25.3.1. The Metamodeling Architecture of MetaCASE Tool MOFLON

TECHNISCHE UNIVERSITÄT DARMSTADT

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

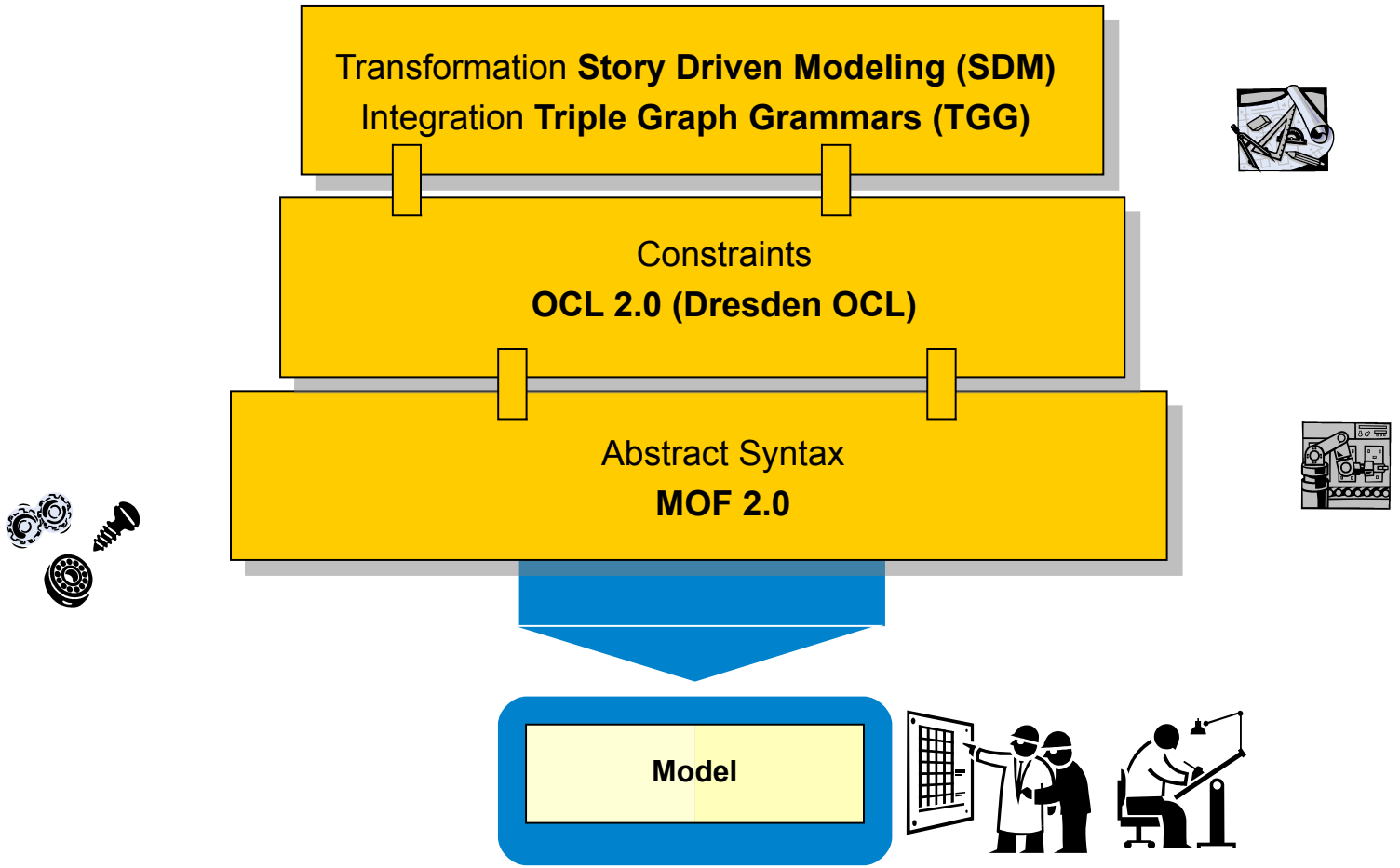
Felix Klar

Felix.Klar@es.tu-darmstadt.de

