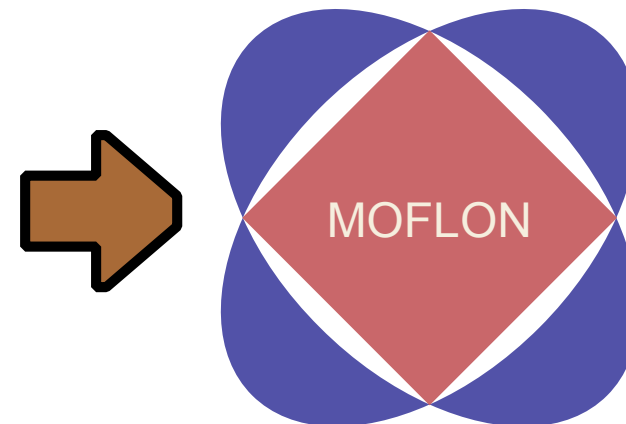


34. The Meta-CASE-Tool (e)MOFLON

A Meta-CASE tool and a 1-TS-Software Factory

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
<http://st.inf.tu-dresden.de>
Version 15-0.5, 16.01.16

- 1) MOFLON Meta-CASE-Werkzeug
- 2) Architecture
- 3) TGG



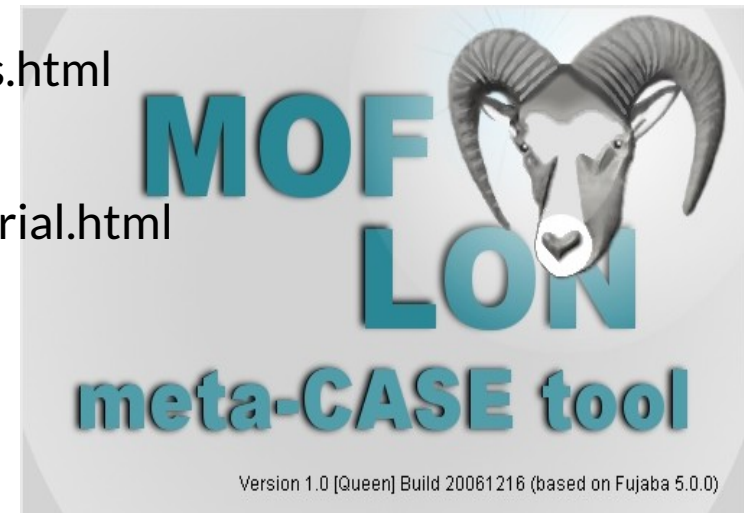
DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur

Reading

2

Model-Driven Software Development in Technical Spaces (MOST)

- ▶ MOFLON Website <http://www.moflon.org>
- ▶ The Eclipse-Version of the tool is called eMOFLON
 - eMOFLON tutorial
 - <http://www.moflon.org/fileadmin/download/moflon-ide/eclipse-plugin/documents/release/eMoflonTutorial.pdf>
- ▶ A Comparison of ATL and Story-Driven Modeling (Fujaba-style GRS)
 - http://www.es.tu-darmstadt.de/fileadmin/download/publications/spatzina/PP_AGTIVE_2011.pdf
- ▶ MOFLON Training
 - <http://moflon.org/documentation/links.html>
- ▶ MOFLON Tutorial
 - <http://moflon.org/documentation/tutorial.html>



34.1. eMOFLON Introduction

- ▶ MOFLON is a Metamodelling Toolset (Meta-CASE tool) of TU Darmstadt, Fachgruppe Real-Time Systems, Prof. Andy Schürr
 - MOFLON uses OCL (logic) for the checking of wellformedness of all models
 - MOFLON is an extension of Fujaba offering graph rewriting www.fujaba.de
 - MOFLON supports Triple Graph Grammars (TGG, see ST-II)
- ▶ eMOFLON supports the Technical Space of E(MOF)
 - OCL 2.0
 - JMI 1.4
 - XMI 2.1
- ▶ EMOFLON relies on metamodel composition of MOF, OCL and metamodel mappings between MOF, XML and Java

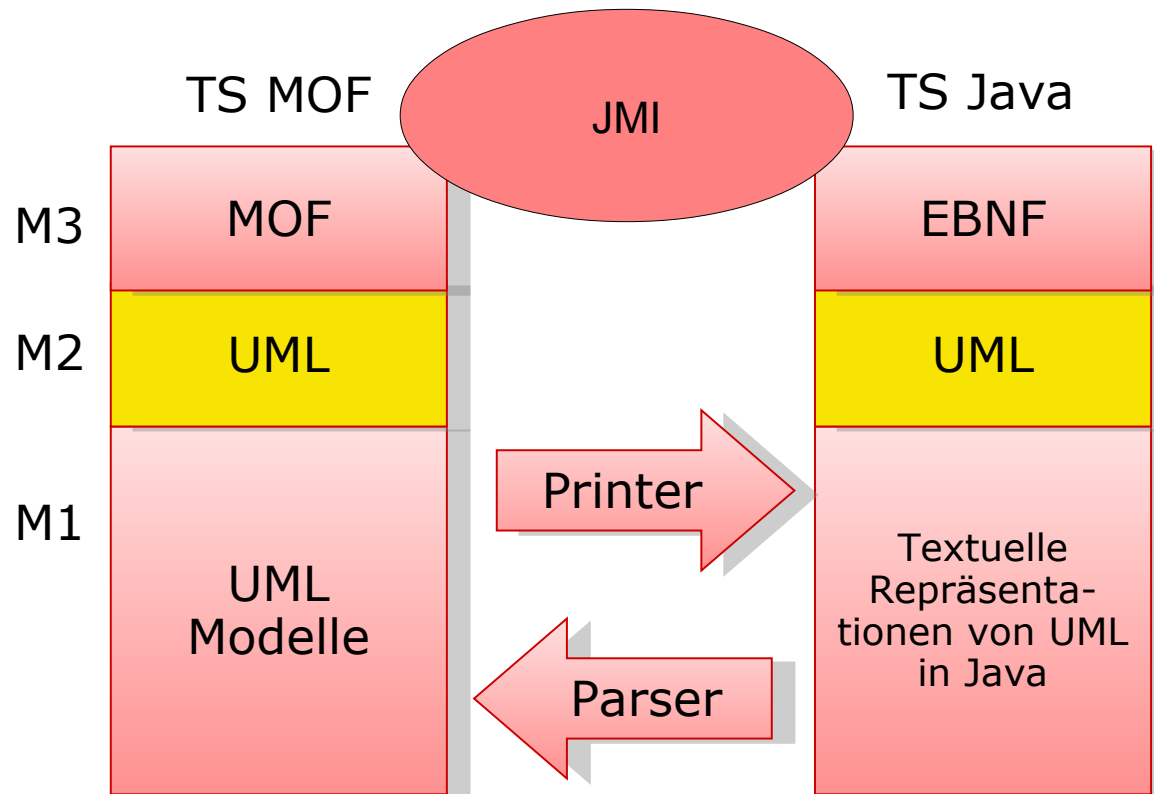


Code Generation with JMI, transformative TS-Bridge for (E)MOF and Java for the Language UML

4

Model-Driven Software Development in Technical Spaces (MOST)

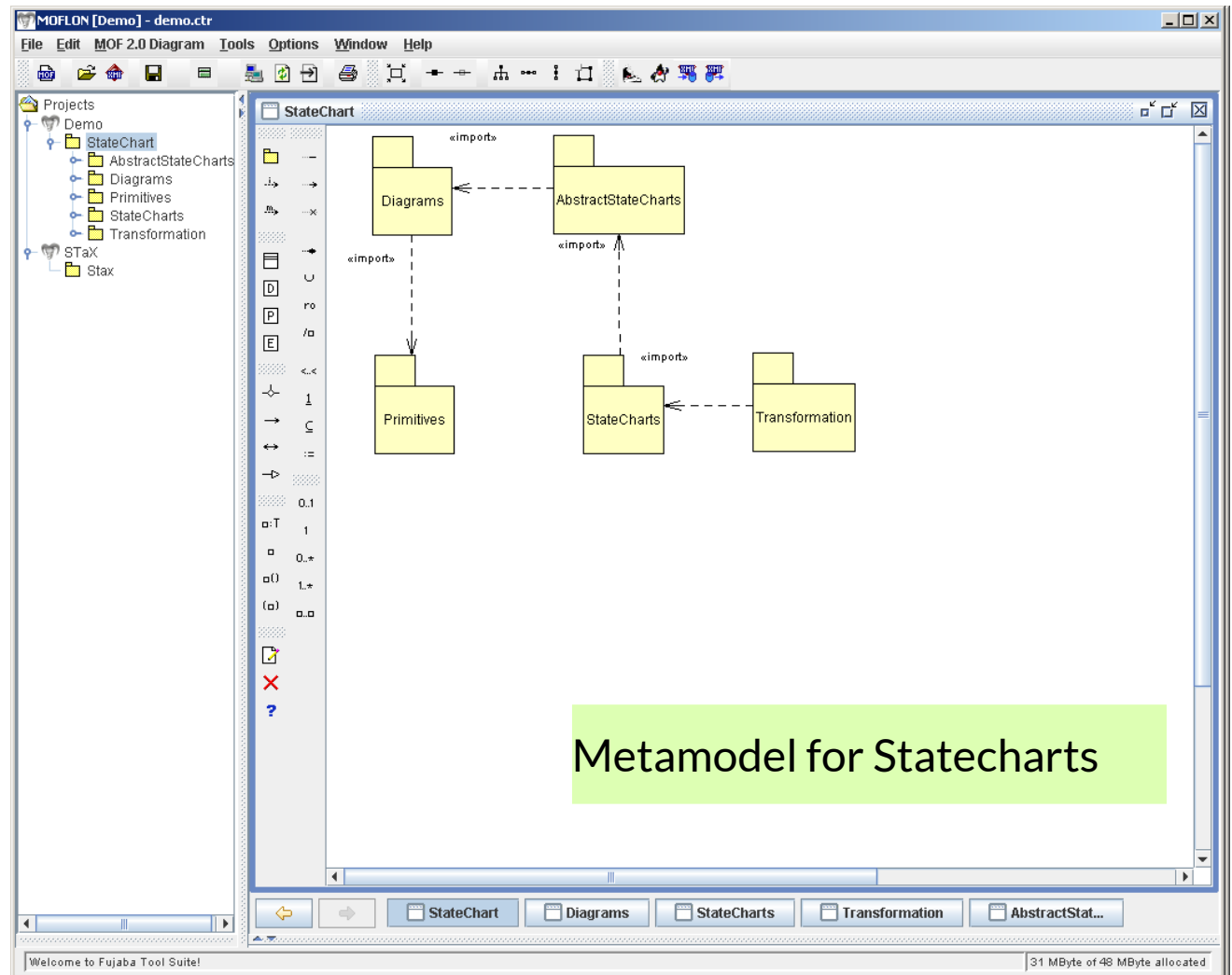
- ▶ Java Metadata Interchange (JMI) is similar to XMI, a TS bridge between (E)MoF and Grammarware
- ▶ Only for UML available (language mapping)



