25.3.1. The Metamodeling Architecture of MetaCASE Tool MOFLON

From: 10 Jahre Dresden-OCL – Workshop
http://dresden-ocl.sourceforge.net/
http://dresden-ocl.sourceforge.net/10years.html

ES Real-Time Systems Lab
Prof. Dr. rer. nat. Andy Schürr
Dept. of Electrical Engineering and Information Technology
Dept. of Computer Science (adjunct Professor)

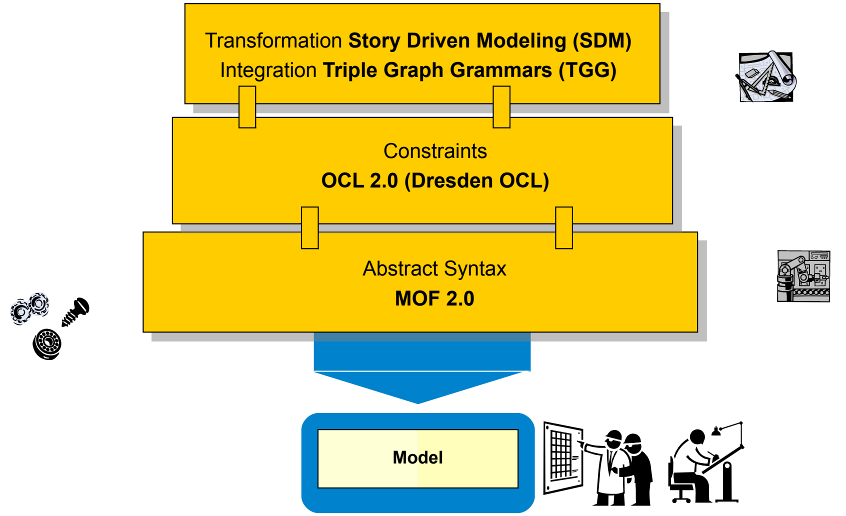
www.es.tu-darmstadt.de

Felix Klar

Felix.Klar@es.tu-darmstadt.de

15.10.2009

Metamodel Architecture of MOFLON

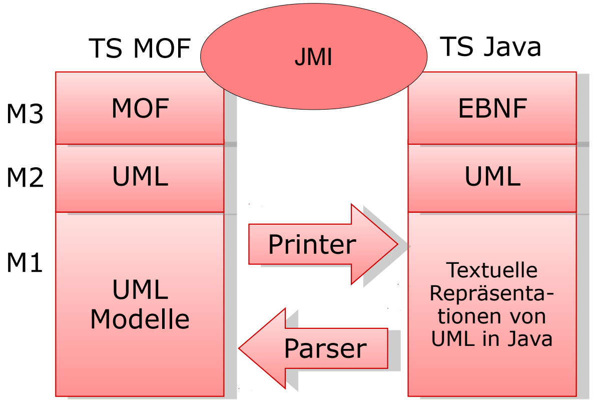

Transformation **Story Driven Modeling (SDM)**
Integration **Triple Graph Grammars (TGG)**
Constraints
OCL 2.0 (Dresden OCL)
Abstract Syntax
MOF 2.0
Model

MOFLON MetaCASE – 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

Einschub: JMI: Transformative TS-Brücke für MOF und Java, Sprache UML

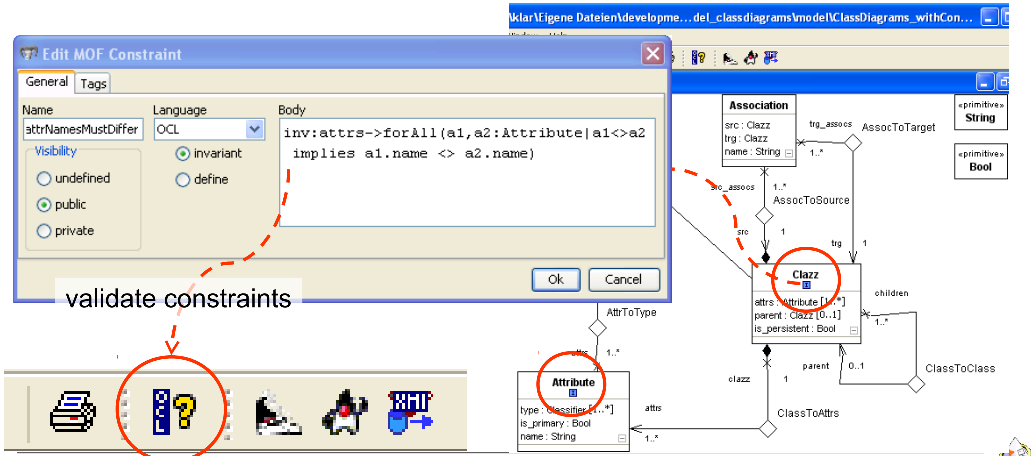
Java Metadata Interchange (JMI) ist eine TS-Halb-Brücke für MOF und EBNF-Space, für die Sprache UML



(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



Name: attrNamesMustDiffer

Language: OCL

Body:

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

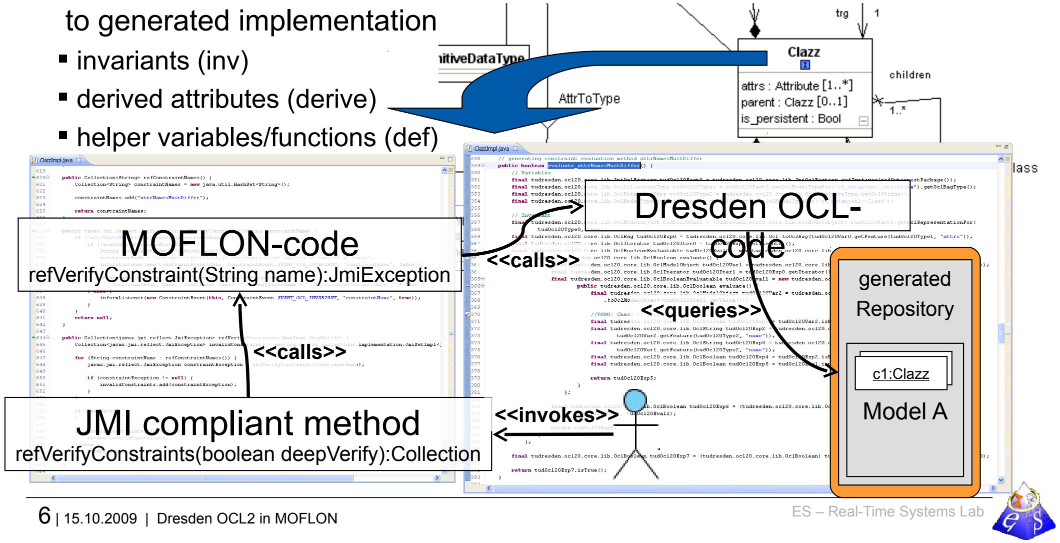
Visibility: ○ undefined ● public ○ private

