43. Das Meta-CASE-Tool MOFLON

1

Prof. Dr. Uwe Aßmann
Technische Universität Dresden
Institut für Software- und Multimediatechnik
http://st.inf.tu-dresden.de
Version 12-1.1, 05.01.13

1) MOFLON Meta-CASE-Werkzeug

Reading

MOFLON Website
http://www.moflon.org

MOFLON Training
http://moflon.org/documentation/links.html

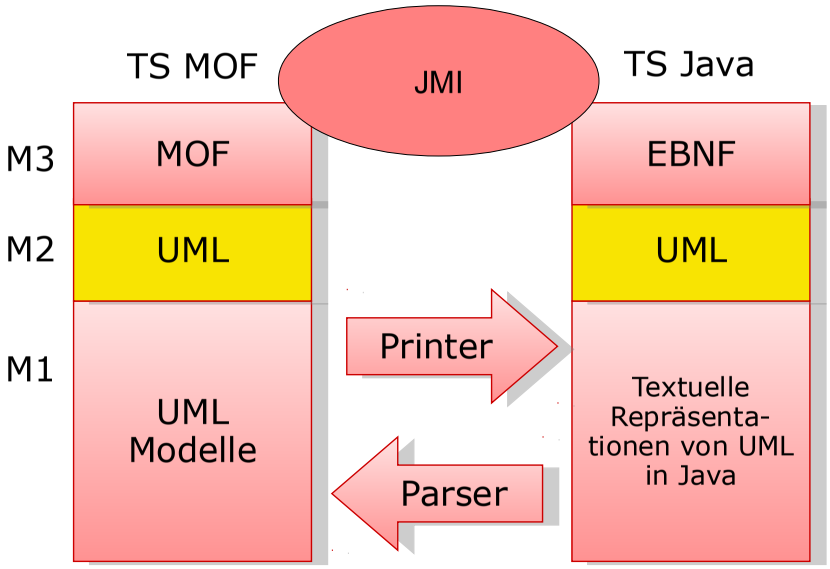
MOFLON Tutorial
http://moflon.org/documentation/tutorial.html

43.3.1. MOFLON Einführung

- MOFLON ist ein Metamodellierungswerkzeug der TU Darmstadt, Fachgruppe Echtzeitsysteme, Prof. Andy Schürr
  - MOFLON nutzt Logik (OCL) zum Checking von Wohlgeformtheitsbedingungen über Modellen (AC-Werkzeug)
  - MOFLON ist eine Fujaba-Erweiterung und bietet daher Graphersetzungssysteme an www.fujaba.de (M-Werkzeug)
  - MOFLON unterstützt Triple Graph Grammars (TGG, siehe ST-II)
- MOFLON unterstützt
  - MOF 2.0
  - OCL 2.0
  - JMI 1.4
  - XMI 2.1

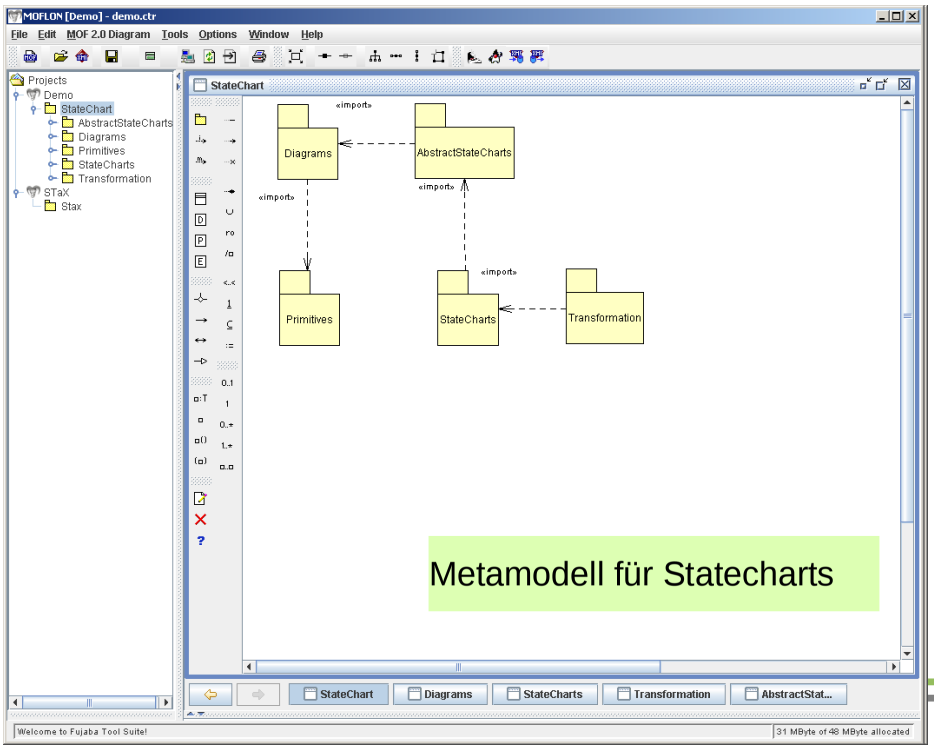
Codegenerierung mit JMI, einer transformative TS-Brücke für MOF und Java, Sprache UML

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

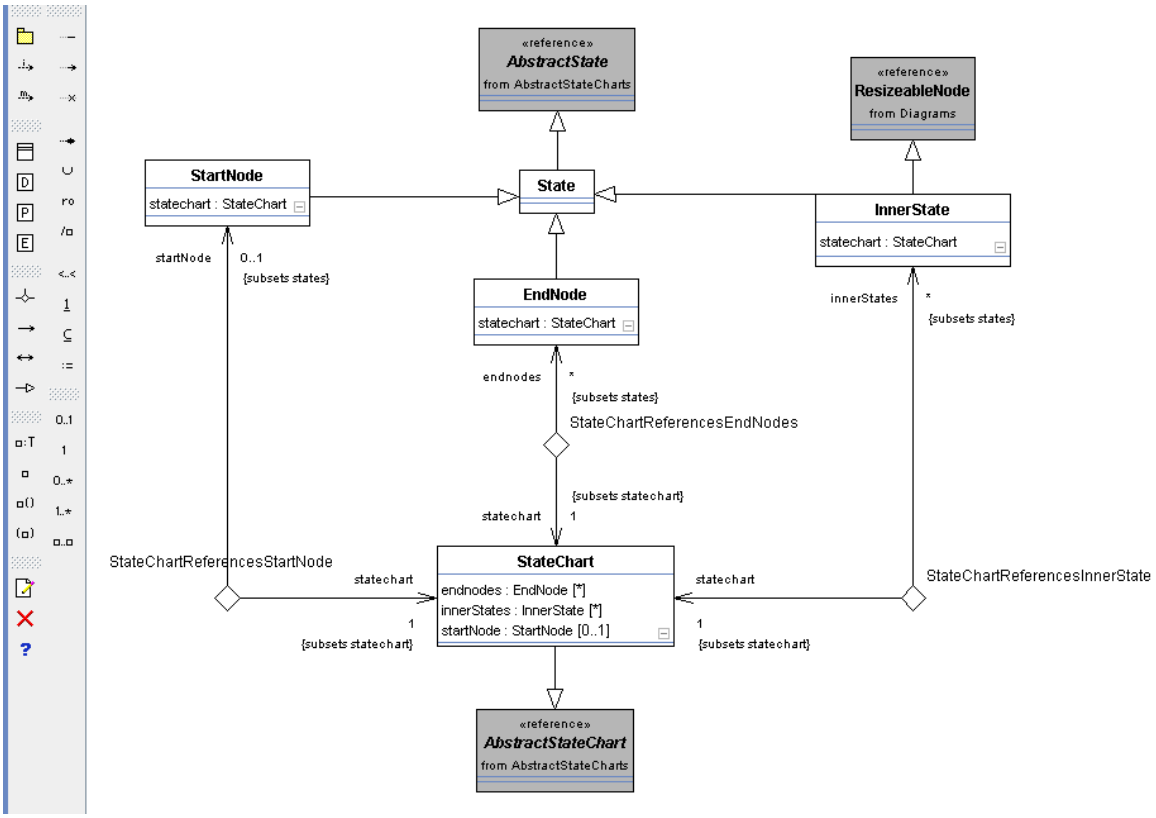


MOFLON Beispiel 1: Metamodell für Statecharts: Vorgehensweise

1) Metamodell erstellen
2) Code generieren (Repository, Constraint-checker)
3) Code über JMI-Schnittstellen verwenden