MOFLON Example 1: Metamodel for Statecharts: Development Process

5 Model-Driven Software Development in Technical Spaces (MOST)

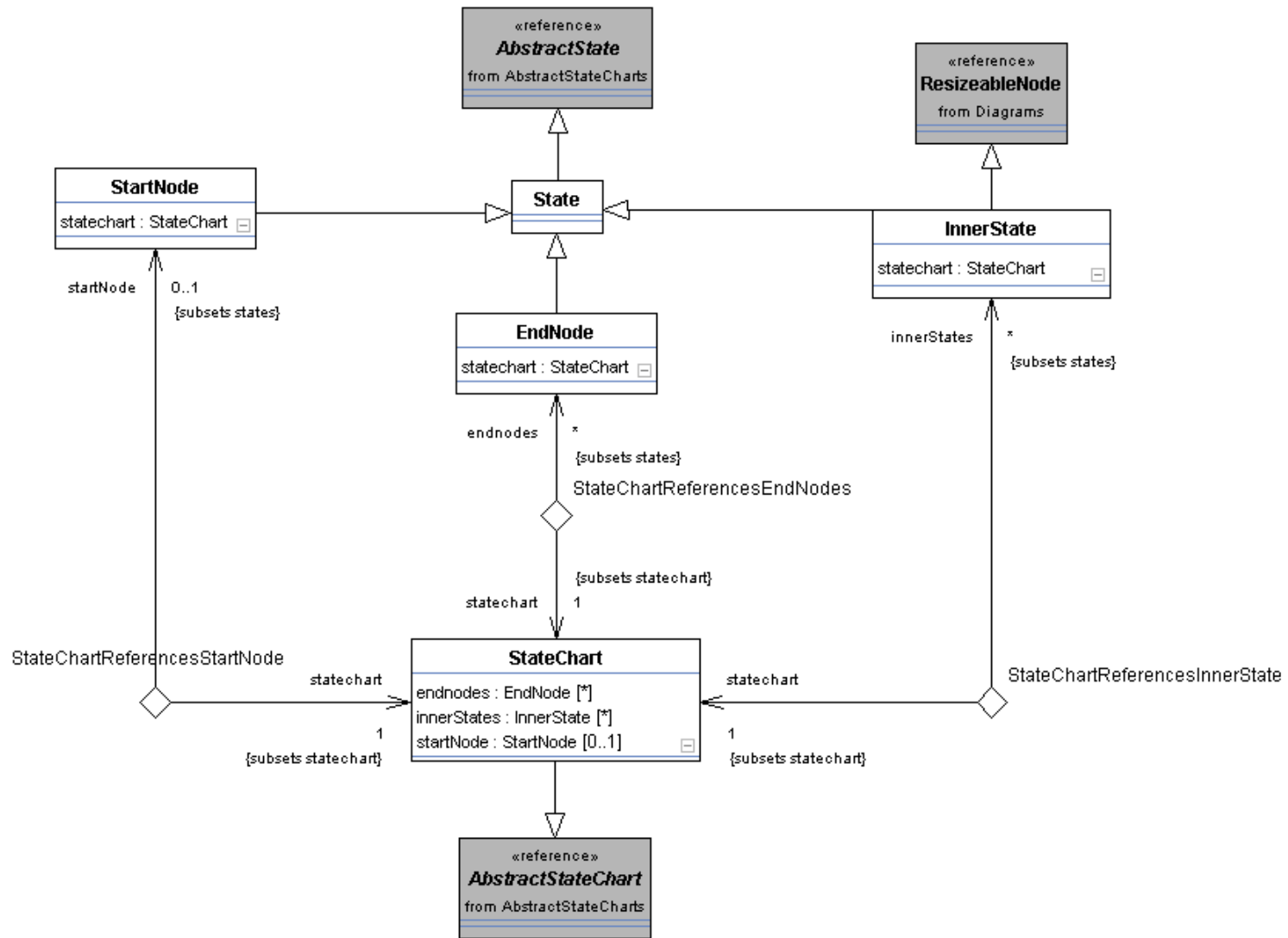
- 1) Create metamodel
- 2) Generate Code (Generate repository with constraint-checker) with the JMI interfaces



Example: 1.a) Metamodel for Statecharts

6

Model-Driven Software Development in Technical Spaces (MOST)

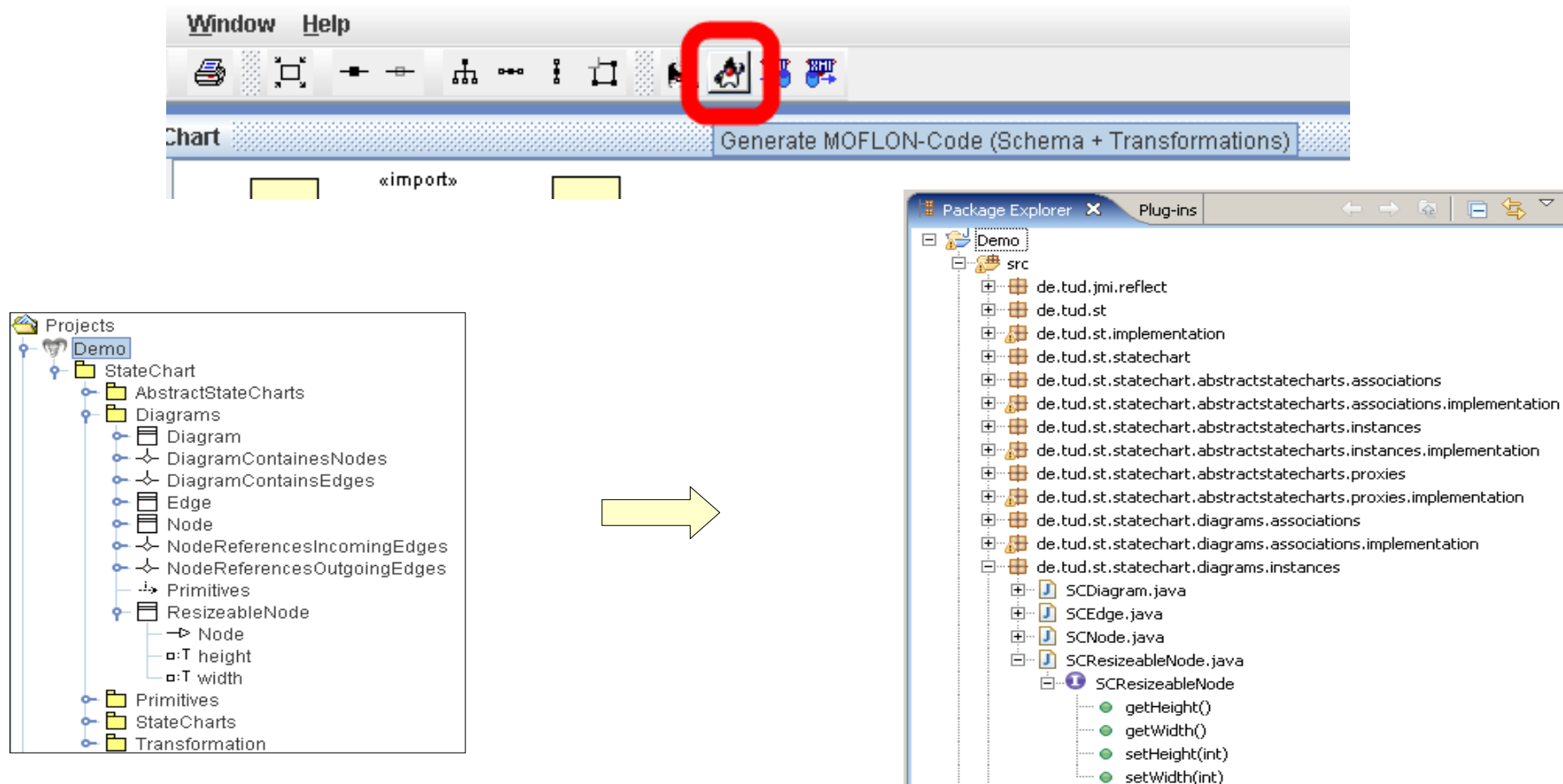


Example: 1.b) Code Generation from Statechart-Metamodel

7

Model-Driven Software Development in Technical Spaces (MOST)