● invariant ○ define

validate constraints

(OCL) Constraints in MOFLON – Generated Implementations

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



MOFLON-code

refVerifyConstraint(String name):JmiException

Dresden OCL-code

<<calls>>

<<queries>>

<<calls>>

generated Repository

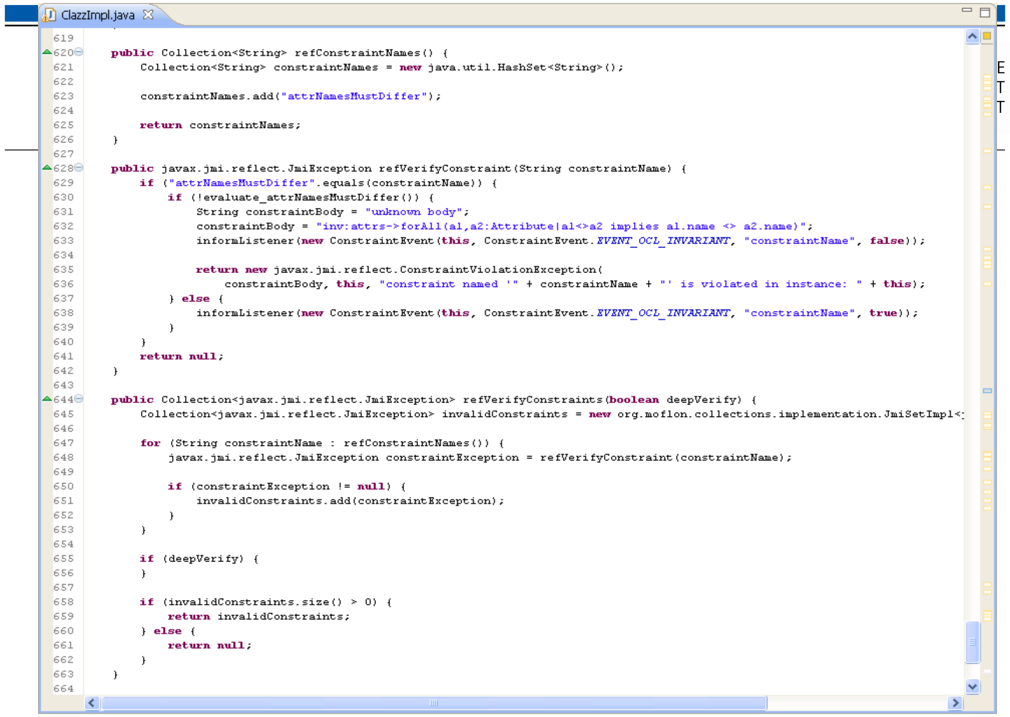
c1:Clazz

Model A

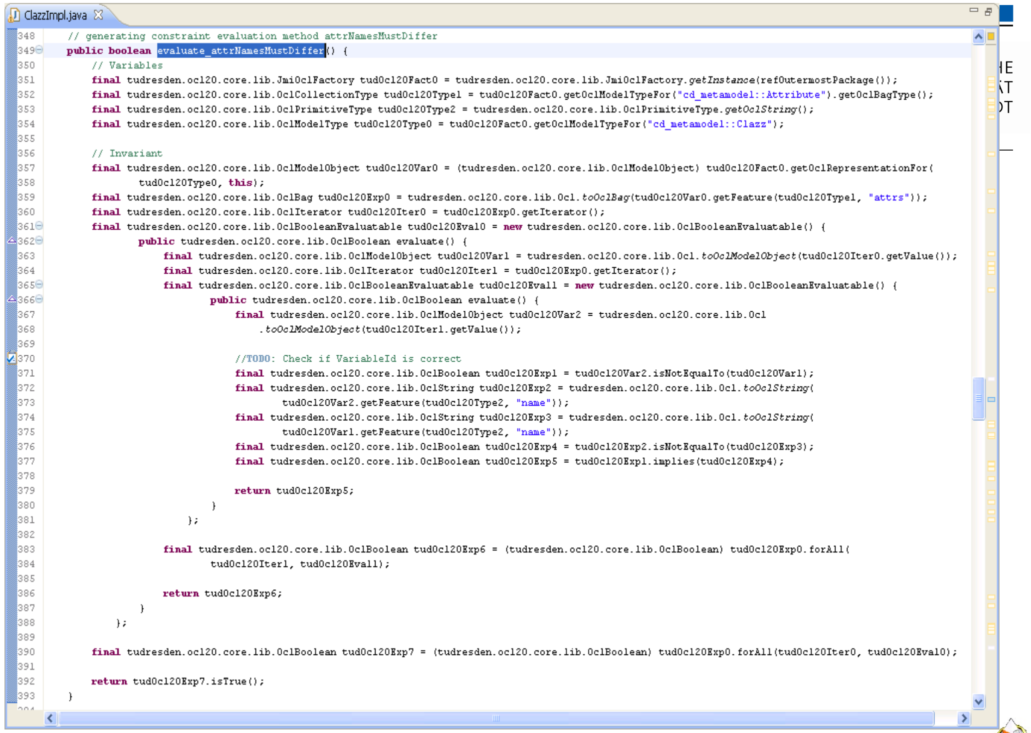
JMI compliant method

refVerifyConstraints(boolean deepVerify):Collection

<<invokes>>

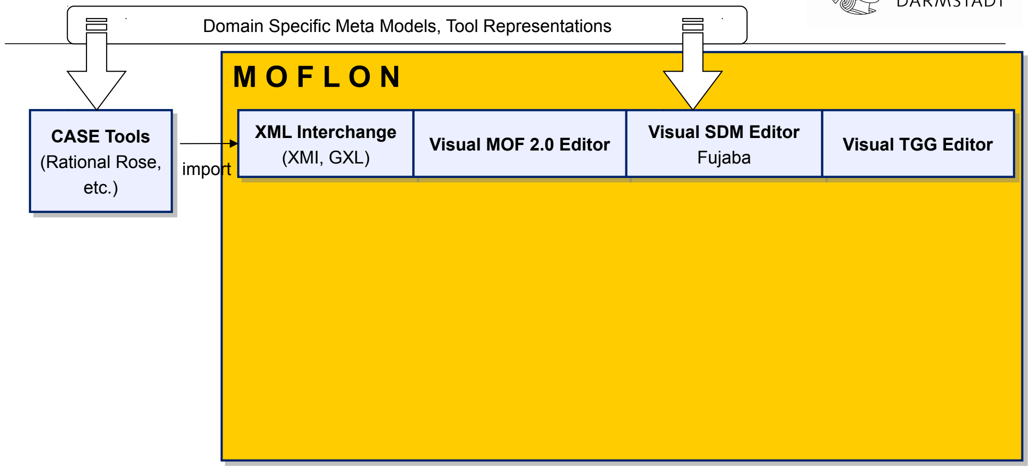


```java
public Collection<String> refConstraintNames() {
    Collection<String> constraintNames = new java.util.HashSet<String>();

    constraintNames.add("attrNamesMustDiffer");

    return constraintNames;
}

public javax.jmi.reflect.JmiException refVerifyConstraint(String constraintName) {
    if ("attrNamesMustDiffer".equals(constraintName)) {
        if (!evaluate_attrNamesMustDiffer()) {
            String constraintBody = "unknown body";
            constraintBody = "inv:attrs->forAll(a1,a2:Attribute|a1<>a2 implies a1.name <> a2.name)";
            informListener(new ConstraintEvent(this, ConstraintEvent.EVENT_OCL_INVARIANT, "constraintName", false));

            return new javax.jmi.reflect.ConstraintViolationException(
                constraintBody, this, "constraint named '" + constraintName + "' is violated in instance: " + this);
        } else {
            informListener(new ConstraintEvent(this, ConstraintEvent.EVENT_OCL_INVARIANT, "constraintName", true));
        }
    }
    return null;
}

public Collection<javax.jmi.reflect.JmiException> refVerifyConstraints(boolean deepVerify) {
    Collection<javax.jmi.reflect.JmiException> invalidConstraints = new org.moflon.collections.implementation.JmiSetImpl<j

    for (String constraintName : refConstraintNames()) {
        javax.jmi.reflect.JmiException constraintException = refVerifyConstraint(constraintName);

        if (constraintException != null) {
            invalidConstraints.add(constraintException);
        }
    }

    if (deepVerify) {
    }

    if (invalidConstraints.size() > 0) {
        return invalidConstraints;
    } else {
        return null;
    }
}
```



```java
// generating constraint evaluation method attrNamesMustDiffer
public boolean evaluate_attrNamesMustDiffer() {
    // Variables
    final tudresden.ocl20.core.lib.JmiOclFactory tud0cl20Fact0 = tudresden.ocl20.core.lib.JmiOclFactory.getInstance(refOutermostPackage());
    final tudresden.ocl20.core.lib.OclCollectionType tud0cl20Type1 = tud0cl20Fact0.getOclModelTypeFor("cd_metamodel::Attribute").getOclBagType();
    final tudresden.ocl20.core.lib.OclPrimitiveType tud0cl20Type2 = tudresden.ocl20.core.lib.OclPrimitiveType.getOclString();
    final tudresden.ocl20.core.lib.OclModelType tud0cl20Type0 = tud0cl20Fact0.getOclModelTypeFor("cd_metamodel::Clazz");

    // Invariant
    final tudresden.ocl20.core.lib.OclModelObject tud0cl20Var0 = (tudresden.ocl20.core.lib.OclModelObject) tud0cl20Fact0.getOclRepresentationFor(
            tud0cl20Type0, this);
    final tudresden.ocl20.core.lib.OclBag tud0cl20Exp0 = tudresden.ocl20.core.lib.Ocl.toOclBag(tud0cl20Var0.getFeature(tud0cl20Type1, "attrs"));
    final tudresden.ocl20.core.lib.OclIterator tud0cl20Iter0 = tud0cl20Exp0.getIterator();
    final tudresden.ocl20.core.lib.OclBooleanEvaluatable tud0cl20Eval0 = new tudresden.ocl20.core.lib.OclBooleanEvaluatable() {
            public tudresden.ocl20.core.lib.OclBoolean evaluate() {
                final tudresden.ocl20.core.lib.OclModelObject tud0cl20Var1 = tudresden.ocl20.core.lib.Ocl.toOclModelObject(tud0cl20Iter0.getValue());
                final tudresden.ocl20.core.lib.OclIterator tud0cl20Iter1 = tud0cl20Exp0.getIterator();
                final tudresden.ocl20.core.lib.OclBooleanEvaluatable tud0cl20Eval1 = new tudresden.ocl20.core.lib.OclBooleanEvaluatable() {
                        public tudresden.ocl20.core.lib.OclBoolean evaluate() {
                            final tudresden.ocl20.core.lib.OclModelObject tud0cl20Var2 = tudresden.ocl20.core.lib.Ocl
                                .toOclModelObject(tud0cl20Iter1.getValue());

                            //TODO: Check if VariableId is correct
                            final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp1 = tud0cl20Var2.isNotEqualTo(tud0cl20Var1);
                            final tudresden.ocl20.core.lib.OclString tud0cl20Exp2 = tudresden.ocl20.core.lib.Ocl.toOclString(
                                    tud0cl20Var2.getFeature(tud0cl20Type2, "name"));
                            final tudresden.ocl20.core.lib.OclString tud0cl20Exp3 = tudresden.ocl20.core.lib.Ocl.toOclString(
                                    tud0cl20Var1.getFeature(tud0cl20Type2, "name"));
                            final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp4 = tud0cl20Exp2.isNotEqualTo(tud0cl20Exp3);
                            final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp5 = tud0cl20Exp1.implies(tud0cl20Exp4);

                            return tud0cl20Exp5;
                        }
                    };

                final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp6 = (tudresden.ocl20.core.lib.OclBoolean) tud0cl20Exp0.forAll(
                        tud0cl20Iter1, tud0cl20Eval1);

                return tud0cl20Exp6;
            }
        };

    final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp7 = (tudresden.ocl20.core.lib.OclBoolean) tud0cl20Exp0.forAll(tud0cl20Iter0, tud0cl20Eval0);

    return tud0cl20Exp7.isTrue();
}
```