15.10.2009

Metamodel Architecture of MOFLON

TECHNISCHE UNIVERSITÄT DARMSTADT



MOFLON MetaCASE – Main Features

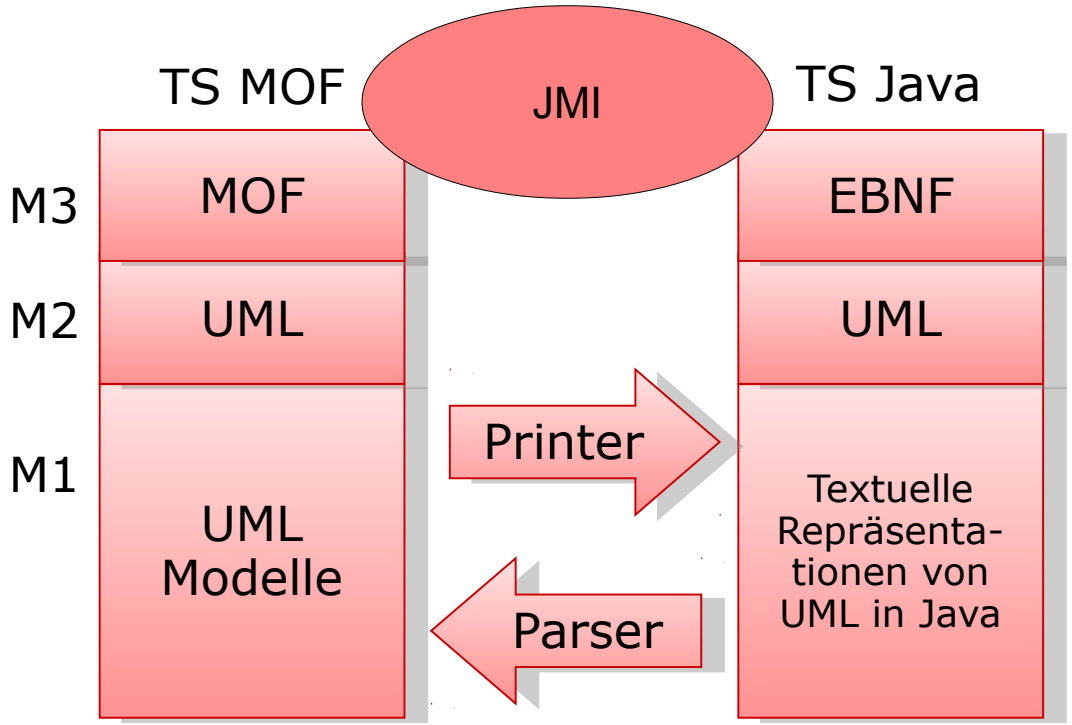
TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