- ▶ Uses JMI interfaces for the repository (metamodel-driven repository)
 - Codegenerator uses String template engine Velocity and XSLT-1.1 XML transformation
- ▶ Generates code for all Methods modeled as Story-diagrams (from Fujaba)






Example: 1.b) Codegeneration from Metamodel for Statecharts





8

Model-Driven Software Development in Technical Spaces (MOST)

Per (E)MOF Package

- Java Package  de.tud.st.statechart
- Interface  SCStateChartPackage.java
- Implementation  SCStateChartPackageImpl.java

Per Metaclass

- Interface  SCNode.java
- Implementation  SCNodeImpl.java
- Proxy Interface  SCNodeClass.java
- Proxy Implementation  SCNodeClassImpl.java

Per Association

- Interface  SCDiagramContainsEdges.java
- Implementation  SCDiagramContainsEdgesImpl.java

Example: 1.c) How to Use Statechart Models in the Generated Repository

- ▶ Initialize root package

```
SCStateChartPackage root = new SCStateChartPackageImpl ();
```

- ▶ Find Proxy of repository

```
root.getSCDiagramsPackage ().getSCNode ();
```

- ▶ Generate nodes (model elements) via Proxy. All interfaces are typed by metaclasses

```
SCNode node = root.getSCDiagramsPackage ().getSCNode ().createSCNode ();
```

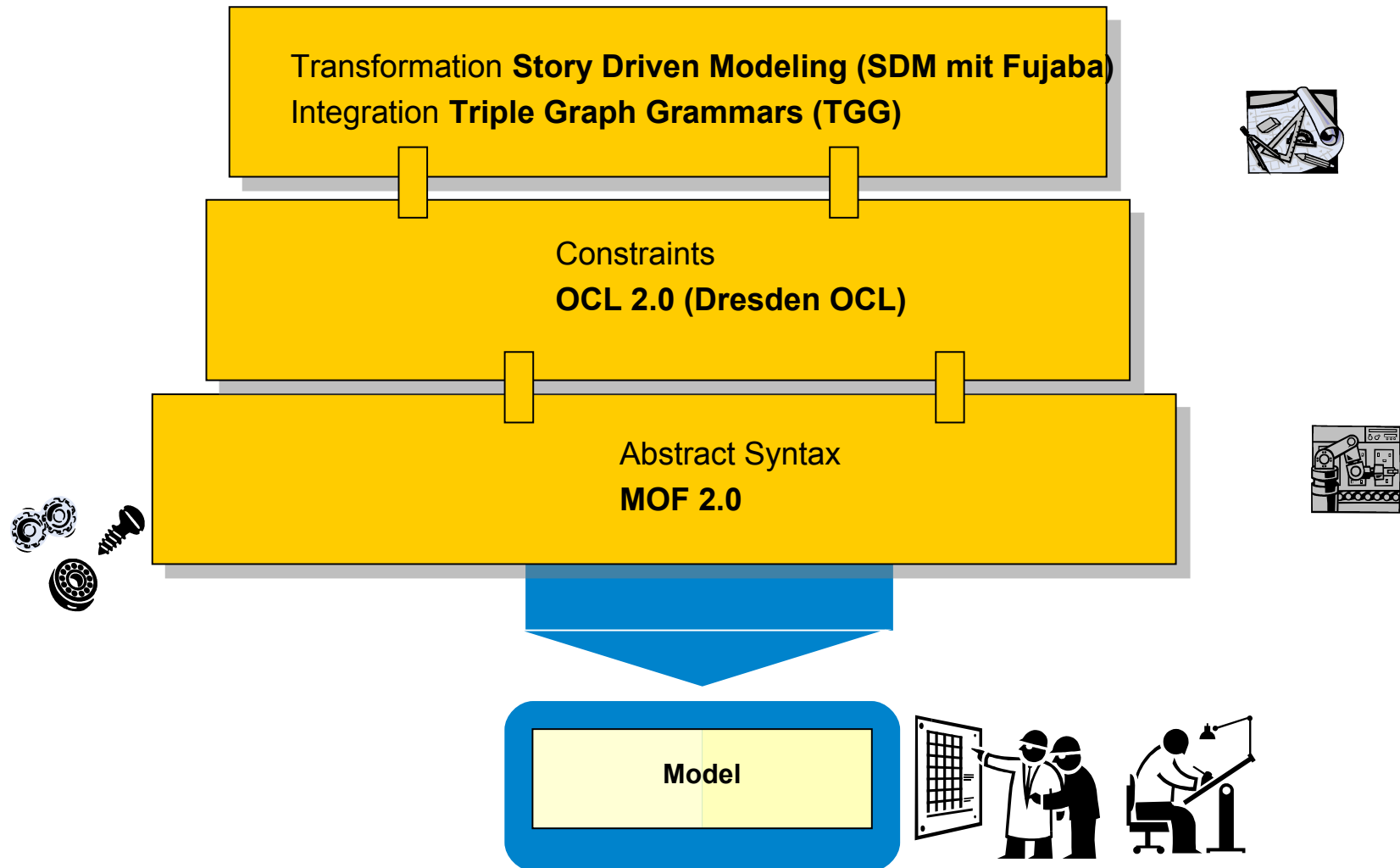
34.2. The Metamodeling Architecture of MetaCASE Tool MOFLON

Slides from: 10 Jahre Dresden-OCL – Workshop
<http://dresden-ocl.sourceforge.net/>
<http://dresden-ocl.sourceforge.net/10years.html>
used by permission



DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur

Metamodel Architecture of MOFLON



MOFLON MetaCASE – Main Features

12

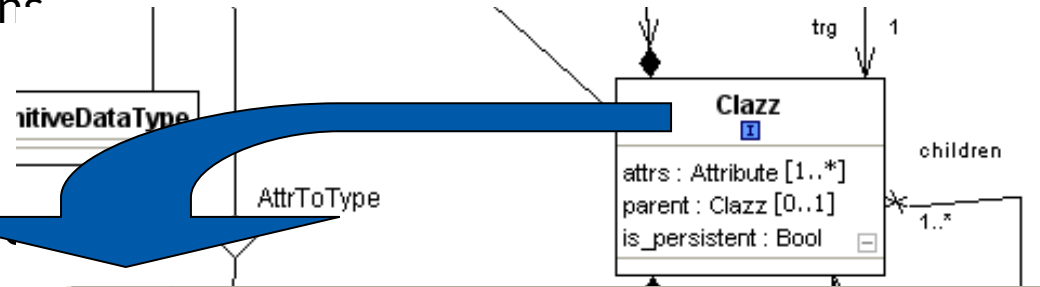
Model-Driven Software Development in Technical Spaces (MOST)

- ▶ 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
- ▶ interoperability (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



(OCL) Constraints in MOFLON – Generated Implementations

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



```

619 public Collection<String> refConstraintNames() {
620     Collection<String> constraintNames = new java.util.HashSet<String>();
621     constraintNames.add("attrNamesMustDiffer");
622     return constraintNames;
623 }
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
    
```

MOFLON-code
 refVerifyConstraint(String name):JmiException

JMI compliant method
 refVerifyConstraints(boolean deepVerify):Collection

<<calls>>

```

949 // generating constraint evaluation method attrNamesMustDiffer
950 public boolean evaluate_attrNamesMustDiffer() {
951     // Variables
952     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp0 = tudresden.oc120.core.lib.OclBoolean.valueOf(false);
953     final tudresden.oc120.core.lib.OclCollectionType tud0c120Type1 = tudresden.oc120.core.lib.OclCollectionType.getOclModelTypeFor("cd_metamodel::Attribute").getOclBagType();
954     final tudresden.oc120.core.lib.OclPrimitiveType tud0c120Type2 = tudresden.oc120.core.lib.OclPrimitiveType.getOclStringType();
955     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp1 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
956     // Trg
957     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp2 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
958     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp3 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
959     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp4 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
960     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp5 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
961     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp6 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
962     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp7 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
963     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp8 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
964     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp9 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
965     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp10 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
966     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp11 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
967     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp12 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
968     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp13 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
969     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp14 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
970     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp15 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
971     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp16 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
972     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp17 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
973     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp18 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
974     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp19 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
975     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp20 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
976     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp21 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
977     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp22 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
978     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp23 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
979     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp24 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
980     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp25 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
981     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp26 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
982     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp27 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
983     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp28 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
984     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp29 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
985     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp30 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
986     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp31 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
987     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp32 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
988     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp33 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
989     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp34 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
990     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp35 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
991     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp36 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
992     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp37 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
993     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp38 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
994     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp39 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
995     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp40 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
996     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp41 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
997     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp42 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
998     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp43 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
999     final tudresden.oc120.core.lib.OclBoolean tud0c120Exp44 = tudresden.oc120.core.lib.OclBoolean.valueOf(true);
1000    }
    
```