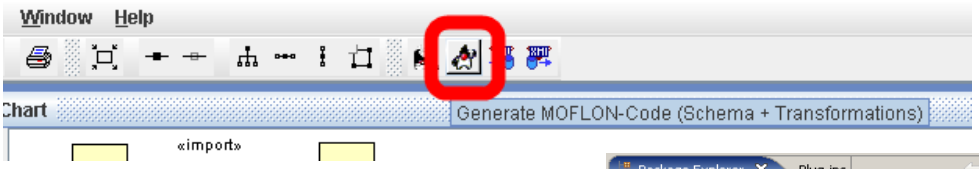


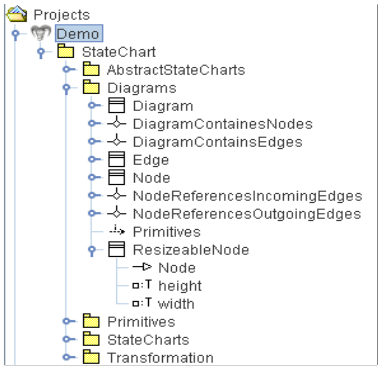
Beispiel: 1.a) Erstellung eines MOF-Metamodells für Statecharts

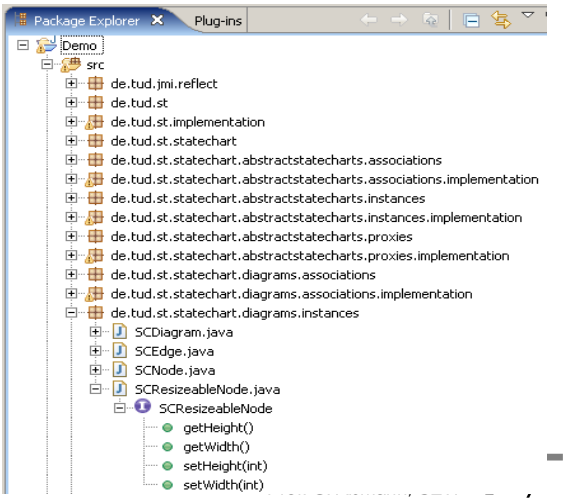


Beispiel: 1.b) Codegenerierung aus Metamodell für Statechart-Modelle

- Erzeugt JMI-Schnittstellen zum Metamodell (metamodellgesteuertes Repositorium)
- Generiert Code für alle als Story-Diagramm (Fujaba) modellierten Methoden
- Codegenerator verwendet Velocity und XSLT 1.1







Beispiel: 1.b) Codegenerierung aus Metamodell für Statechart-Modelle

Code generieren

Pro Package

- Java Paket
- Schnittstelle
- Implementierung

de.tud.st.statechart
SCStateChartPackage.java
SCStateChartPackageImpl.java

Pro Klasse

- Schnittstelle
- Implementierung
- Proxy Schnittstelle
- Proxy Implementierung

SCNode.java
SCNodeImpl.java
SCNodeClass.java
SCNodeClassImpl.java

Pro Assoziation

- Schnittstelle
- Implementierung

SCDiagramContainsEdges.java
SCDiagramContainsEdgesImpl.java

Beispiel: 1.c) Codeverwendung von Statechart-Modellen

- Wurzelpaket instanzieren

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

- Proxy anfordern

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

- Über den Proxy Instanzen erzeugen

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


43.3.2. The Metamodeling Architecture of MetaCASE Tool MOFLON

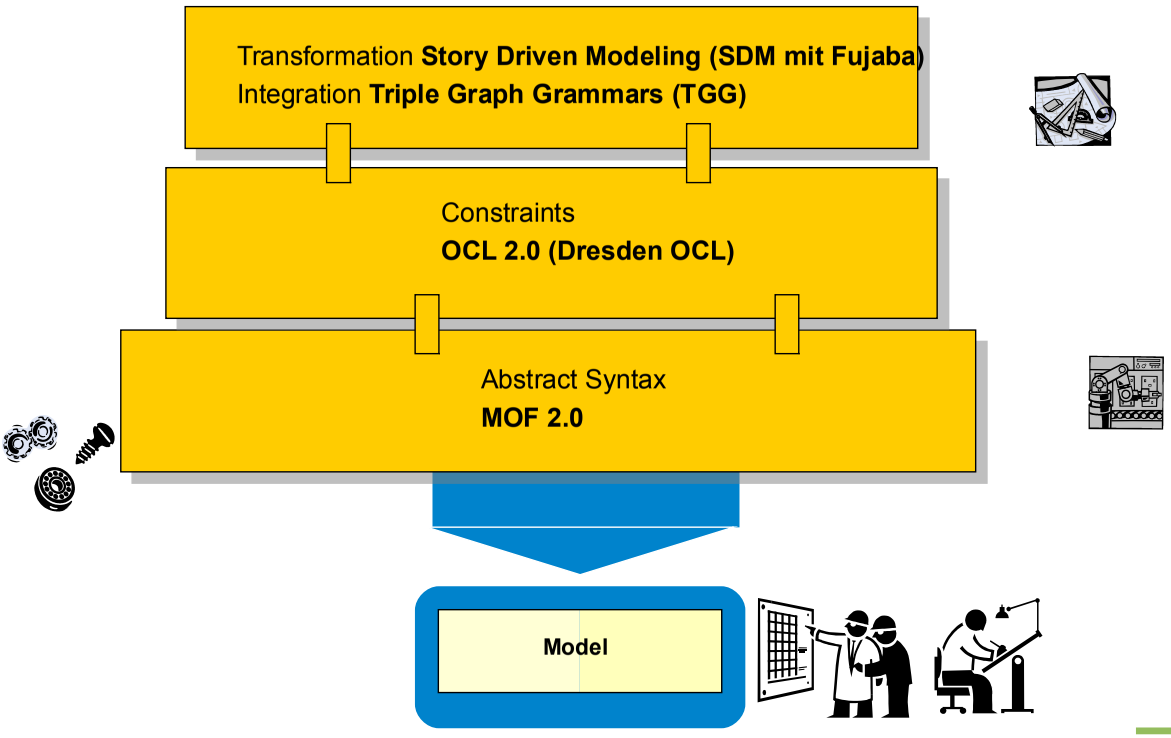
TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Metamodel Architecture of MOFLON