TECHNISCHE UNIVERSITÄT DARMSTADT

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

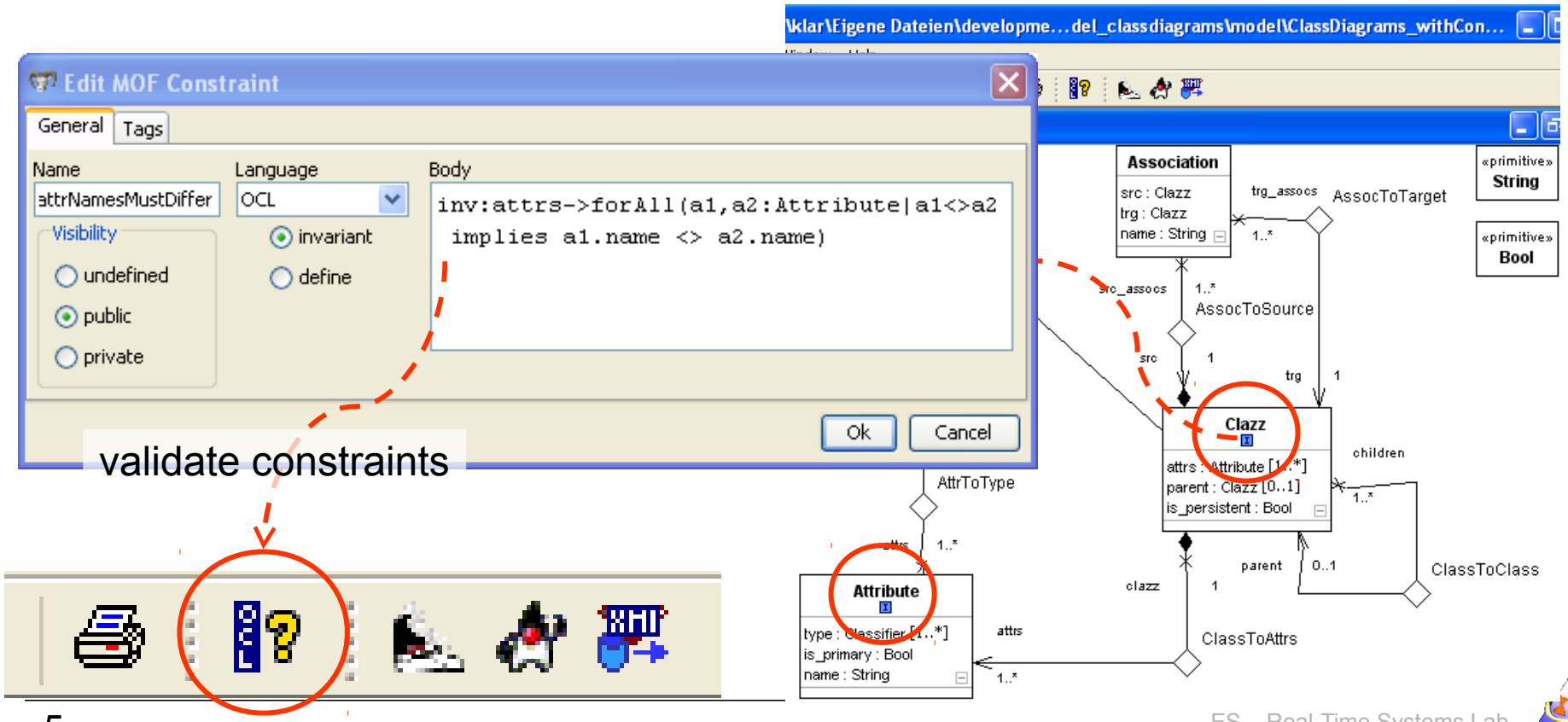


(OCL) Constraints in MOFLON – MOF Editor

TECHNISCHE UNIVERSITÄT DARMSTADT

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

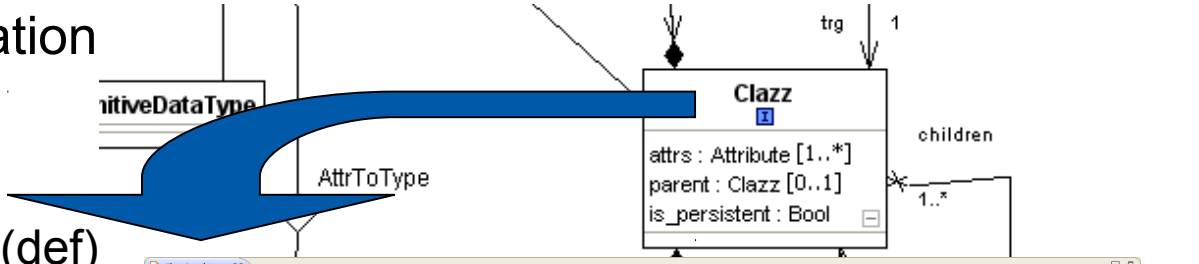
→ MOFLON MOF editor may add constraints to elements