Dresden OCL-code

<<queries>>

<<invokes>>

generated Repository
 c1:Clazz
 Model A



```
ClazzImpl.java X
619
620 public Collection<String> refConstraintNames() {
621     Collection<String> constraintNames = new java.util.HashSet<String>();
622
623     constraintNames.add("attrNamesMustDiffer");
624
625     return constraintNames;
626 }
627
628 public javax.jmi.reflect.JmiException refVerifyConstraint(String constraintName) {
629     if ("attrNamesMustDiffer".equals(constraintName)) {
630         if (!evaluate_attrNamesMustDiffer()) {
631             String constraintBody = "unknown body";
632             constraintBody = "inv:attrs->forall(a1,a2:Attribute|a1<>a2 implies a1.name <> a2.name)";
633             informListener(new ConstraintEvent(this, ConstraintEvent.EVENT_OCL_INVARIANT, "constraintName", false));
634
635             return new javax.jmi.reflect.ConstraintViolationException(
636                 constraintBody, this, "constraint named '" + constraintName + "' is violated in instance: " + this);
637         } else {
638             informListener(new ConstraintEvent(this, ConstraintEvent.EVENT_OCL_INVARIANT, "constraintName", true));
639         }
640     }
641     return null;
642 }
643
644 public Collection<javax.jmi.reflect.JmiException> refVerifyConstraints(boolean deepVerify) {
645     Collection<javax.jmi.reflect.JmiException> invalidConstraints = new org.moflon.collections.implementation.JmiSetImpl<
646
647     for (String constraintName : refConstraintNames()) {
648         javax.jmi.reflect.JmiException constraintException = refVerifyConstraint(constraintName);
649
650         if (constraintException != null) {
651             invalidConstraints.add(constraintException);
652         }
653     }
654
655     if (deepVerify) {
656     }
657
658     if (invalidConstraints.size() > 0) {
659         return invalidConstraints;
660     } else {
661         return null;
662     }
663 }
664
```

JMI compliant
method

```
ClazzImpl.java X
348 // generating constraint evaluation method attrNamesMustDiffer
349 public boolean evaluate_attrNamesMustDiffer() {
350     // Variables
351     final tudresden.oc120.core.lib.JmiOclFactory tudOcl20Fact0 = tudresden.oc120.core.lib.JmiOclFactory.getInstance(refOutermostPackage());
352     final tudresden.oc120.core.lib.OclCollectionType tudOcl20Type1 = tudOcl20Fact0.getOclModelTypeFor("cd_metamodel::Attribute").getOclBagType();
353     final tudresden.oc120.core.lib.OclPrimitiveType tudOcl20Type2 = tudresden.oc120.core.lib.OclPrimitiveType.getOclString();
354     final tudresden.oc120.core.lib.OclModelType tudOcl20Type0 = tudOcl20Fact0.getOclModelTypeFor("cd_metamodel::Clazz");
355
356     // Invariant
357     final tudresden.oc120.core.lib.OclModelObject tudOcl20Var0 = (tudresden.oc120.core.lib.OclModelObject) tudOcl20Fact0.getOclRepresentationFor(
358         tudOcl20Type0, this);
359     final tudresden.oc120.core.lib.OclBag tudOcl20Exp0 = tudresden.oc120.core.lib.Ocl.toOclBag(tudOcl20Var0.getFeature(tudOcl20Type1, "attrs"));
360     final tudresden.oc120.core.lib.OclIterator tudOcl20Iter0 = tudOcl20Exp0.getIterator();
361     final tudresden.oc120.core.lib.OclBooleanEvaluatable tudOcl20Eval0 = new tudresden.oc120.core.lib.OclBooleanEvaluatable() {
362         public tudresden.oc120.core.lib.OclBoolean evaluate() {
363             final tudresden.oc120.core.lib.OclModelObject tudOcl20Var1 = tudresden.oc120.core.lib.Ocl.toOclModelObject(tudOcl20Iter0.getValue());
364             final tudresden.oc120.core.lib.OclIterator tudOcl20Iter1 = tudOcl20Exp0.getIterator();
365             final tudresden.oc120.core.lib.OclBooleanEvaluatable tudOcl20Eval1 = new tudresden.oc120.core.lib.OclBooleanEvaluatable() {
366                 public tudresden.oc120.core.lib.OclBoolean evaluate() {
367                     final tudresden.oc120.core.lib.OclModelObject tudOcl20Var2 = tudresden.oc120.core.lib.Ocl
368                         .toOclModelObject(tudOcl20Iter1.getValue());
369
370                     //TODO: Check if VariableId is correct
371                     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp1 = tudOcl20Var2.isNotEqualTo(tudOcl20Var1);
372                     final tudresden.oc120.core.lib.OclString tudOcl20Exp2 = tudresden.oc120.core.lib.Ocl.toOclString(
373                         tudOcl20Var2.getFeature(tudOcl20Type2, "name"));
374                     final tudresden.oc120.core.lib.OclString tudOcl20Exp3 = tudresden.oc120.core.lib.Ocl.toOclString(
375                         tudOcl20Var1.getFeature(tudOcl20Type2, "name"));
376                     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp4 = tudOcl20Exp2.isNotEqualTo(tudOcl20Exp3);
377                     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp5 = tudOcl20Exp1.implies(tudOcl20Exp4);
378
379                     return tudOcl20Exp5;
380                 }
381             };
382
383             final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp6 = (tudresden.oc120.core.lib.OclBoolean) tudOcl20Exp0.forAll(
384                 tudOcl20Iter1, tudOcl20Eval1);
385
386             return tudOcl20Exp6;
387         }
388     };
389
390     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp7 = (tudresden.oc120.core.lib.OclBoolean) tudOcl20Exp0.forAll(tudOcl20Iter0, tudOcl20Eval0);
391
392     return tudOcl20Exp7.isTrue();
393 }
394
```

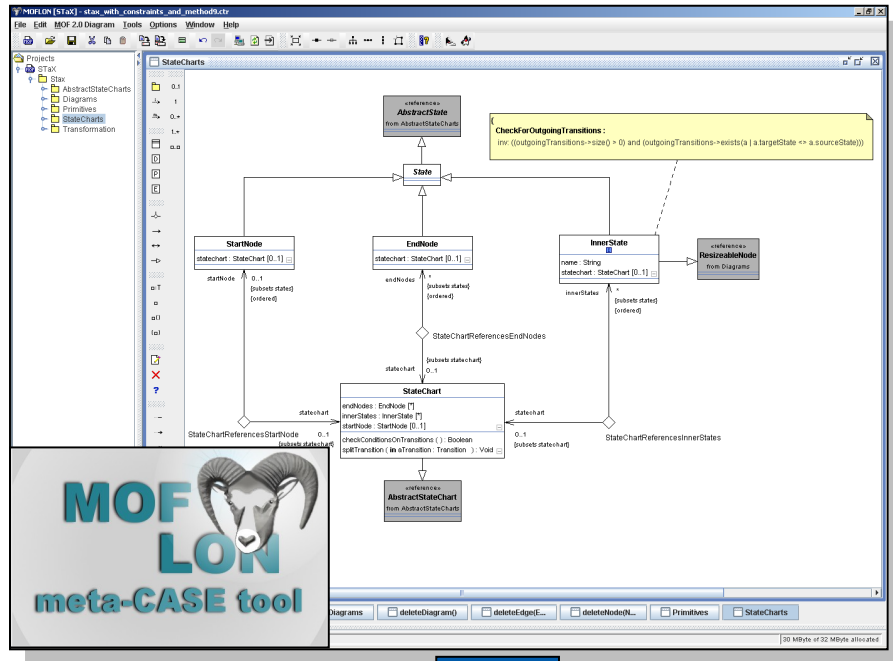
Generated
Code
from
Dresden OCL



Result of MOFLON Example 1 – Statechart Editor (STaX)

