins
- interoperabilty (import / export)
- transform metamodel instances with model transformations (SDM, TGG)
- generate code (JMI-compliant) from DSLs
- instantiate models of the DSL (= repositories)
- basic editing support for generated repositories
- Standard compliance!

Related Approaches

	standards	approaches based on graph-/modeltransformation						classic meta-CASE approaches					text based approaches		
	MOF, OCL, QVT	MOFLON	Fujaba & TGG	Progres & TGG	GME & GReAT	EMF & Tefkat	AToM^3	MetaEdit+	Microsoft DSL	EMF & GMF	Pounamu	DiaGen	EBNF & TXL	SQL	XML
Abstract syntax	+	+	+	+	o	o	o	+	+	o	+	+	+	o	+
Concrete syntax	--	--	--	+	+	--	+	+	+	+	+	+	--	--	--
Static semantics	+	+	o	+	+	+	o	o	--	+	o	+	o	o	--
Dynamic semantics	+	+	+	+	+	+	+	o	o	--	--	--	+	--	o
Model analysis	+	+	+	+	+	o	+	o	--	+	--	o	+	o	+
Model transformation	+	+	+	+	+	+	+	o	--	--	--	o	+	o	+
Model integration	+	+	+	+	o	+	--	--	--	--	--	--	o	--	o
Acceptability	+	+	o	--	o	+	--	+	--	o	+	o	o	+	+
Scaleability	+	+	--	o	--	o	--	o	--	--	--	--	--	--	o
Tool availability	--	o	o	+	+	+	+	+	o	o	+	+	+	+	o
Expressiveness	+	+	o	+	+	o	o	o	o	o	o	o	+	o	o

from Amelunxen, Königs, Rötschke, and Schürr,
„MOSL: Composing a Visual Language for a Metamodeling Framework“
in IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC 2006), September, 2006, 81-84