MOFLON – Architecture



Domain Specific Meta Models, Tool Representations

CASE Tools (Rational Rose, etc.)

import

MOFLON

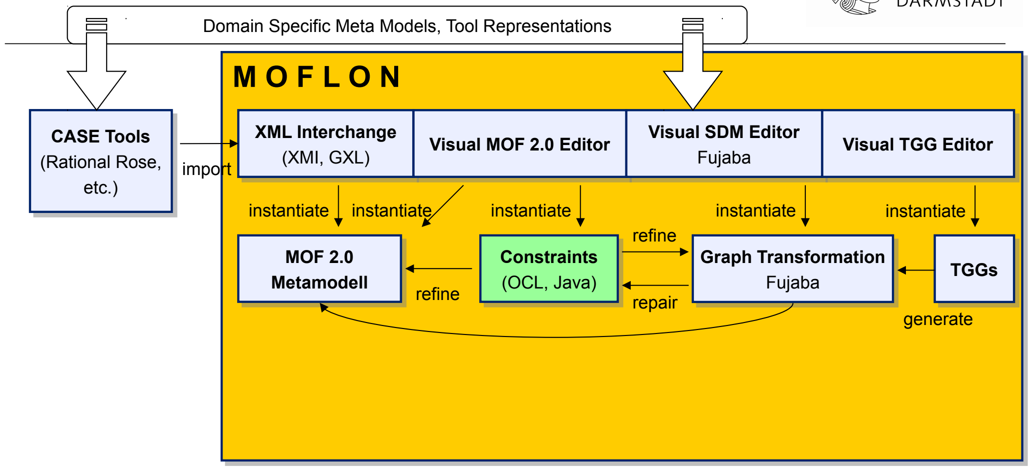
XML Interchange (XMI, GXL)

Visual MOF 2.0 Editor

Visual SDM Editor Fujaba

Visual TGG Editor

MOFLON – Architecture



Domain Specific Meta Models, Tool Representations

CASE Tools (Rational Rose, etc.)

import

MOFLON

XML Interchange (XMI, GXL)

Visual MOF 2.0 Editor

Visual SDM Editor Fujaba

Visual TGG Editor

instantiate

instantiate

instantiate

instantiate

instantiate

MOF 2.0 Metamodell

refine

Constraints (OCL, Java)

refine

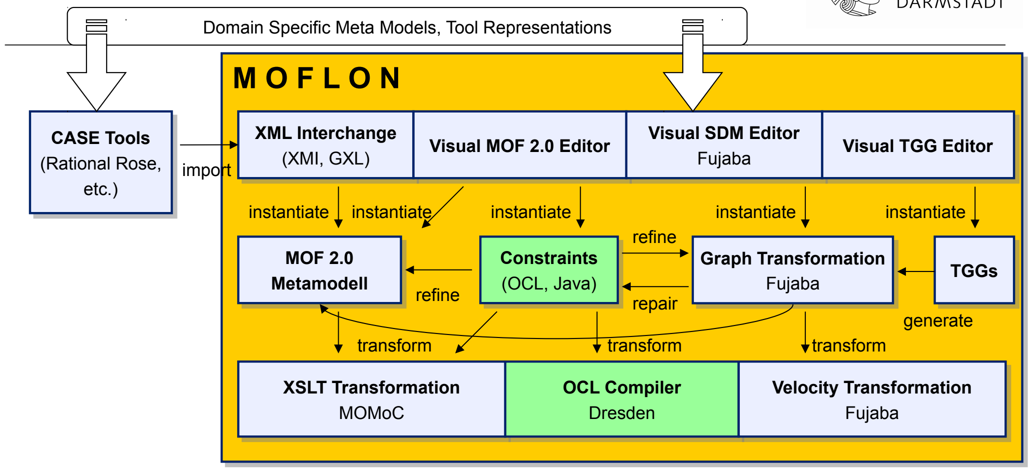
repair

Graph Transformation Fujaba

TGGs

generate

MOFLON – Architecture



Domain Specific Meta Models, Tool Representations

CASE Tools (Rational Rose, etc.)

import

MOFLON

XML Interchange (XMI, GXL)