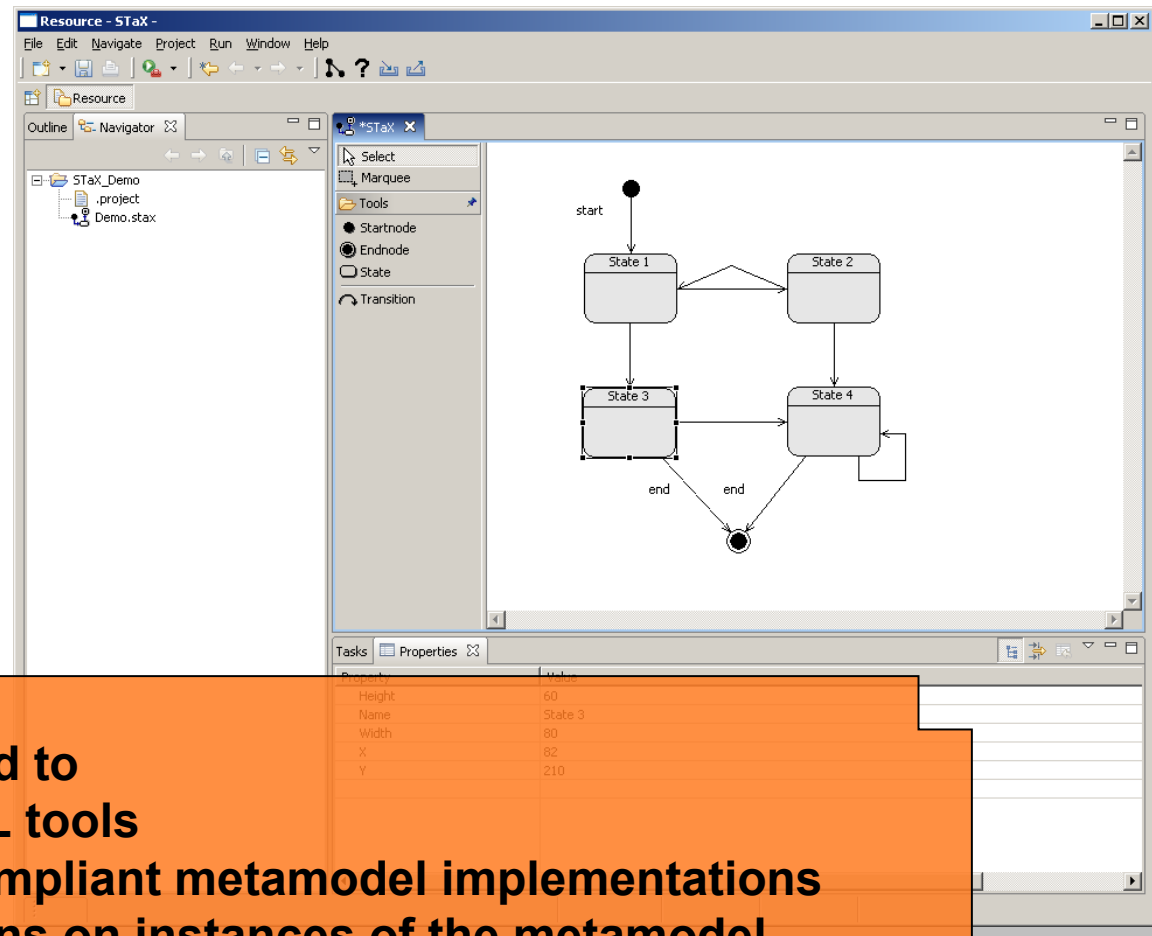
17

Model-Driven Software Development in Technical Spaces (MOST)



Editor:

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



+

MOFLON is mainly used to

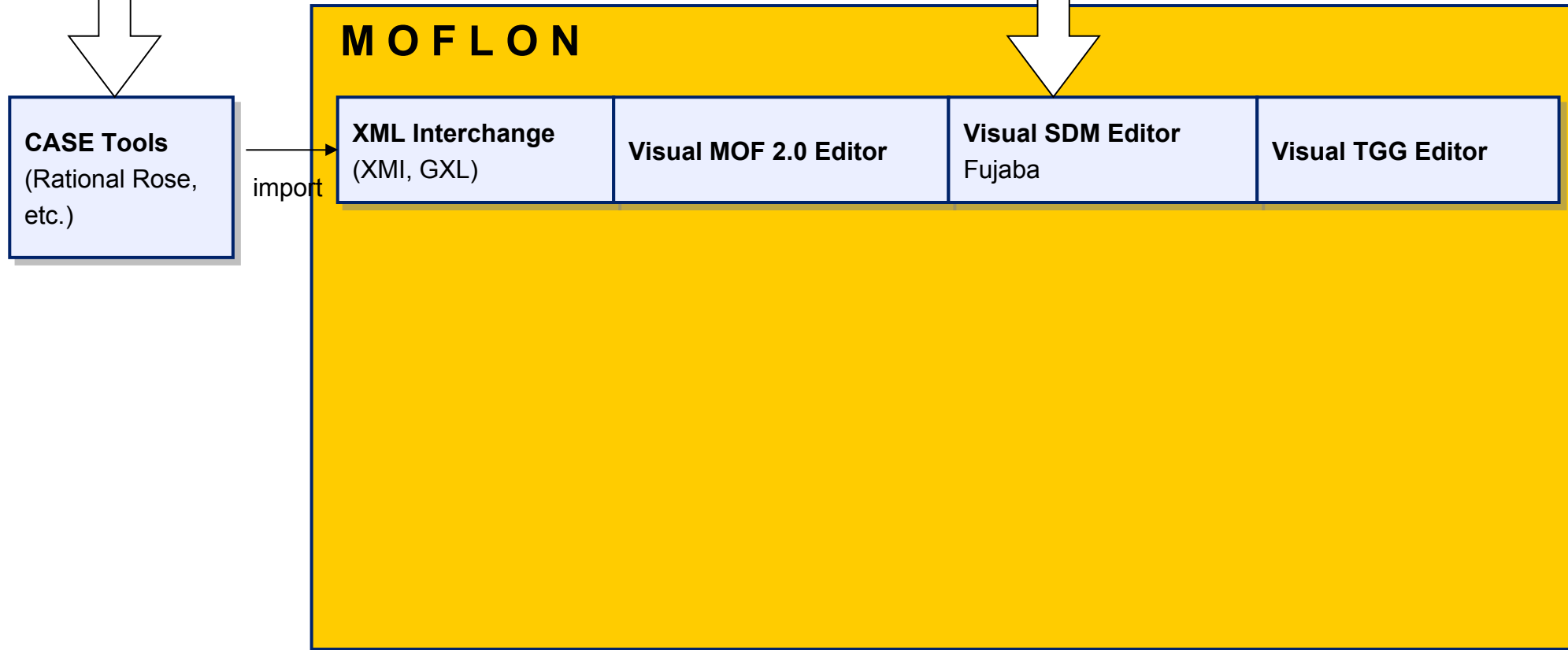
- integrate existing DSL tools
- generate standard compliant metamodel implementations
- specify transformations on instances of the metamodel

34.3.3 MOFLON – Architecture

18

Model-Driven Software Development in Technical Spaces (MOST)

Domain Specific Meta Models, Tool Representations



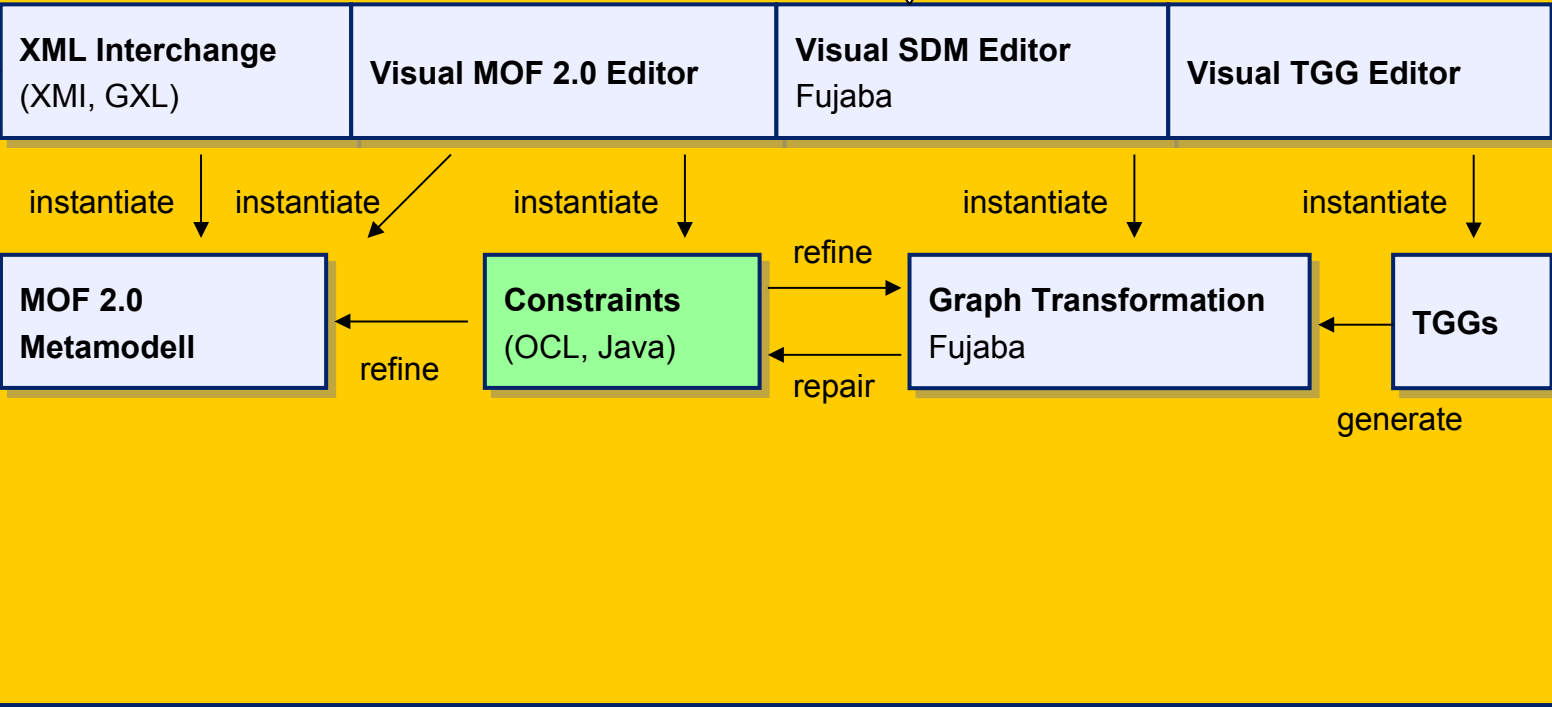
MOFLON - Architecture

Domain Specific Meta Models, Tool Representations

CASE Tools
(Rational Rose, etc.)