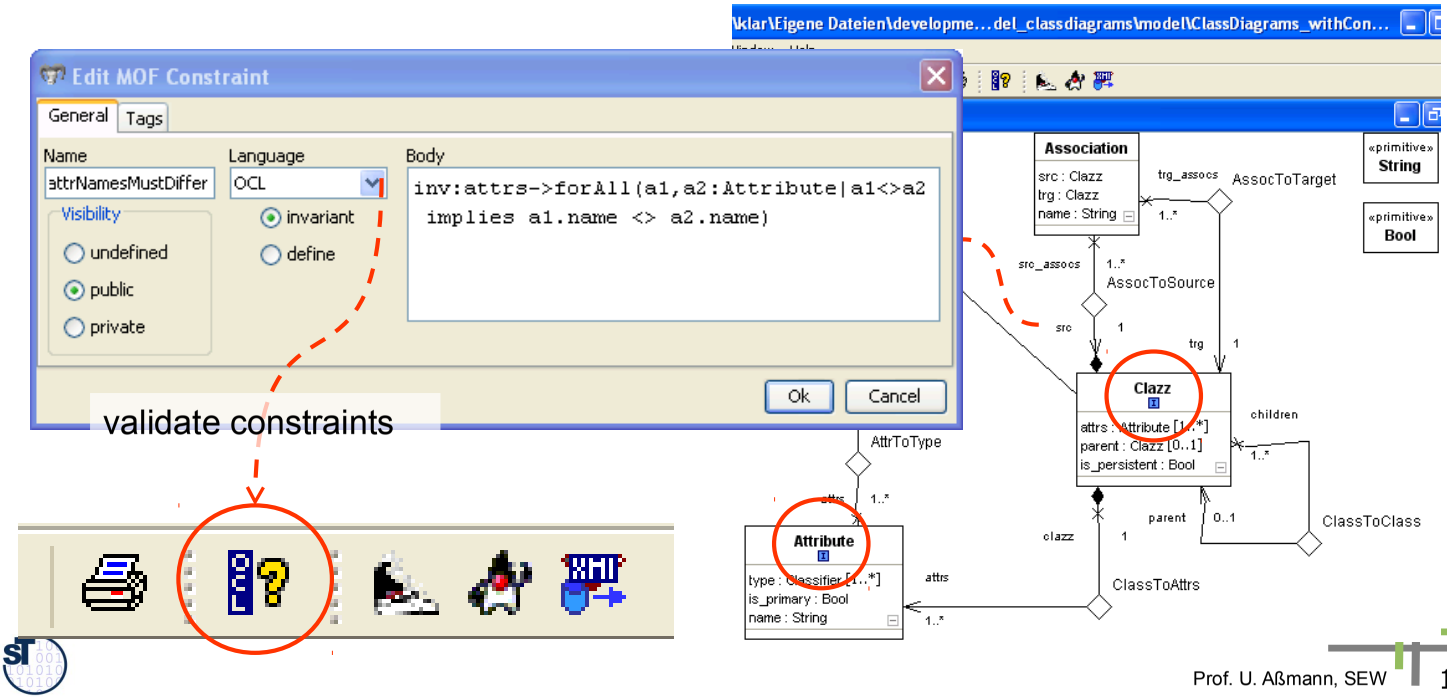
MOFLON MetaCASE – Main Features

- 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

(OCL) Constraints in MOFLON – MOF Editor

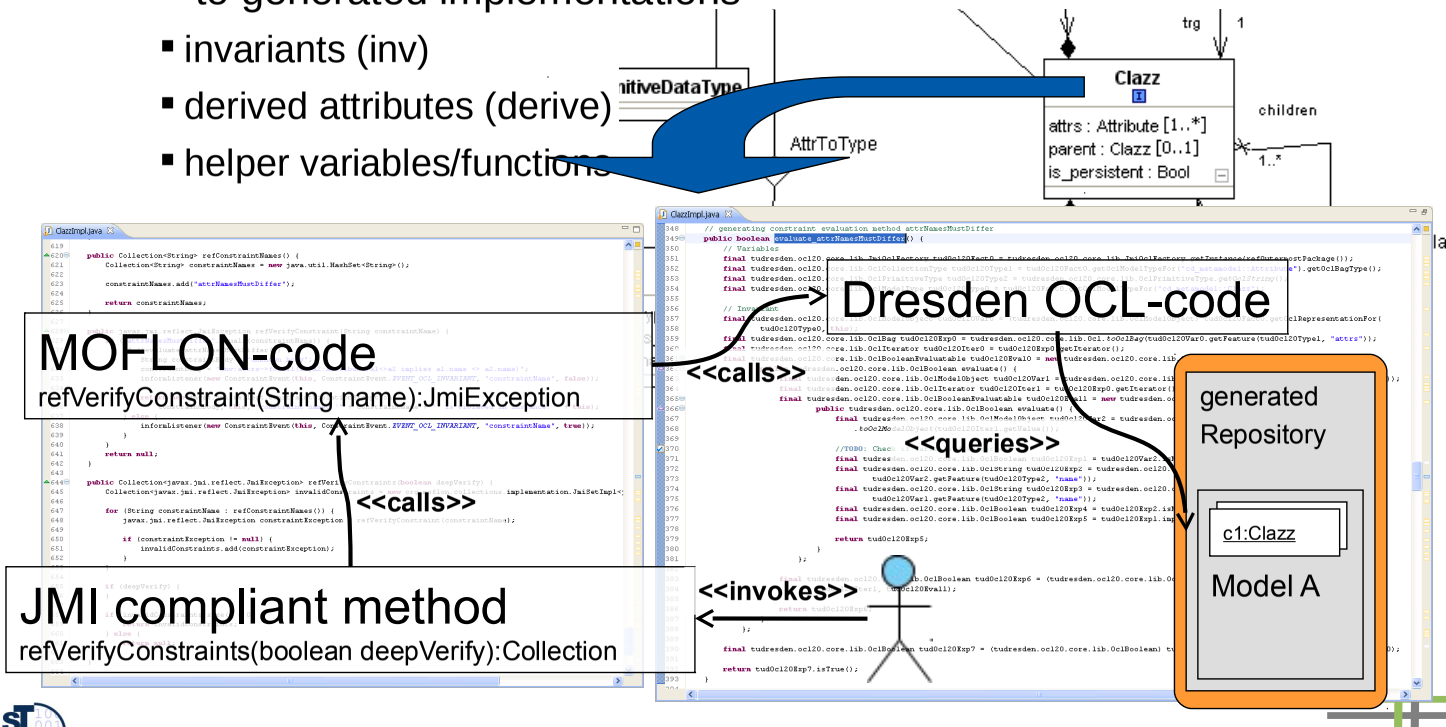
- MOF allows to add constraints to every MOF element
- MOFLON has an underlying MOF metamodel repository

→ MOFLON MOF editor may add constraints to elements



(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



ClazzImpl.java

```
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
```

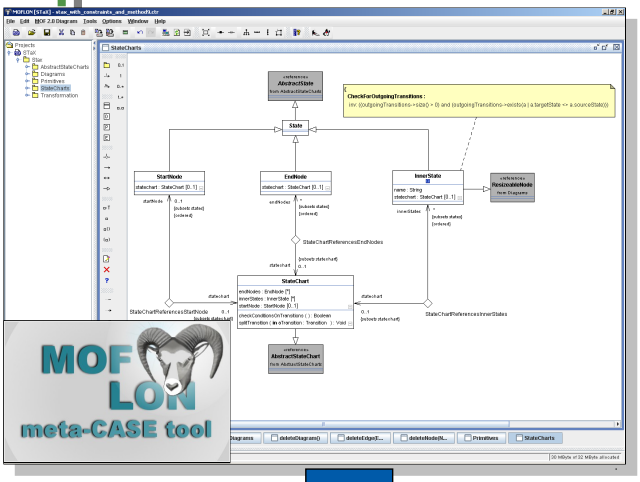
JMI compiliant method

ClazzImpl.java

```
348     // generating constraint evaluation method attrNamesMustDiffer
349     public boolean evaluate_attrNamesMustDiffer() {
350         // Variables
351         final tudresden.ocl20.core.lib.JmiOclFactory tud0cl20Fact0 = tudresden.ocl20.core.lib.JmiOclFactory.getInstance(refOutermostPackage());
352         final tudresden.ocl20.core.lib.OclCollectionType tud0cl20Type1 = tud0cl20Fact0.getOclModelTypeFor("cd_metamodel::Attribute").getOclBagType();
353         final tudresden.ocl20.core.lib.OclPrimitiveType tud0cl20Type2 = tudresden.ocl20.core.lib.OclPrimitiveType.getOclString();
354         final tudresden.ocl20.core.lib.OclModelType tud0cl20Type0 = tud0cl20Fact0.getOclModelTypeFor("cd_metamodel::Clazz");
355
356         // Invariant
357         final tudresden.ocl20.core.lib.OclModelObject tud0cl20Var0 = (tudresden.ocl20.core.lib.OclModelObject) tud0cl20Fact0.getOclRepresentationFor(
358                 tud0cl20Type0, this);
359         final tudresden.ocl20.core.lib.OclBag tud0cl20Exp0 = tudresden.ocl20.core.lib.Ocl.toOclBag(tud0cl20Var0.getFeature(tud0cl20Type1, "attrs"));
360         final tudresden.ocl20.core.lib.OclIterator tud0cl20Iter0 = tud0cl20Exp0.getIterator();
361         final tudresden.ocl20.core.lib.OclBooleanEvaluatable tud0cl20Eval0 = new tudresden.ocl20.core.lib.OclBooleanEvaluatable() {
362             public tudresden.ocl20.core.lib.OclBoolean evaluate() {
363                 final tudresden.ocl20.core.lib.OclModelObject tud0cl20Var1 = tudresden.ocl20.core.lib.Ocl.toOclModelObject(tud0cl20Iter0.getValue());
364                 final tudresden.ocl20.core.lib.OclIterator tud0cl20Iter1 = tud0cl20Exp0.getIterator();
365                 final tudresden.ocl20.core.lib.OclBooleanEvaluatable tud0cl20Eval1 = new tudresden.ocl20.core.lib.OclBooleanEvaluatable() {
366                     public tudresden.ocl20.core.lib.OclBoolean evaluate() {
367                         final tudresden.ocl20.core.lib.OclModelObject tud0cl20Var2 = tudresden.ocl20.core.lib.Ocl
368                             .toOclModelObject(tud0cl20Iter1.getValue());
369
370                         //TODO: Check if VariableId is correct
371                         final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp1 = tud0cl20Var2.isNotEqualTo(tud0cl20Var1);
372                         final tudresden.ocl20.core.lib.OclString tud0cl20Exp2 = tudresden.ocl20.core.lib.Ocl.toOclString(
373                                 tud0cl20Var2.getFeature(tud0cl20Type2, "name"));
374                         final tudresden.ocl20.core.lib.OclString tud0cl20Exp3 = tudresden.ocl20.core.lib.Ocl.toOclString(
375                                 tud0cl20Var1.getFeature(tud0cl20Type2, "name"));
376                         final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp4 = tud0cl20Exp2.isNotEqualTo(tud0cl20Exp3);
377                         final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp5 = tud0cl20Exp1.implies(tud0cl20Exp4);
378
379                         return tud0cl20Exp5;
380                     }
381                 };
382
383                 final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp6 = (tudresden.ocl20.core.lib.OclBoolean) tud0cl20Exp0.forAll(
384                         tud0cl20Iter1, tud0cl20Eval1);
385
386                 return tud0cl20Exp6;
387             }
388         };
389
390         final tudresden.ocl20.core.lib.OclBoolean tud0cl20Exp7 = (tudresden.ocl20.core.lib.OclBoolean) tud0cl20Exp0.forAll(tud0cl20Iter0, tud0cl20Eval0);
391
392         return tud0cl20Exp7.isTrue();
393     }
```