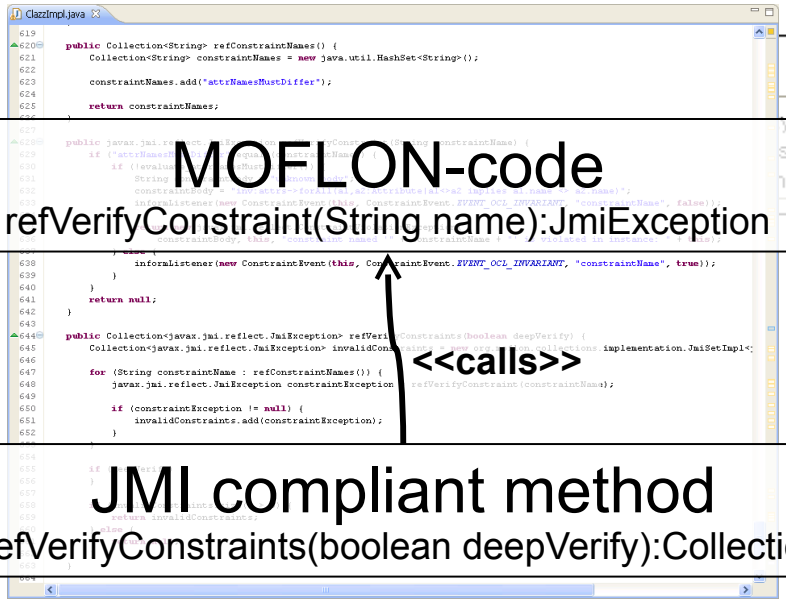


(OCL) Constraints in MOFLON – Generated Implementations

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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)