import

MOFLON

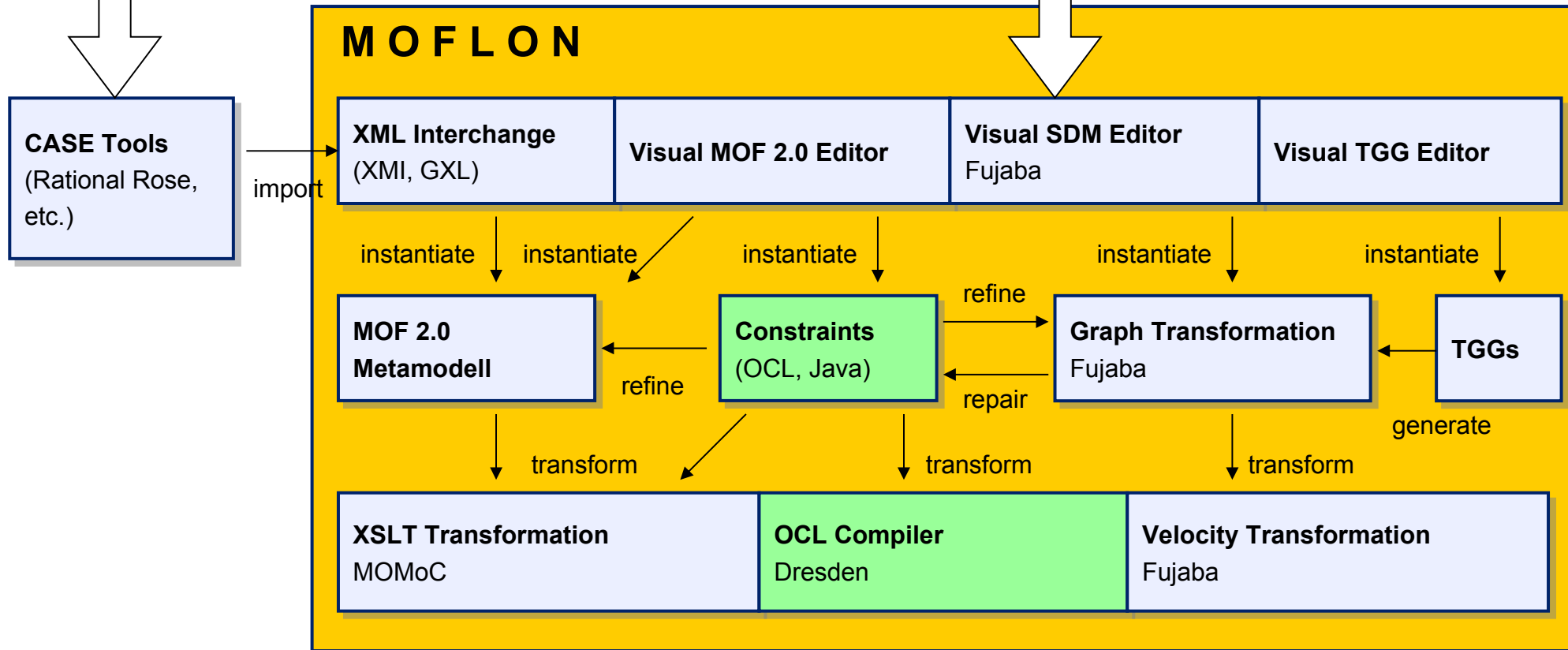


MOFLON – Architecture

20

Model-Driven Software Development in Technical Spaces (MOST)

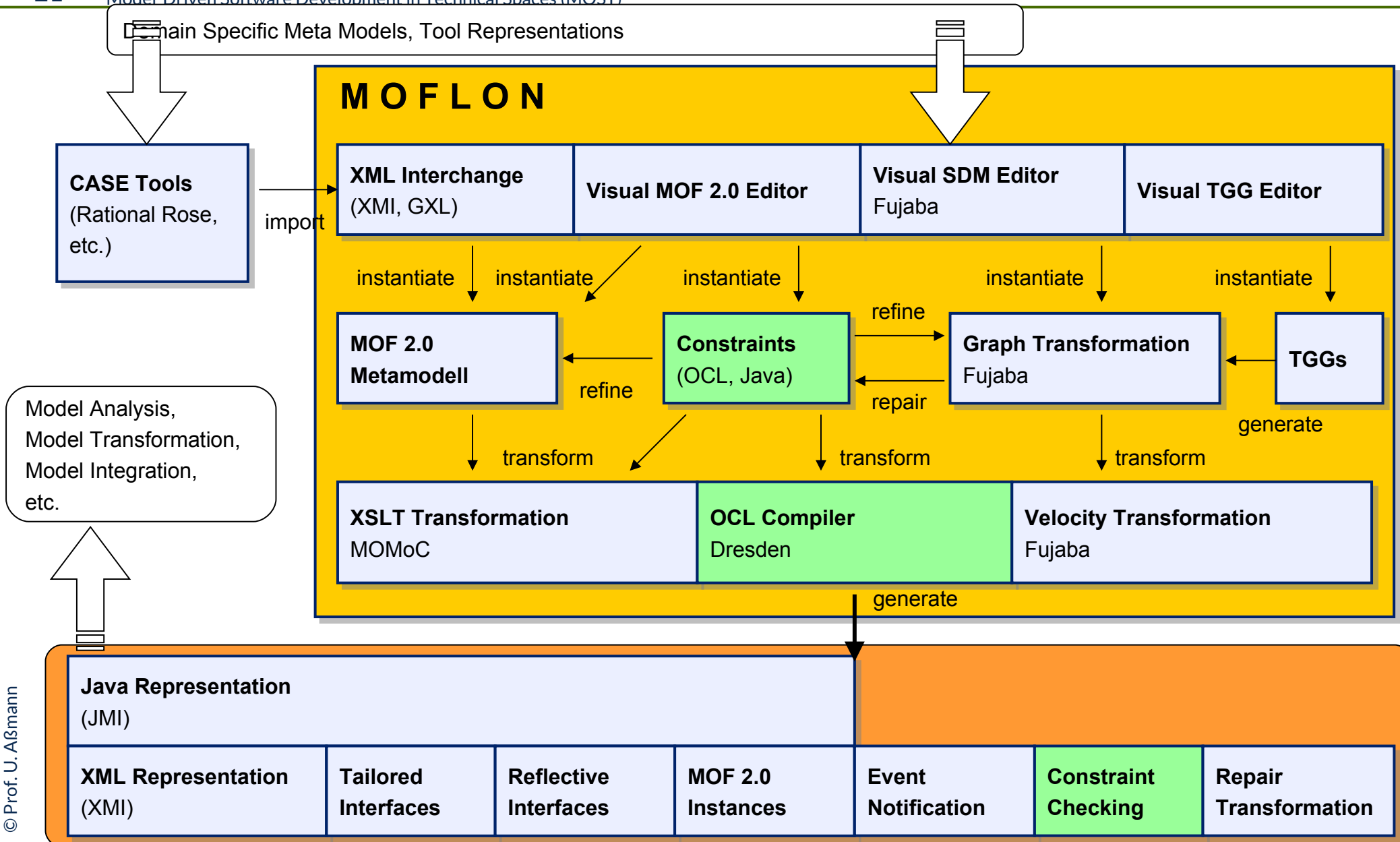
Domain Specific Meta Models, Tool Representations



MOFLON - Architecture

21

Model-Driven Software Development in Technical Spaces (MOST)



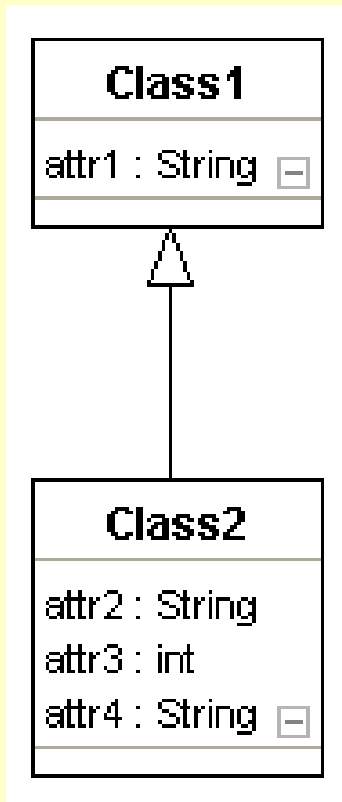
34.3. Triple Graph Grammars in MOFLON



34.3.1 Example 2: Integration with TGG – Object-Relational Mapping (ORM) from Class Diagrams to Database Schema

domain specific language,
e.g. Class Diagrams

domain specific language,
e.g. Database Schemata



Server: localhost Database: icgt2008 Table: class1

Browse Structure SQL Search Insert

	Field	Type	Collation	Attributes	Null
<input type="checkbox"/>	attr1	varchar(1024)	latin1_general_ci		No
<input type="checkbox"/>	attr2	varchar(1024)	latin1_general_ci		No
<input type="checkbox"/>	attr3	int(11)			No
<input type="checkbox"/>	attr4	varchar(1024)	latin1_general_ci		No