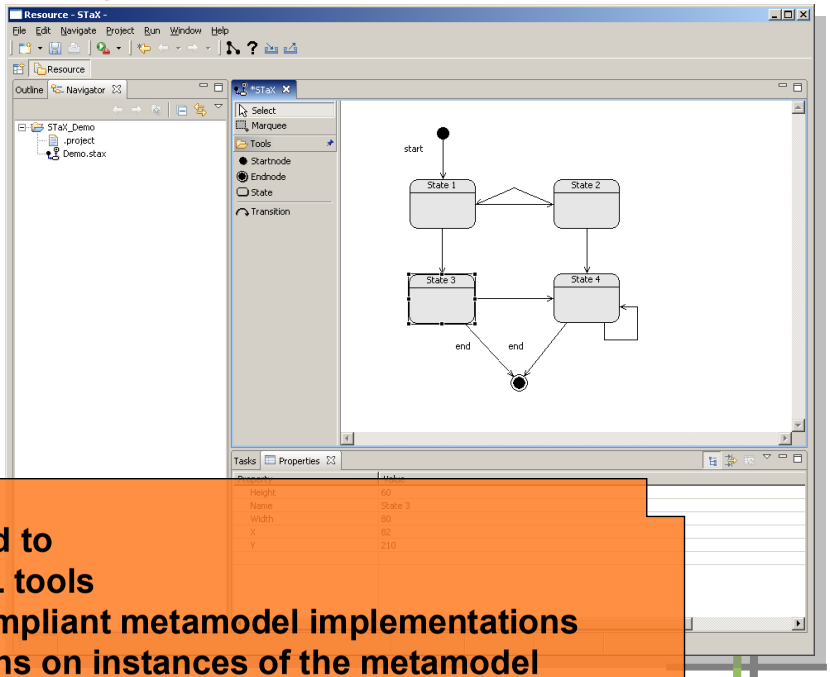
Generated Code from Dresden OCL

Result of MOFLON Example 1 – Statechart Editor (STaX)



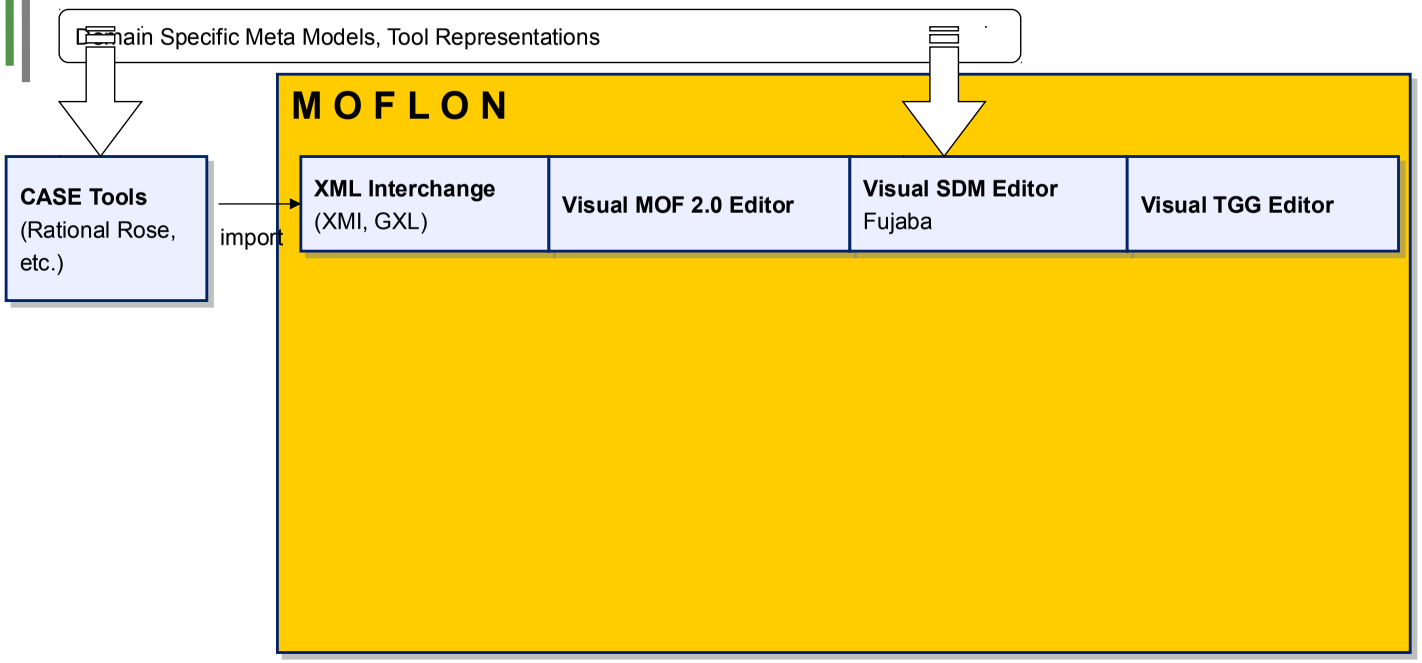
+

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

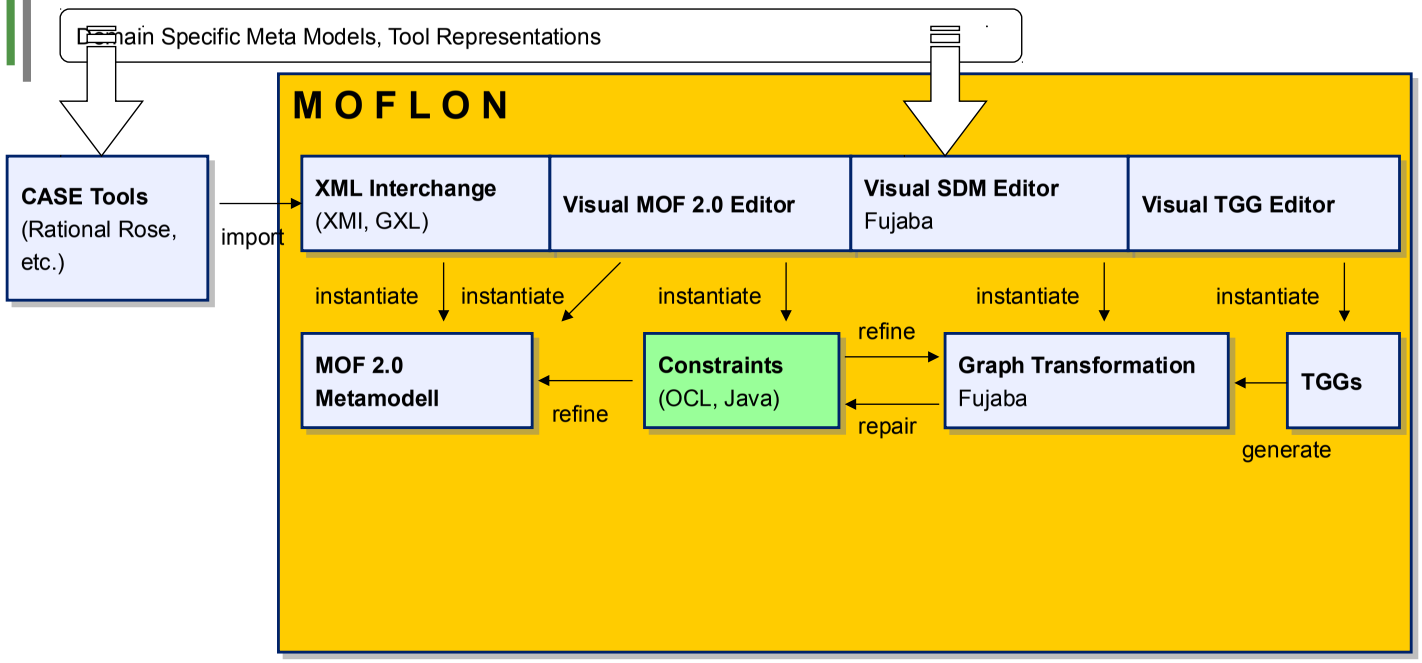


MOFLON is mainly used to
- **integrate existing DSL tools**
- **generate standard compliant metamodel implementations**
- **specify transformations on instances of the metamodel**