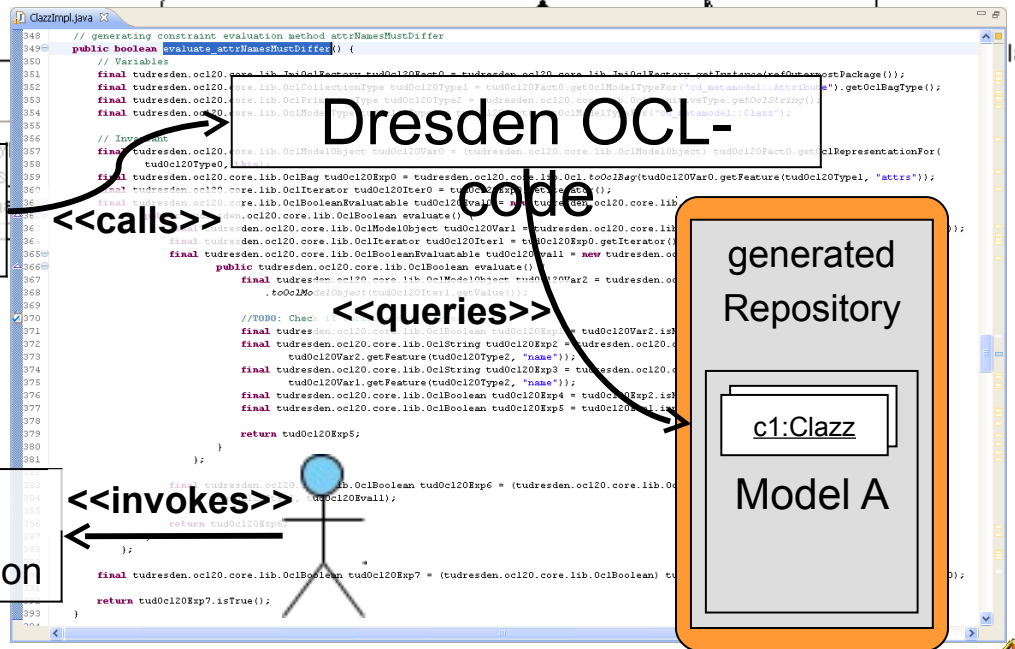
MOFLON-code

refVerifyConstraint(String name):JmiException

<<calls>>

JMI compliant method

refVerifyConstraints(boolean deepVerify):Collection



Dresden OCL-code

<<calls>>

<<queries>>

<<invokes>>

generated Repository

c1:Clazz

Model A

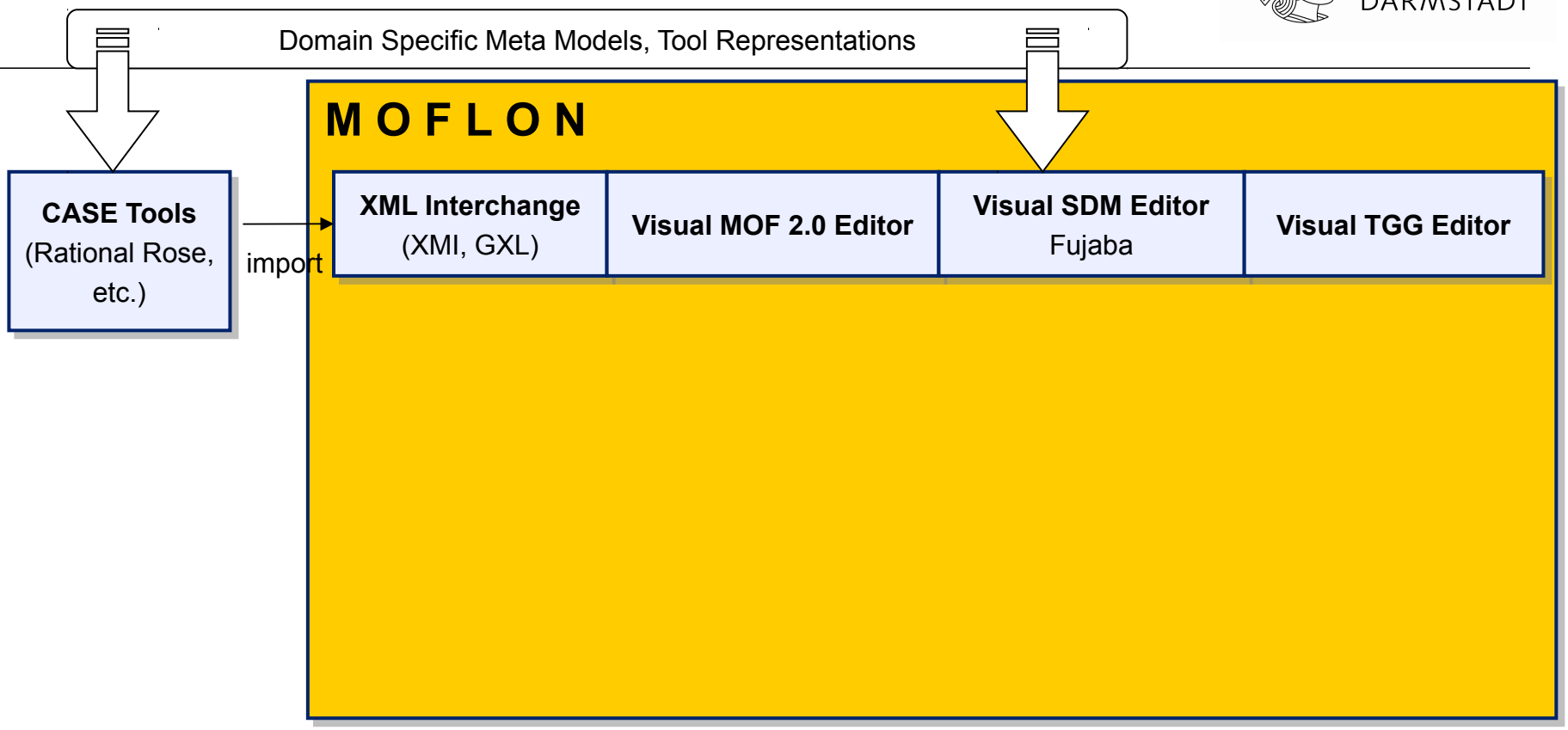
ClazzImpl.java

```
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
```