Table class1

Table class2

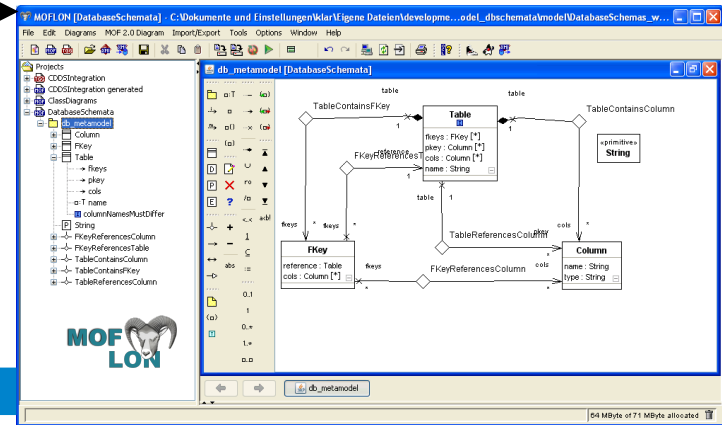
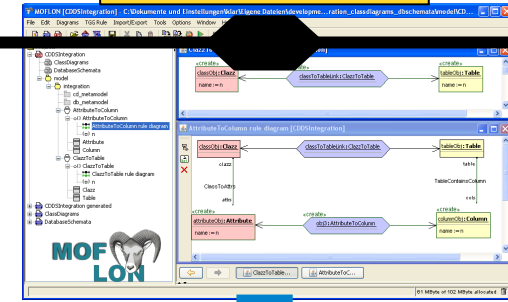
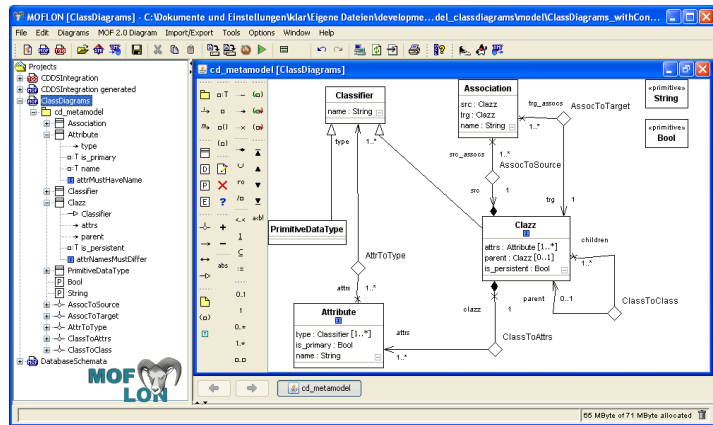


Example 2: Tool Integration Scenario TiE-CDDs: (ClassDiagrams / DatabaseSchema)

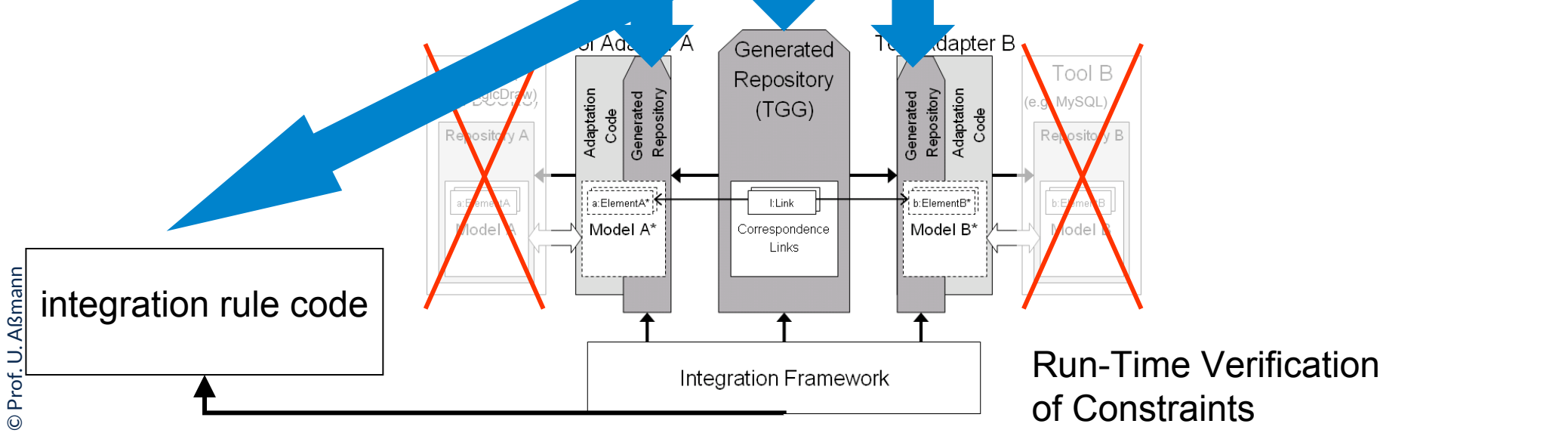
Class Diagrams Metamodel

TGGs relate

Database Schemata Metamodel



MOFLON generates



TiE-CDDS – Constraints in Class Diagrams (1)

Generate Code from MOF model (CD metamodel)

25

Model-Driven Software Development in Technical Spaces (MOST)

The image displays the MOFLON [ClassDiagrams] application interface. The main window shows a class diagram metamodel with classes: Classifier, Association, PrimitiveDataType, Attribute, and Classz. The Classz class is highlighted with a red circle. A context menu is open over the project browser, with 'Generate MOFLON-Code' selected. The 'Edit MOF Constraint' dialog is also visible, showing the constraint 'attrNamesMustDiffer' with the OCL body: `inv: attrs->forAll(a1, a2 : Attribute | a1 <> a2 implies a1.name <> a2.name)`.

MOFLON [ClassDiagrams] - C:\Dokumente und Einstellungen\klar\Eigene Dateien\developme..._del_classdiagrams\mode\ClassDiagrams_withCon...

MOFLON [ClassDiagrams] - C:\Dokumente und Einstellungen\klar\Eigene Dateien...

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

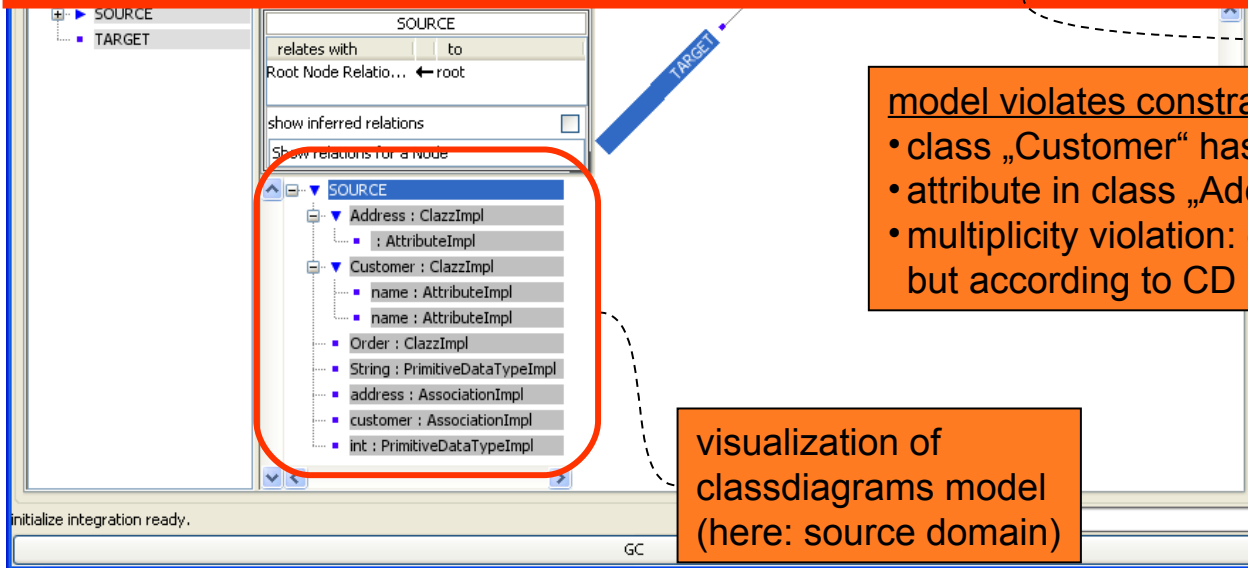
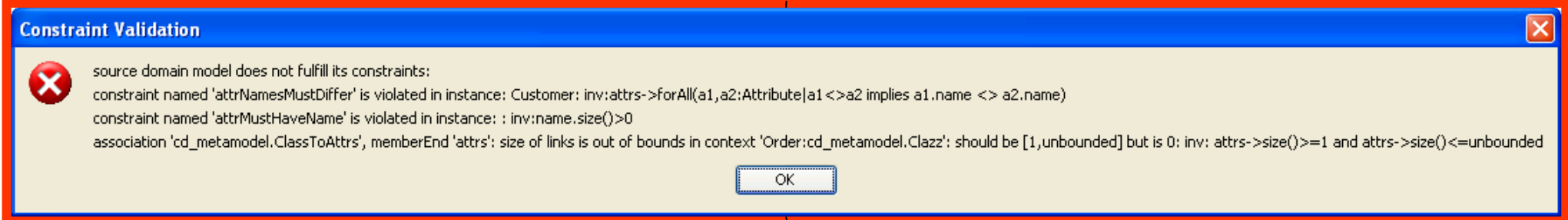
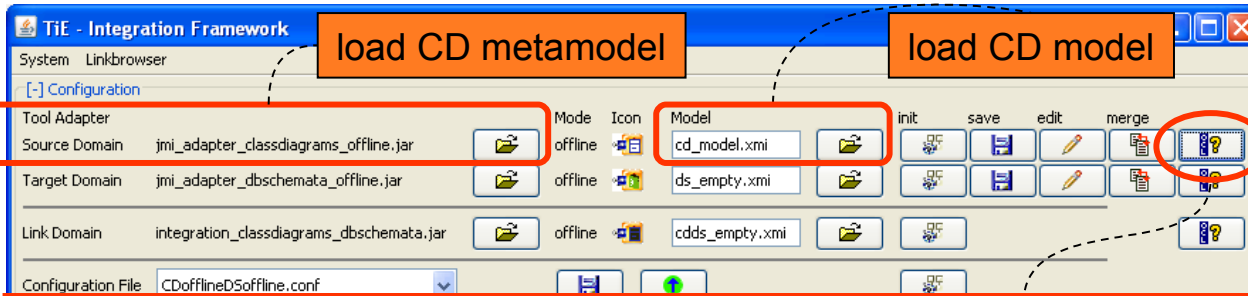
Ok Cancel