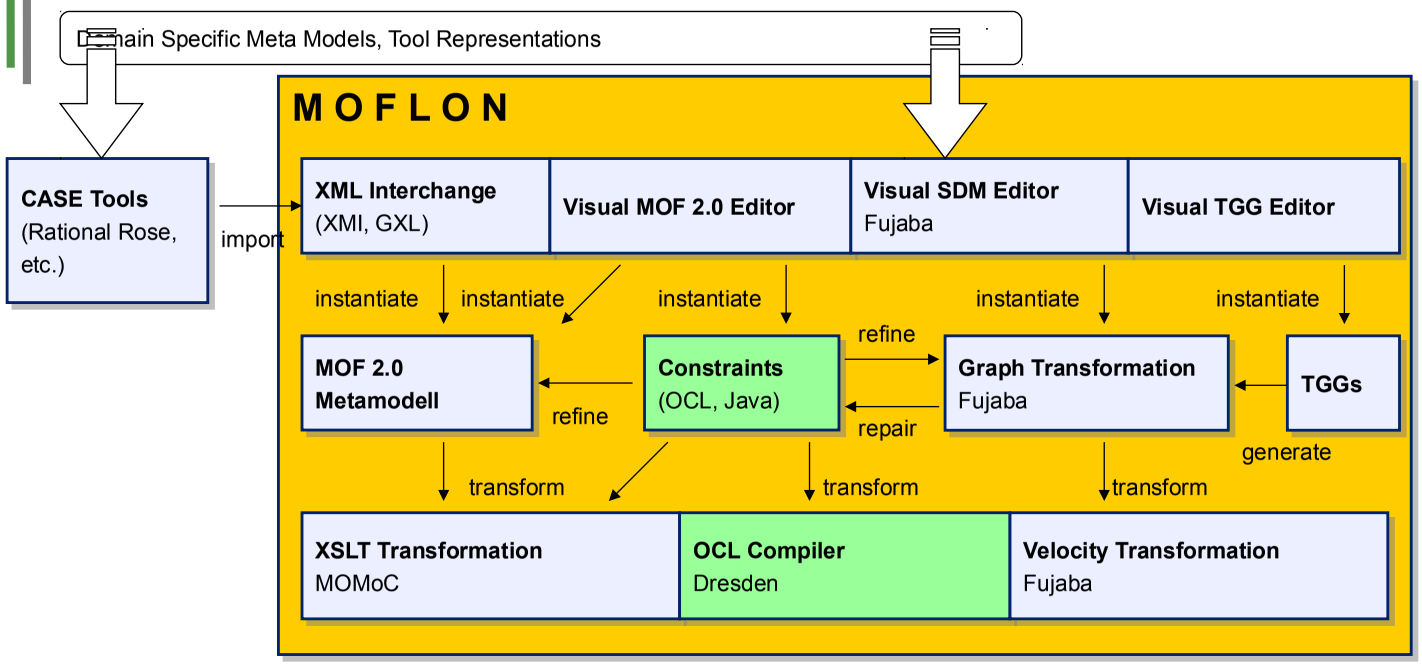
43.3.3 MOFLON – Architecture



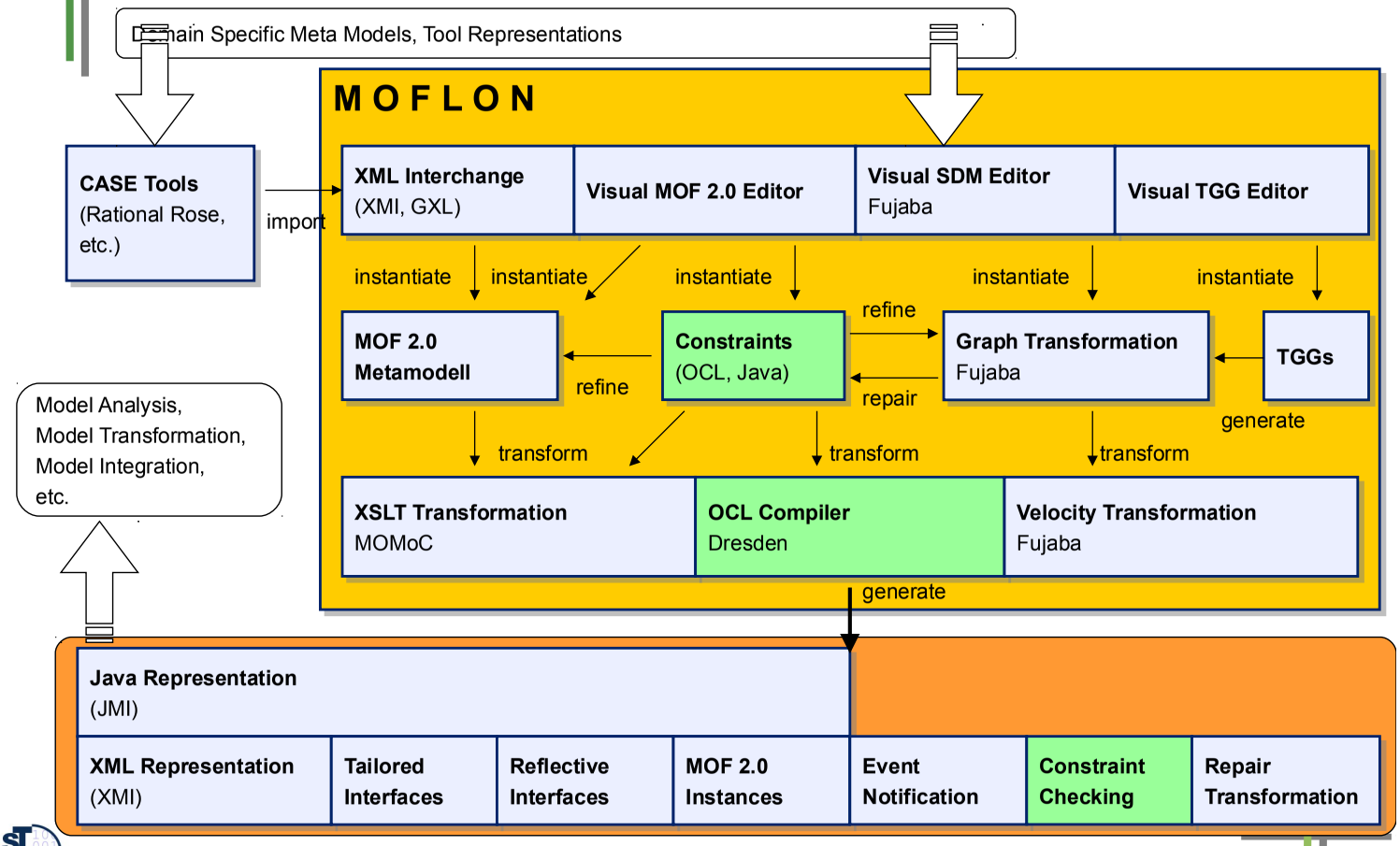
MOFLON – Architecture



MOFLON – Architecture



MOFLON – Architecture



43.3.4 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

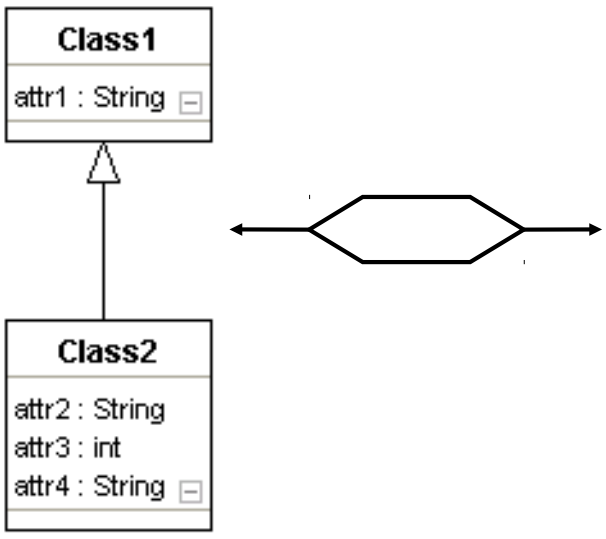


Table class1

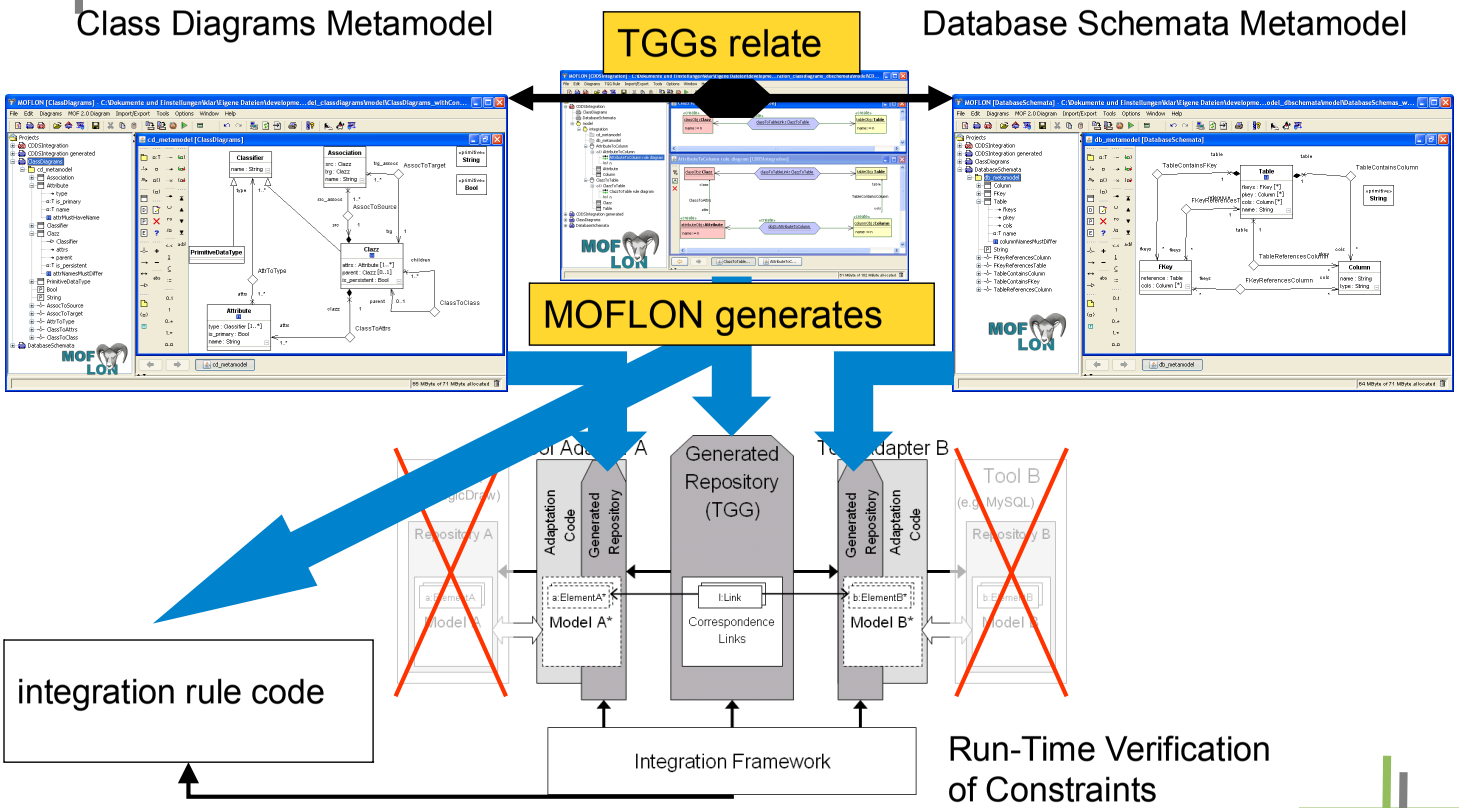