Visual MOF 2.0 Editor

Visual SDM Editor Fujaba

Visual TGG Editor

instantiate

instantiate

instantiate

instantiate

instantiate

MOF 2.0 Metamodell

refine

Constraints (OCL, Java)

refine

repair

Graph Transformation Fujaba

TGGs

generate

transform

transform

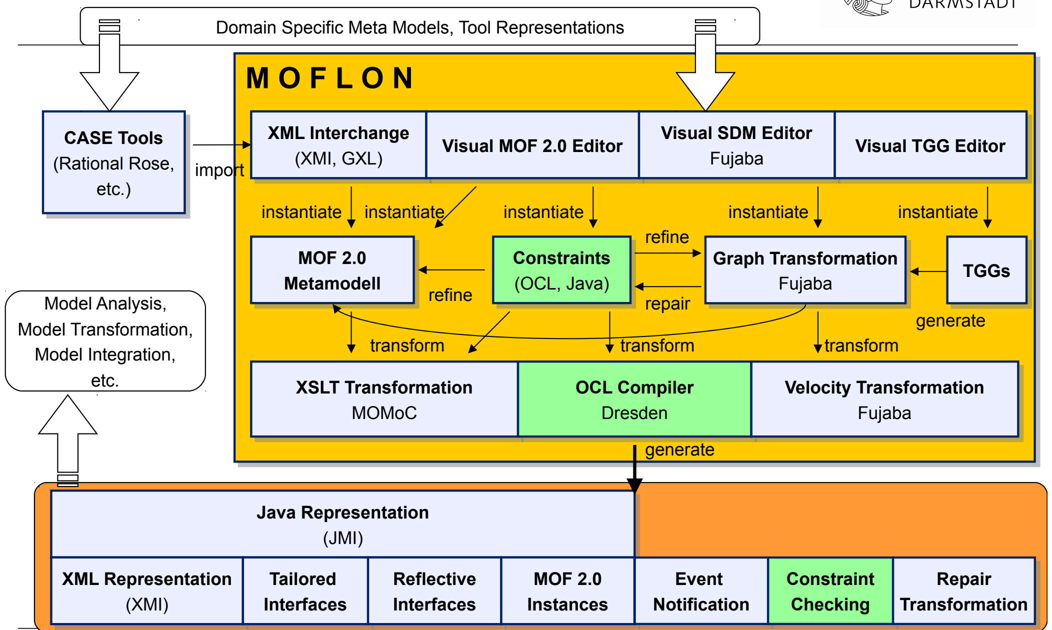
transform

XSLT Transformation MOMoC

OCL Compiler Dresden

Velocity Transformation Fujaba

MOFLON – Architecture



Domain Specific Meta Models, Tool Representations

CASE Tools (Rational Rose, etc.)

import

MOFLON

XML Interchange (XMI, GXL)

Visual MOF 2.0 Editor

Visual SDM Editor Fujaba

Visual TGG Editor

instantiate

instantiate

instantiate

instantiate

instantiate

MOF 2.0 Metamodell

refine

Constraints (OCL, Java)

refine

repair

Graph Transformation Fujaba

TGGs

generate

transform

transform

transform

XSLT Transformation MOMoC

OCL Compiler Dresden

Velocity Transformation Fujaba

generate

Model Analysis, Model Transformation, Model Integration, etc.

Java Representation (JMI)

XML Representation (XMI)