Generate MOFLON-Code (Schema + Transformations)



TiE-CDDS – Constraints in Class Diagrams (2)

Integration Framework



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 – Constraints in Class Diagrams (3)

Model Browser

27

Model-Driven Software Development in Technical Spaces (MOST)

© Prof. U. Aßmann

initialize integration ready.

model is fixed in generic model editor

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

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

TiE-CDDS – Constraints in Class Diagrams (4)

Integration Framework

System Linkbrowser

[-] Configuration

Tool Adapter	Mode	Icon	Model	init	save	edit	merge
Source Domain: jmi_adapter_classdiagrams_offline.jar	offline		cd_model.xml				
Target Domain: jmi_adapter_dbschemata_offline.jar	unknown		ds_empty.xml				
Link Domain: integration_classdiagrams_dbschemata.jar	unknown		cdds_empty.xml				

Configuration File: CDofflineDSoffline.conf

[-] Action

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

Configuration File: last.conf

[-] Output

LinkBrowser Log

root

- SOURCE
- TARGET

Close Up View CircleView

relates with to

show inferred relations

Show relations for a Node

SOURCE

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

initialize source ready.

GC

translation process may start now...

Constraint Validation

source domain model fulfills its constraints

OK



TiE-CDDS – Constraints in Class Diagrams (5)

Forward Translation to DB representation

29

Model-Driven Software Development in Technical Spaces (MOST)

The image displays two screenshots of the TiE - Integration Framework software interface, illustrating the forward translation process from class diagrams to a database representation.

Left Screenshot (Configuration and Action Settings):

- System Linkbrowser:** Configuration for Tool Adapter, Source Domain, Target Domain, and Link Domain.
- Configuration File:** CDOfflineDSOffline.conf
- Action:** Algorithm: Forward Translation (Batch, Simple); Strategy: Unsorted Simple; Log Level: WARN.
- Output:** LinkBrowser showing SOURCE and TARGET nodes.
- Close Up View:** Details of the SOURCE node, including classes like Address, Customer, and Order, and their attributes.

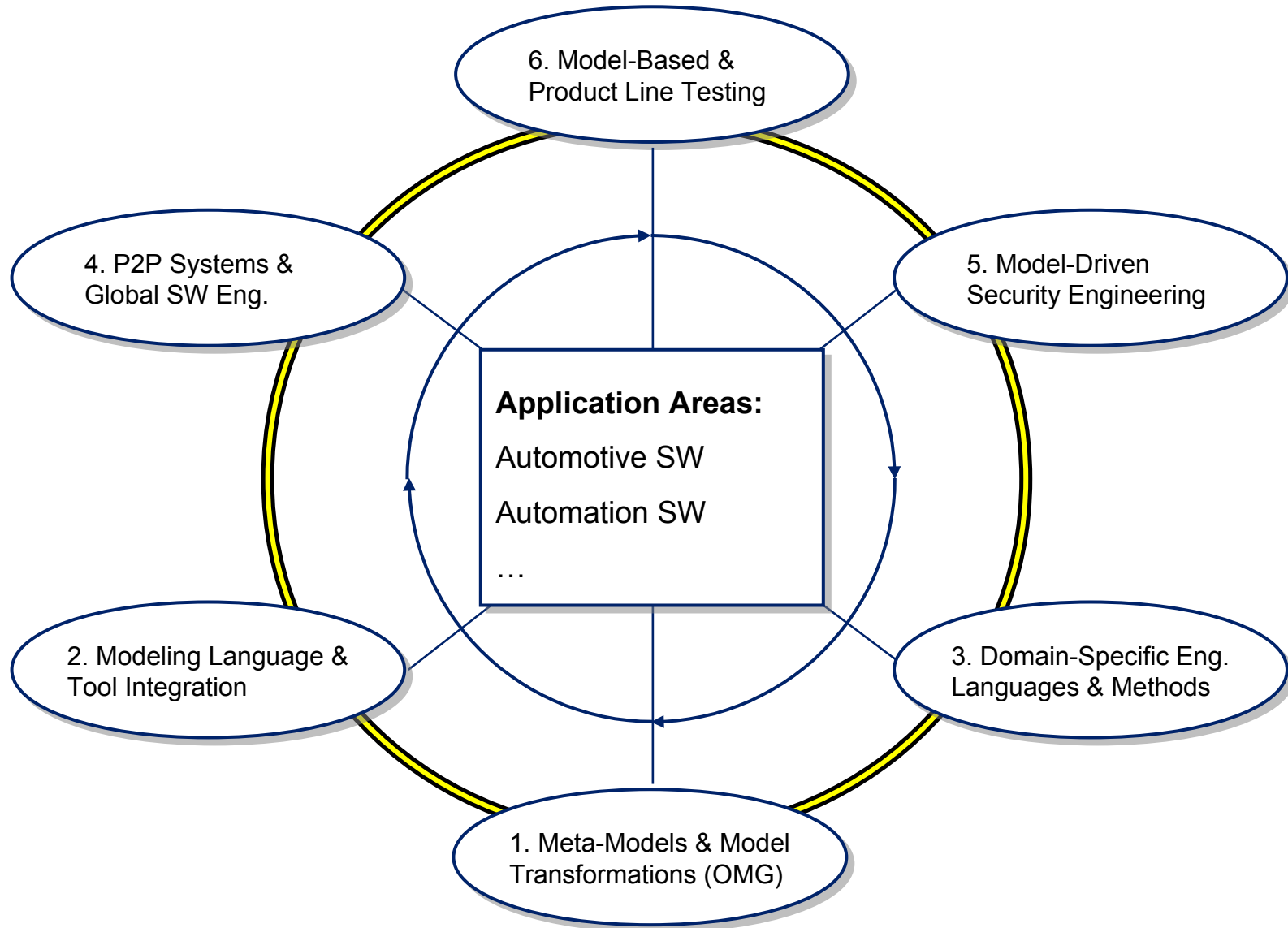
Right Screenshot (Output and Database Representation):

- System Linkbrowser:** Configuration for Tool Adapter, Source Domain, Target Domain, and Link Domain.
- Configuration File:** last.conf
- Output:** LinkBrowser showing SOURCE and TARGET nodes.
- Close Up View:** Details of the SOURCE node, including classes like Address, Customer, and Order, and their attributes.
- Database Representation:** A diagram showing the mapping of source classes to database tables. The source classes are mapped to tables: Address (Table), Customer (Table), and Order (Table). The attributes are mapped to columns: street (Column), name (Column), surname (Column), and id (Column). The diagram also shows the relationships between these tables and columns.

MOFLON is Bootstrapped

- ▶ TU Darmstadt bootstraps the MOFLON MOF Metamodel periodically
 - Since 2013, ported to EMOF
- Bootstrap has important advantages:
 - If more OCL constraints are added to the (e)MOF metamodel
 - Regenerate MOFLON MOF implementation
 - Activate the extended constraint checking in MOFLON (model verification, model consistency checking, model wellformedness)

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



Related Approaches

standards	approaches based on graph-/modeltransformation					classic meta-CASE approaches					text based approaches				
	MOF, OCL, QVT	Fujaba & TGG MOFLON	Progres & TGG	GME & TGG	EMF & GReAT	EMF & Tefkat	AToM ³	Microsoft MetaEdit+	EMF & DSL	Pounamu	EBNF & TXL DiaGen	SQL	XML		
Abstract syntax	+	+	+	+	0	0	0	+	+	0	+	+	+	0	+
Concrete syntax	--	--	--	+	+	--	+	+	+	+	+	+	--	--	--
Static semantics	+	+	0	+	+	+	0	0	--	+	0	+	0	0	--
Dynamic semantics	+	+	+	+	+	+	+	0	0	--	--	--	+	--	0
Model analysis	+	+	+	+	+	0	+	0	--	+	--	0	+	0	+
Model transformation	+	+	+	+	+	+	+	0	--	--	--	0	+	0	+
Model integration	+	+	+	+	0	+	--	--	--	--	--	0	--	0	0
Acceptability	+	+	0	--	0	+	--	+	--	0	+	0	0	+	+
Scaleability	+	+	--	0	--	0	--	0	--	--	--	--	--	--	0
Tool availability	--	0	0	+	+	+	+	+	0	0	+	+	+	+	0
Expressiveness	+	+	0	+	+	0	0	0	0	0	0	0	+	0	0

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. Röttschke, 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.

The End

Some slides are courtesy Florian Heidenreich and Felix Klar

Thank you for your attention...



<http://www.moflon.org>