Server: localhost ▸ Database: icgt2008 ▸ Table: class1

Browse | Structure | SQL | Search | Insert

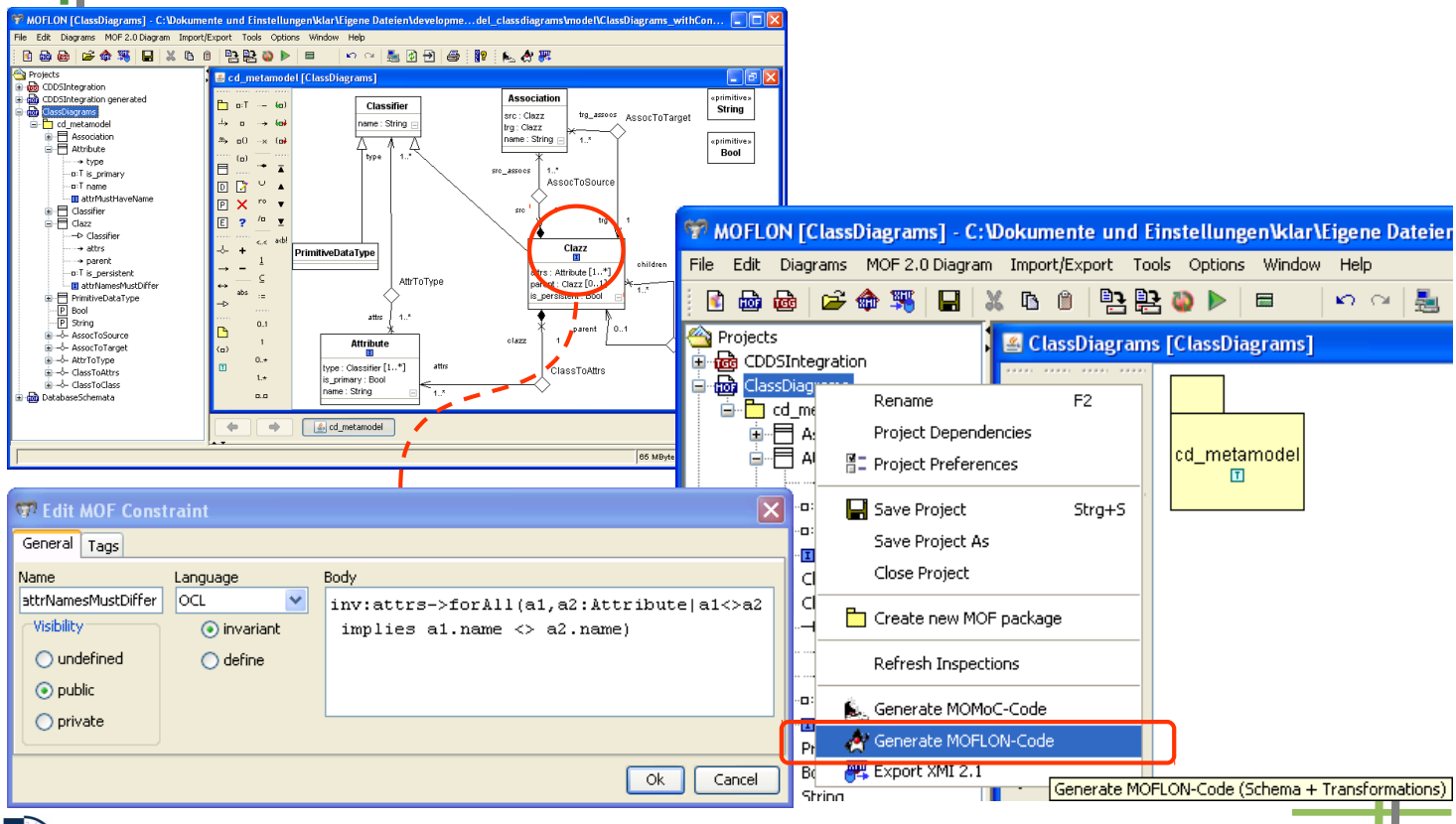
	Field	Type	Collation	Attributes	Null
☐	attr1	varchar(1024)	latin1_general_ci		No
☐	attr2	varchar(1024)	latin1_general_ci		No
☐	attr3	int(11)			No
☐	attr4	varchar(1024)	latin1_general_ci		No