Tailored Interfaces

Reflective Interfaces

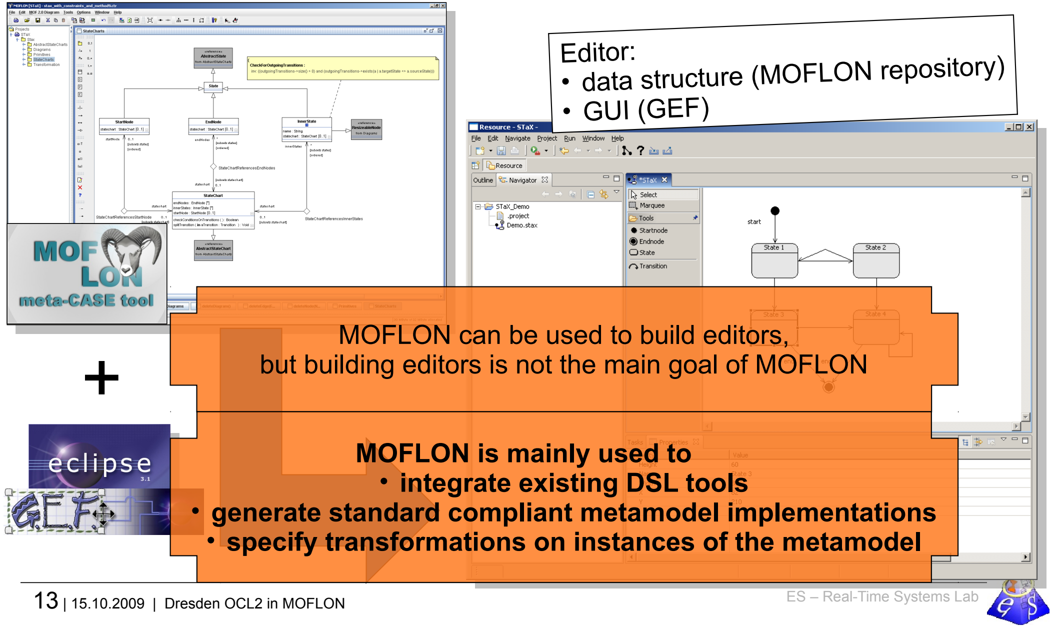
MOF 2.0 Instances

Event Notification

Constraint Checking

Repair Transformation

26.2 MOFLON Case Study – Statechart Editor (STaX)

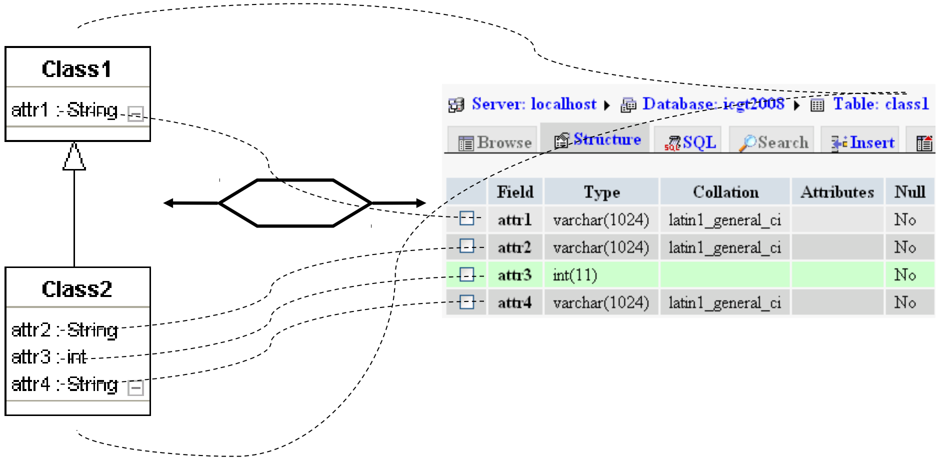


Editor:
- data structure (MOFLON repository)
- GUI (GEF)

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 with TGG – Class diagrams / database schemata



Server: localhost ▸ Database: icgt2008 ▸ Table: class1

Browse | Structure | SQL | Search | Insert

	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

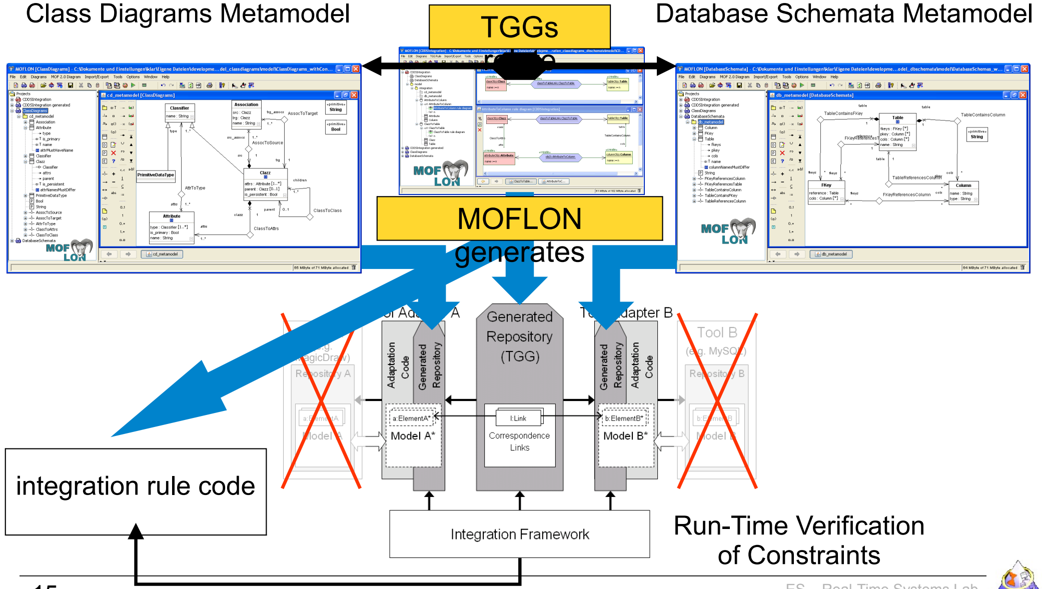
domain specific language, e.g. Class Diagrams

domain specific language, e.g. Database Schemata

Case Study 2: Tool Integration Scenario TiECDDS: (ClassD / DatabaseSchema)

Class Diagrams Metamodel

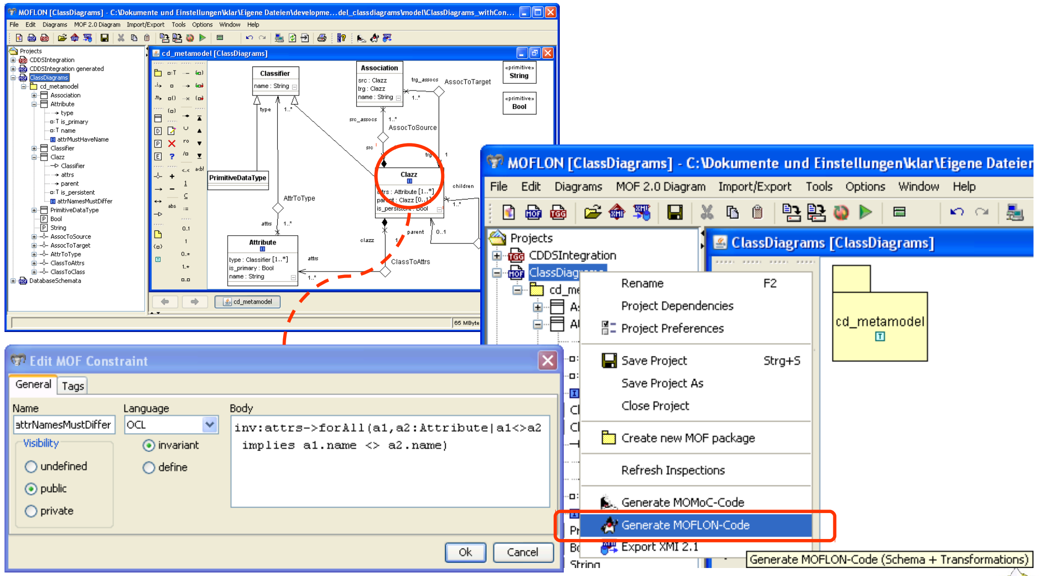
Database Schemata Metamodel



integration rule code

Run-Time Verification of Constraints

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



Edit MOF Constraint

General | Tags

Name: attrNamesMustDiffer

Language: OCL

Body:

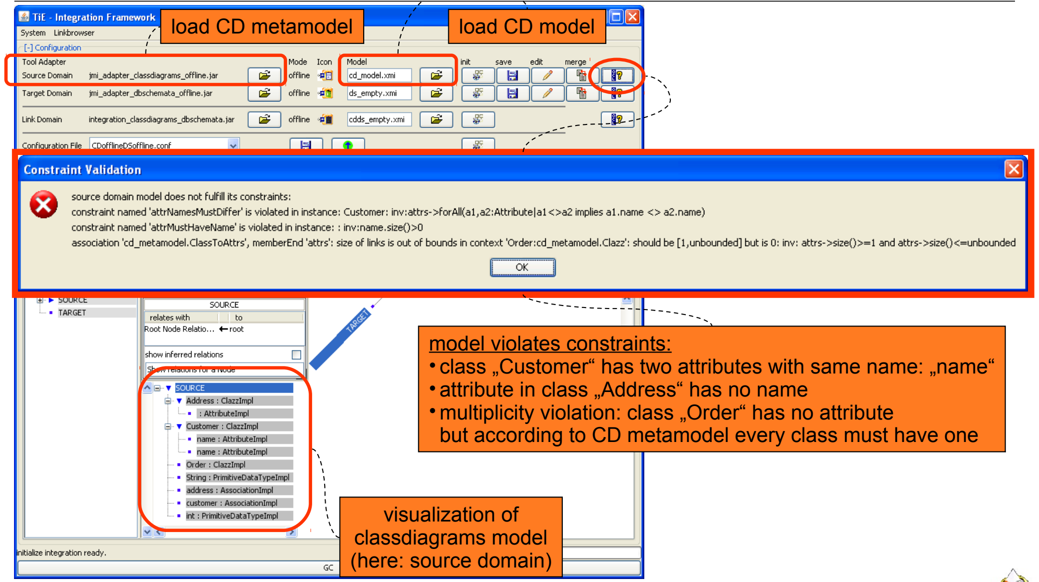
```
inv:attrs->forAll(a1,a2:Attribute|a1<>a2
 implies a1.name <> a2.name)
```

Visibility: ○ undefined ● public ○ private

● invariant ○ define

Ok | Cancel

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



load CD metamodel

load CD model

TiE - Integration Framework

Tool Adapter		Mode	Model
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

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

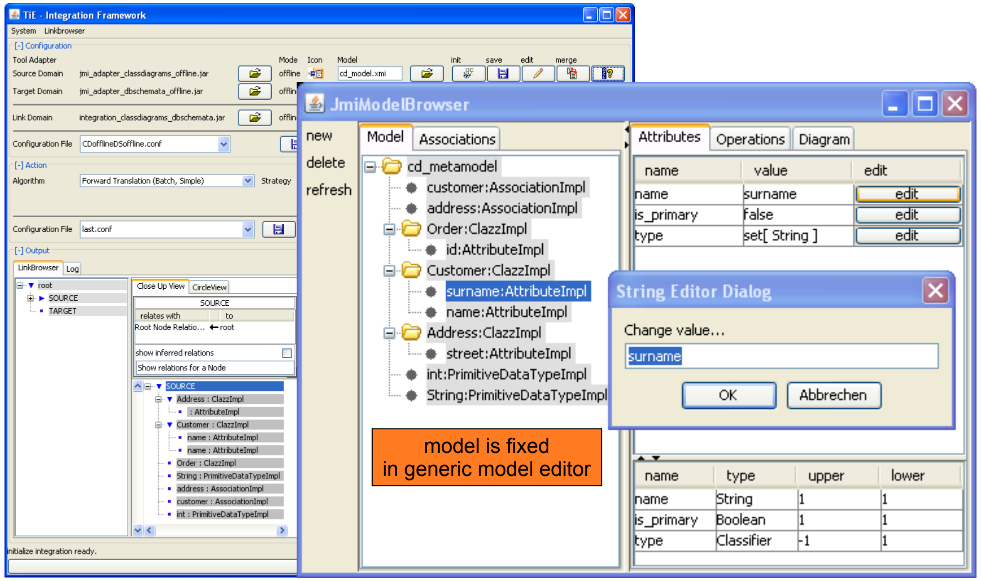
```
SOURCE
  Address : ClazzImpl
    : AttributeImpl
  Customer : ClazzImpl
    name : AttributeImpl
    name : AttributeImpl
  Order : ClazzImpl
  String : PrimitiveDataTypeImpl
  address : AssociationImpl
  customer : AssociationImpl
  int : PrimitiveDataTypeImpl
```

initialize integration ready.