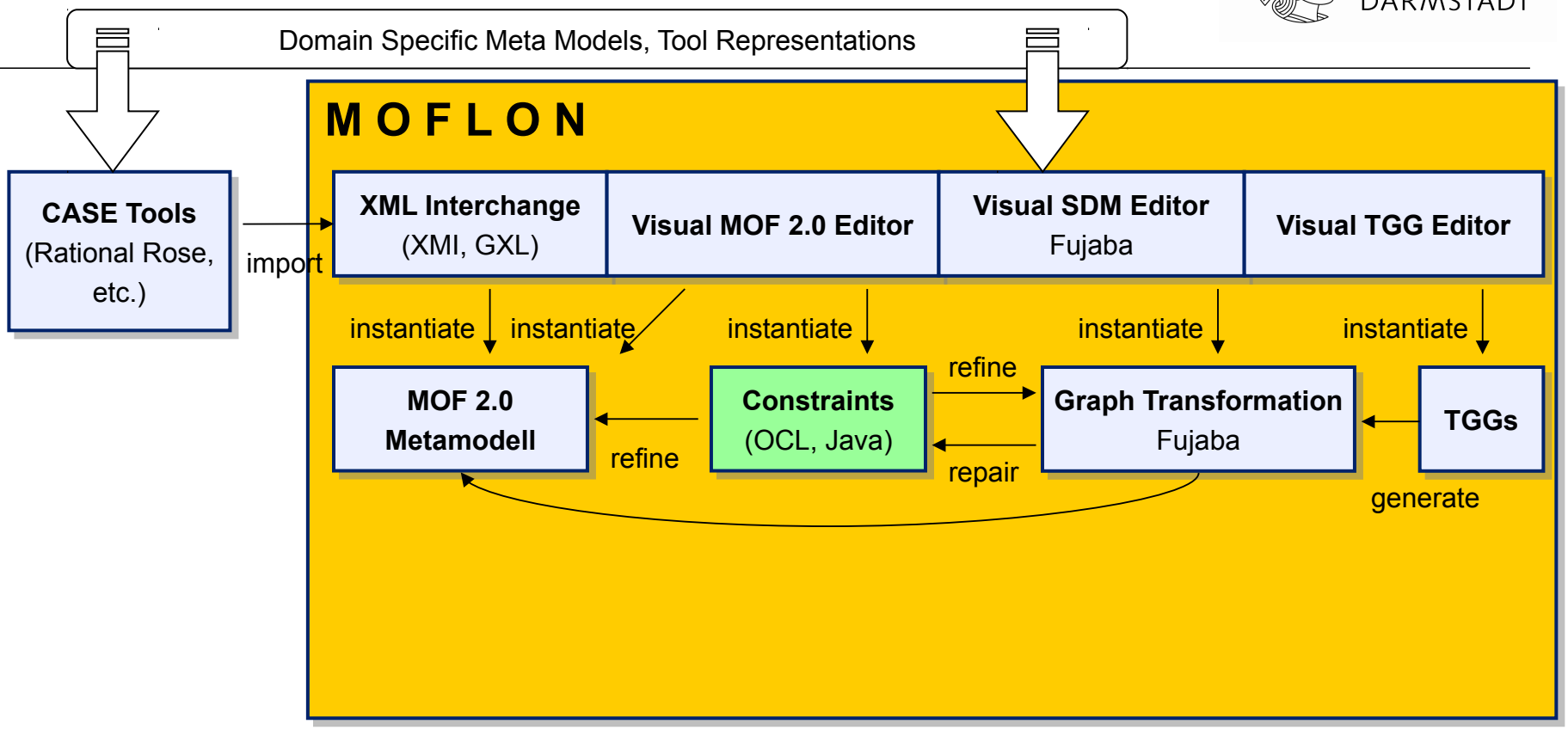

ClazzImpl.java

```
348     // generating constraint evaluation method attrNamesMustDiffer
349     public boolean evaluate_attrNamesMustDiffer() {
350         // Variables
351         final tudresden.ocl20.core.lib.Jmi0clFactory tud0cl20Fact0 = tudresden.ocl20.core.lib.Jmi0clFactory.getInstance(refOutermostPackage());
352         final tudresden.ocl20.core.lib.0clCollectionType tud0cl20Type1 = tud0cl20Fact0.get0clModelTypeFor("cd_metamodel::Attribute").get0clBagType();
353         final tudresden.ocl20.core.lib.0clPrimitiveType tud0cl20Type2 = tudresden.ocl20.core.lib.0clPrimitiveType.get0clString();
354         final tudresden.ocl20.core.lib.0clModelType tud0cl20Type0 = tud0cl20Fact0.get0clModelTypeFor("cd_metamodel::Clazz");
355
356         // Invariant
357         final tudresden.ocl20.core.lib.0clModel0bject tud0cl20Var0 = (tudresden.ocl20.core.lib.0clModel0bject) tud0cl20Fact0.get0clRepresentationFor(
358                 tud0cl20Type0, this);
359         final tudresden.ocl20.core.lib.0clBag tud0cl20Exp0 = tudresden.ocl20.core.lib.0cl.to0clBag(tud0cl20Var0.getFeature(tud0cl20Type1, "attrs"));
360         final tudresden.ocl20.core.lib.0clIterator tud0cl20Iter0 = tud0cl20Exp0.getIterator();
361         final tudresden.ocl20.core.lib.0clBooleanEvaluatable tud0cl20Eval0 = new tudresden.ocl20.core.lib.0clBooleanEvaluatable() {
362             public tudresden.ocl20.core.lib.0clBoolean evaluate() {
363                 final tudresden.ocl20.core.lib.0clModel0bject tud0cl20Var1 = tudresden.ocl20.core.lib.0cl.to0clModel0bject(tud0cl20Iter0.getValue());
364                 final tudresden.ocl20.core.lib.0clIterator tud0cl20Iter1 = tud0cl20Exp0.getIterator();
365                 final tudresden.ocl20.core.lib.0clBooleanEvaluatable tud0cl20Eval1 = new tudresden.ocl20.core.lib.0clBooleanEvaluatable() {