Table class2

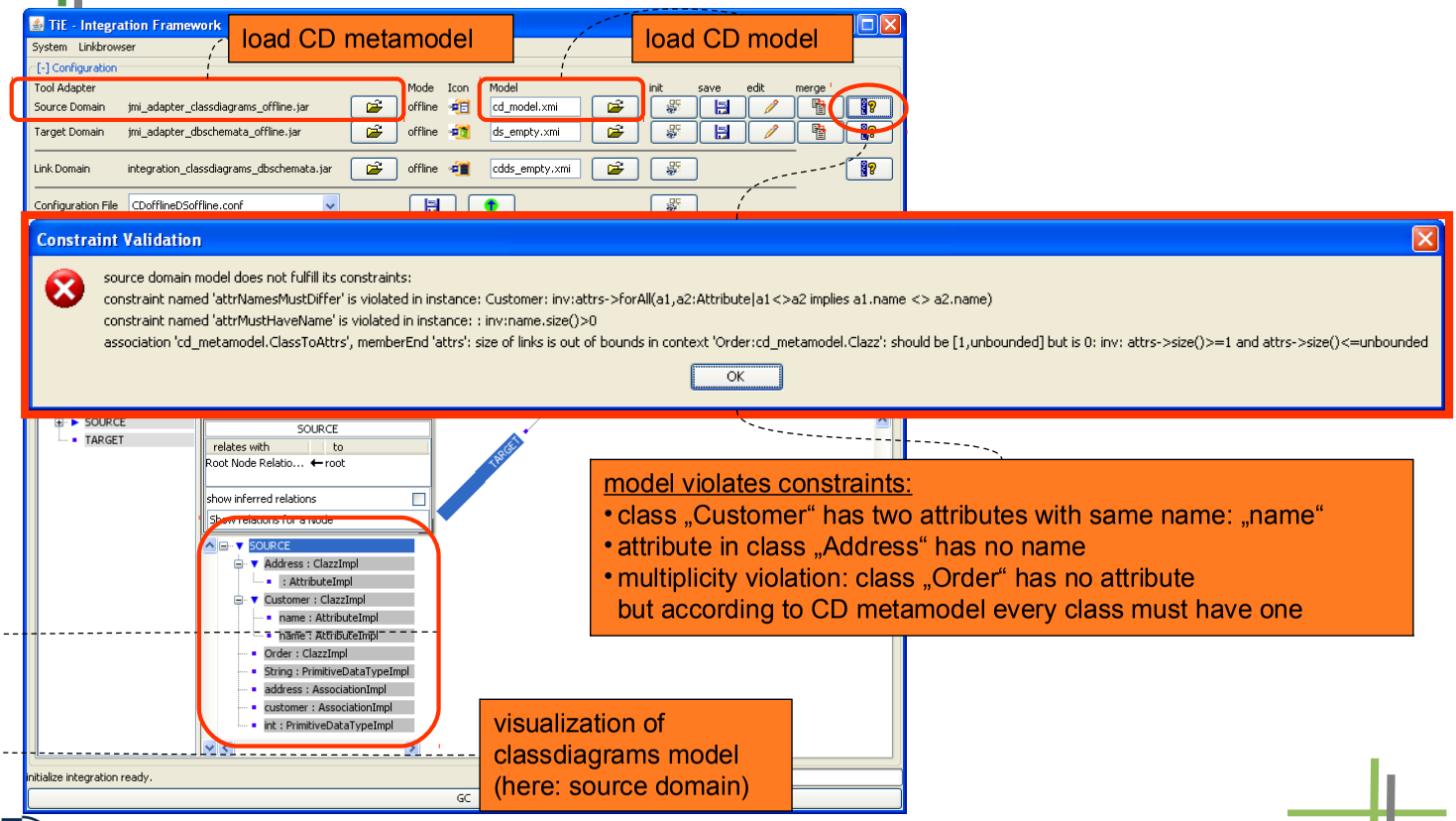
Example 2: Tool Integration Scenario TiE-CDDS: (ClassDiagrams / DatabaseSchema)



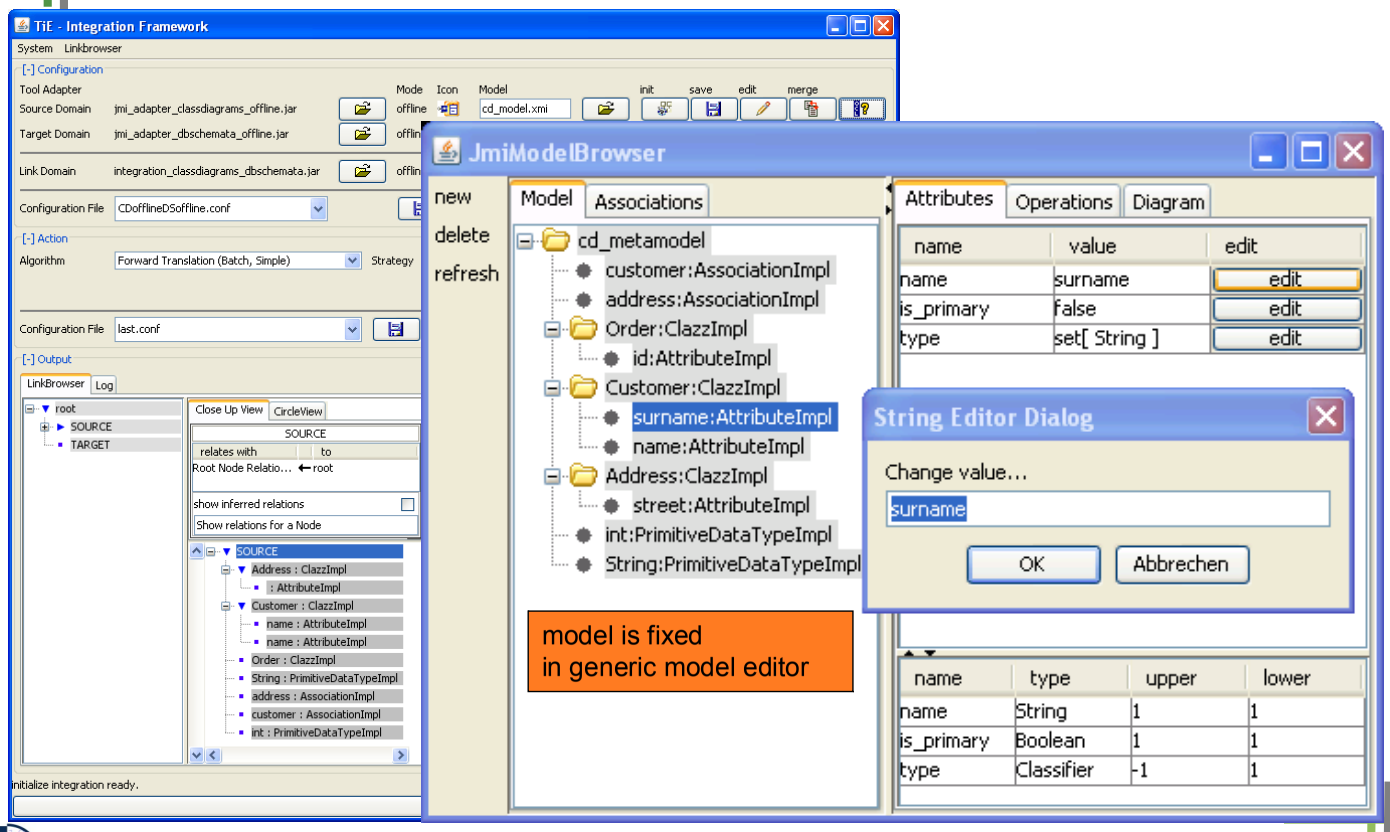
TiE-CDDS – Constraints in Class Diagrams (1) Generate Code from MOF model (CD metamodel)



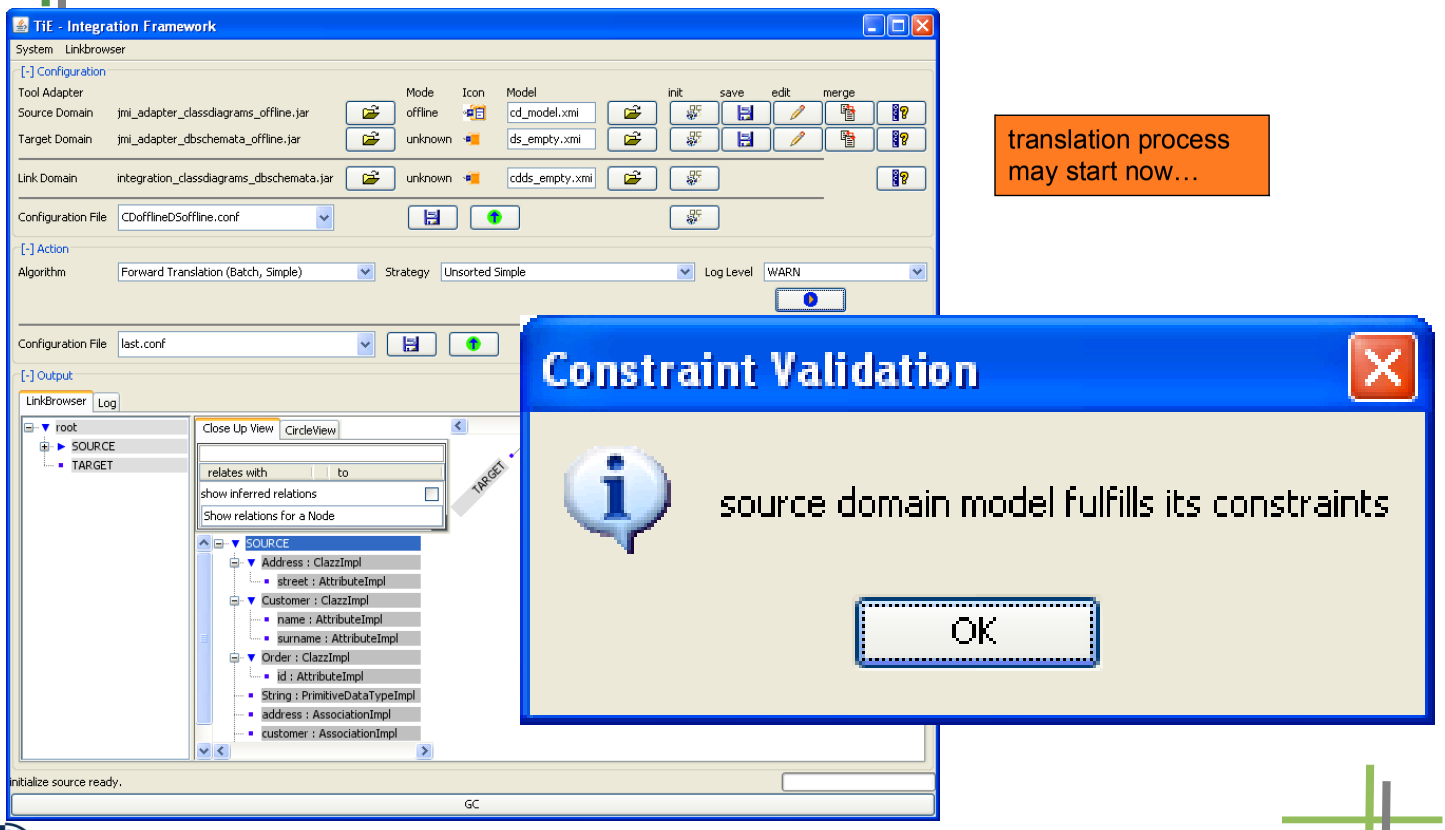
TiE-CDDS – Constraints in Class Diagrams (2)
Integration Framework