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



JmiModelBrowser

```
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

String Editor Dialog

Change value…

surname

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

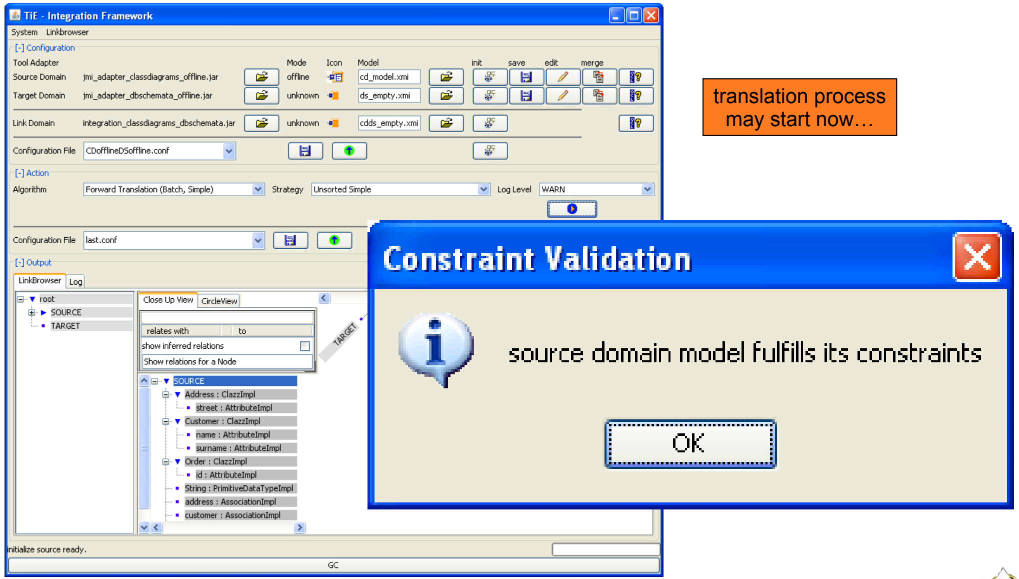
Algorithm: Forward Translation (Batch, Simple)

Configuration File: last.conf

initialize integration ready.

model is fixed in generic model editor

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



TiE - Integration Framework

Tool Adapter		Mode	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   Log Level: WARN

Configuration File: last.conf

Constraint Validation

source domain model fulfills its constraints

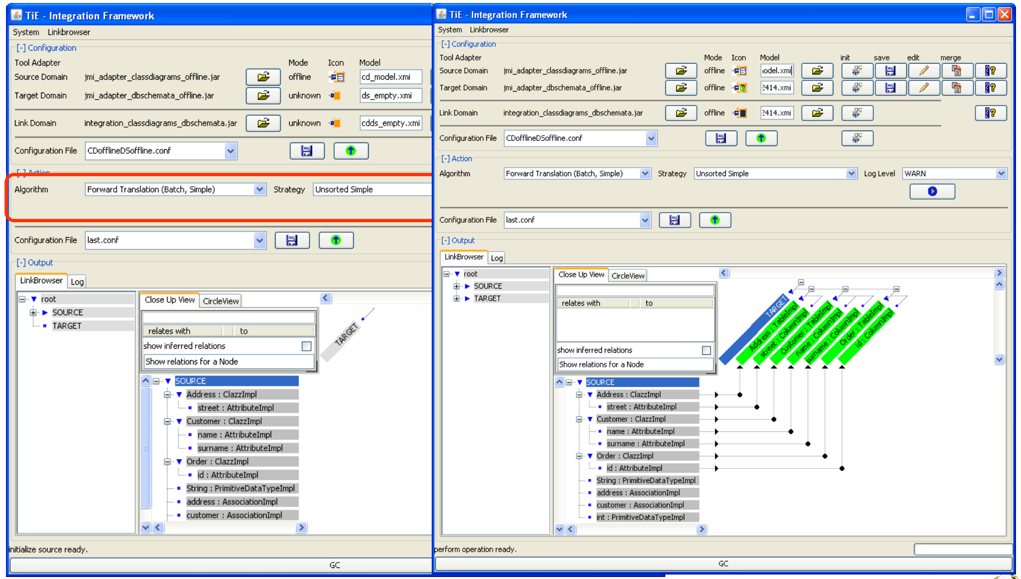
OK

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

initialize source ready.

translation process may start now…

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



TiE - Integration Framework

Tool Adapter		Mode	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

initialize source ready.

perform operation ready.

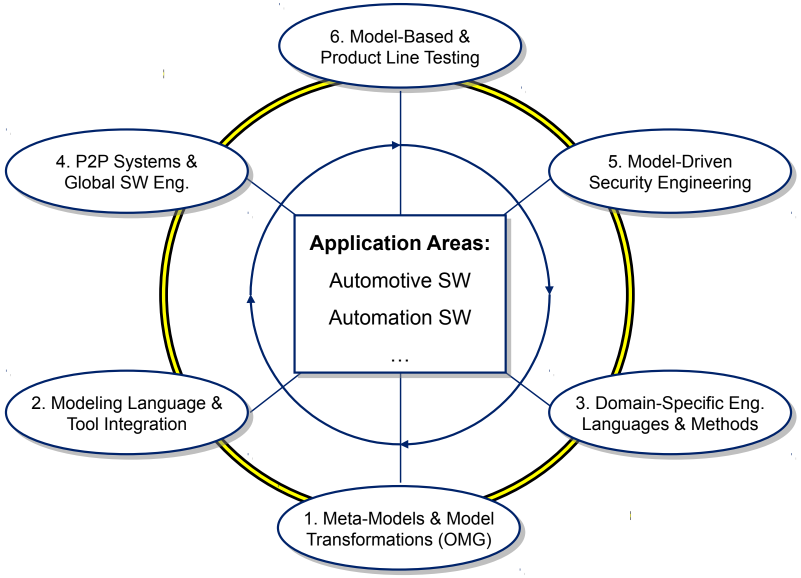
```
SOURCE
  Address : ClazzImpl
    street : AttributeImpl
  Customer : ClazzImpl
    name : AttributeImpl
    surname : AttributeImpl
  Order : ClazzImpl
    id : AttributeImpl
  String : PrimitiveDataTypeImpl
  address : AssociationImpl
  customer : AssociationImpl
  int : PrimitiveDataTypeImpl
```

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

Model-Driven Software Development at Real-Time Systems Lab (Prof. Schürr)


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)

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	AToM³	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

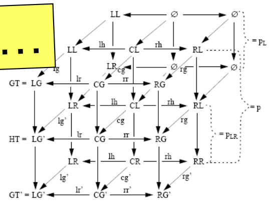
from Amelunxen, Königs, Rötschke, and Schürr,
„MOSL: Composing a Visual Language for a Metamodeling Framework“
in IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC 2006), September, 2006, 81-84

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…



MOFLON

http://www.moflon.org