TiE-CDDS – Constraints in Class Diagrams (3)
Model Browser

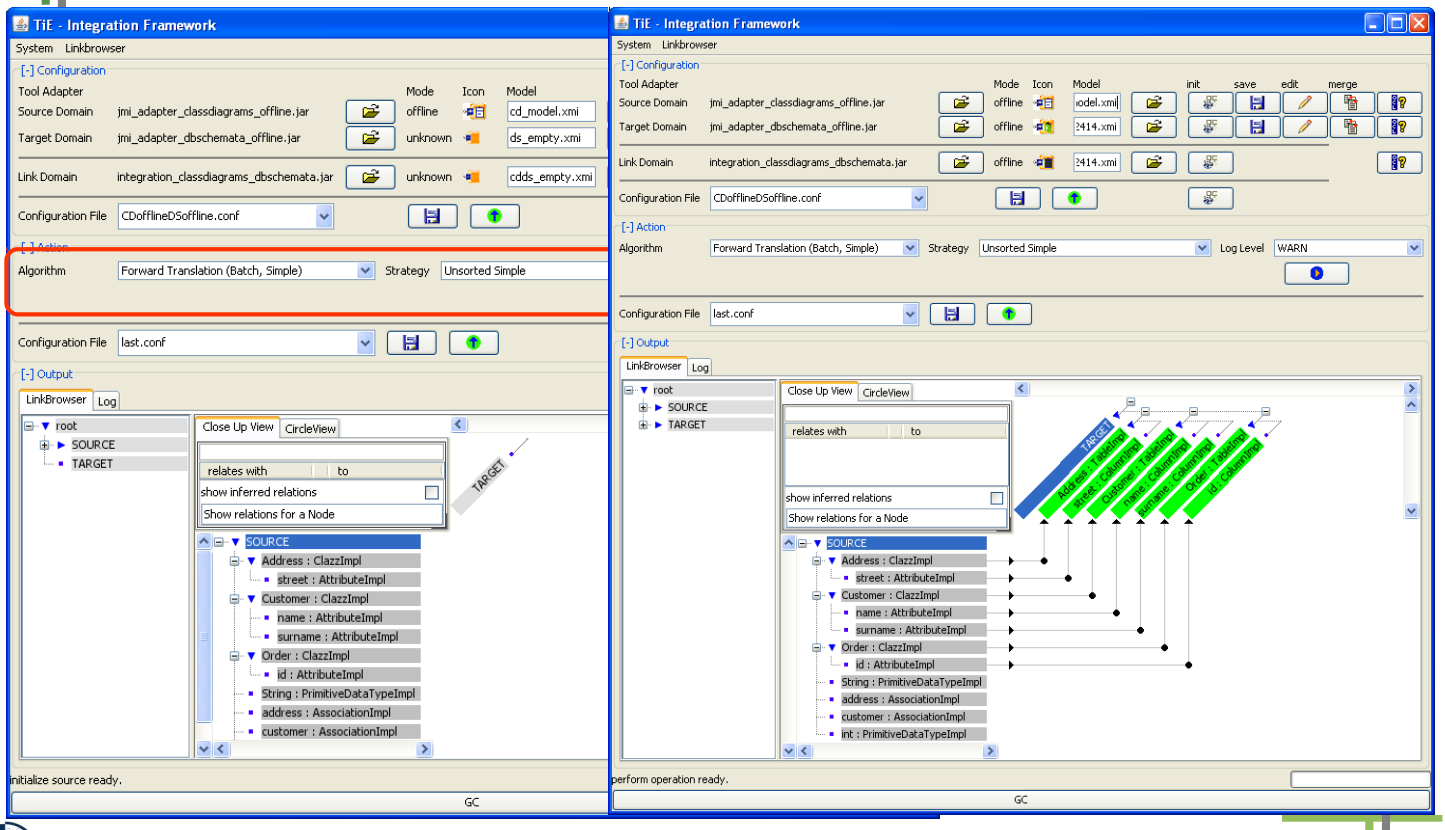


TiE-CDDS – Constraints in Class Diagrams (4)
Integration Framework



translation process may start now…

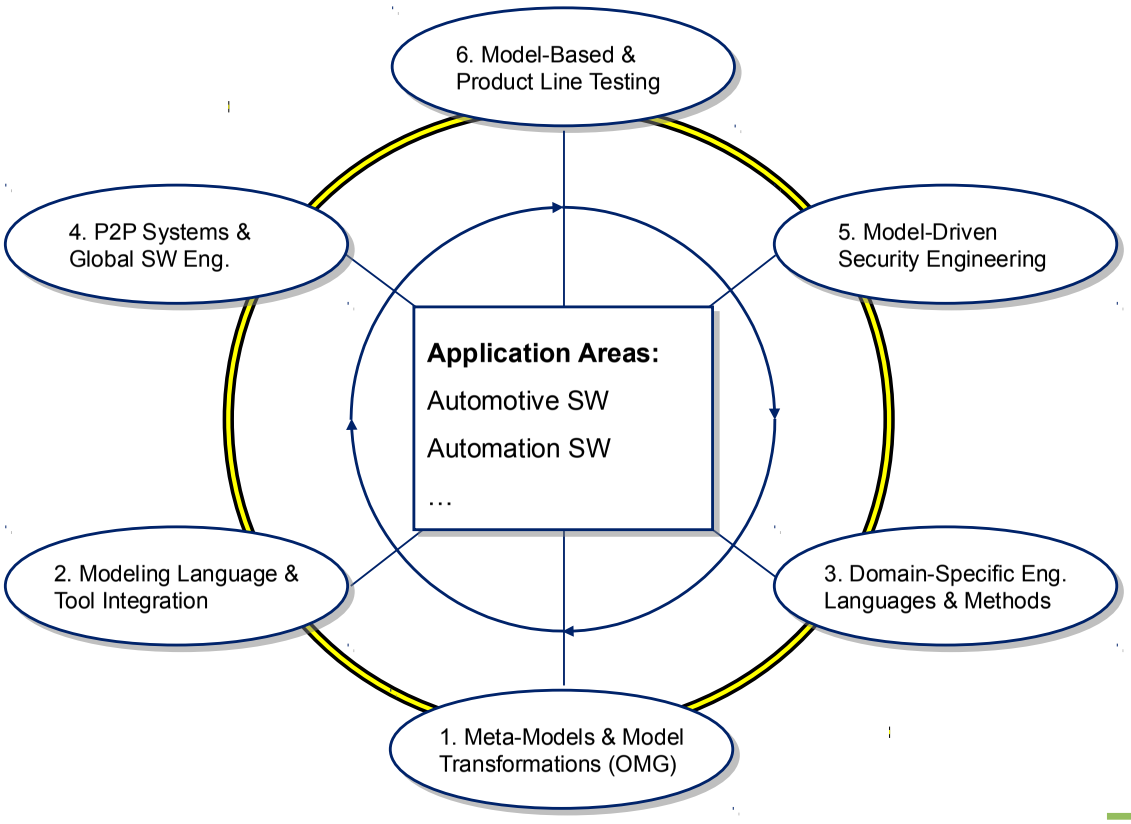
TiE-CDDS – Constraints in Class Diagrams (5)
Forward Translation to DB representation



Future Work – OCL

- 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 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	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

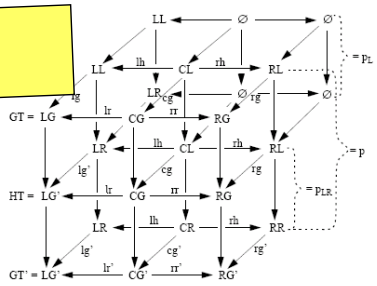
Further reading

- 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.

The End

Some slides are courtesy Florian Heidenreich and Felix Klar

Thank you for your attention…



http://www.moflon.org