366                     public tudresden.ocl20.core.lib.0clBoolean evaluate() {
367                         final tudresden.ocl20.core.lib.0clModel0bject tud0cl20Var2 = tudresden.ocl20.core.lib.0cl
368                             .to0clModel0bject(tud0cl20Iter1.getValue());
369
370                         //TODO: Check if VariableId is correct
371                         final tudresden.ocl20.core.lib.0clBoolean tud0cl20Exp1 = tud0cl20Var2.isNotEqualTo(tud0cl20Var1);
372                         final tudresden.ocl20.core.lib.0clString tud0cl20Exp2 = tudresden.ocl20.core.lib.0cl.to0clString(
373                                 tud0cl20Var2.getFeature(tud0cl20Type2, "name"));
374                         final tudresden.ocl20.core.lib.0clString tud0cl20Exp3 = tudresden.ocl20.core.lib.0cl.to0clString(
375                                 tud0cl20Var1.getFeature(tud0cl20Type2, "name"));
376                         final tudresden.ocl20.core.lib.0clBoolean tud0cl20Exp4 = tud0cl20Exp2.isNotEqualTo(tud0cl20Exp3);
377                         final tudresden.ocl20.core.lib.0clBoolean tud0cl20Exp5 = tud0cl20Exp1.implies(tud0cl20Exp4);
378
379                         return tud0cl20Exp5;
380                     }
381                 };
382
383                 final tudresden.ocl20.core.lib.0clBoolean tud0cl20Exp6 = (tudresden.ocl20.core.lib.0clBoolean) tud0cl20Exp0.forAll(
384                         tud0cl20Iter1, tud0cl20Eval1);
385
386                 return tud0cl20Exp6;
387             }
388         };
389
390         final tudresden.ocl20.core.lib.0clBoolean tud0cl20Exp7 = (tudresden.ocl20.core.lib.0clBoolean) tud0cl20Exp0.forAll(tud0cl20Iter0, tud0cl20Eval0);
391
392         return tud0cl20Exp7.isTrue();
393     }
```

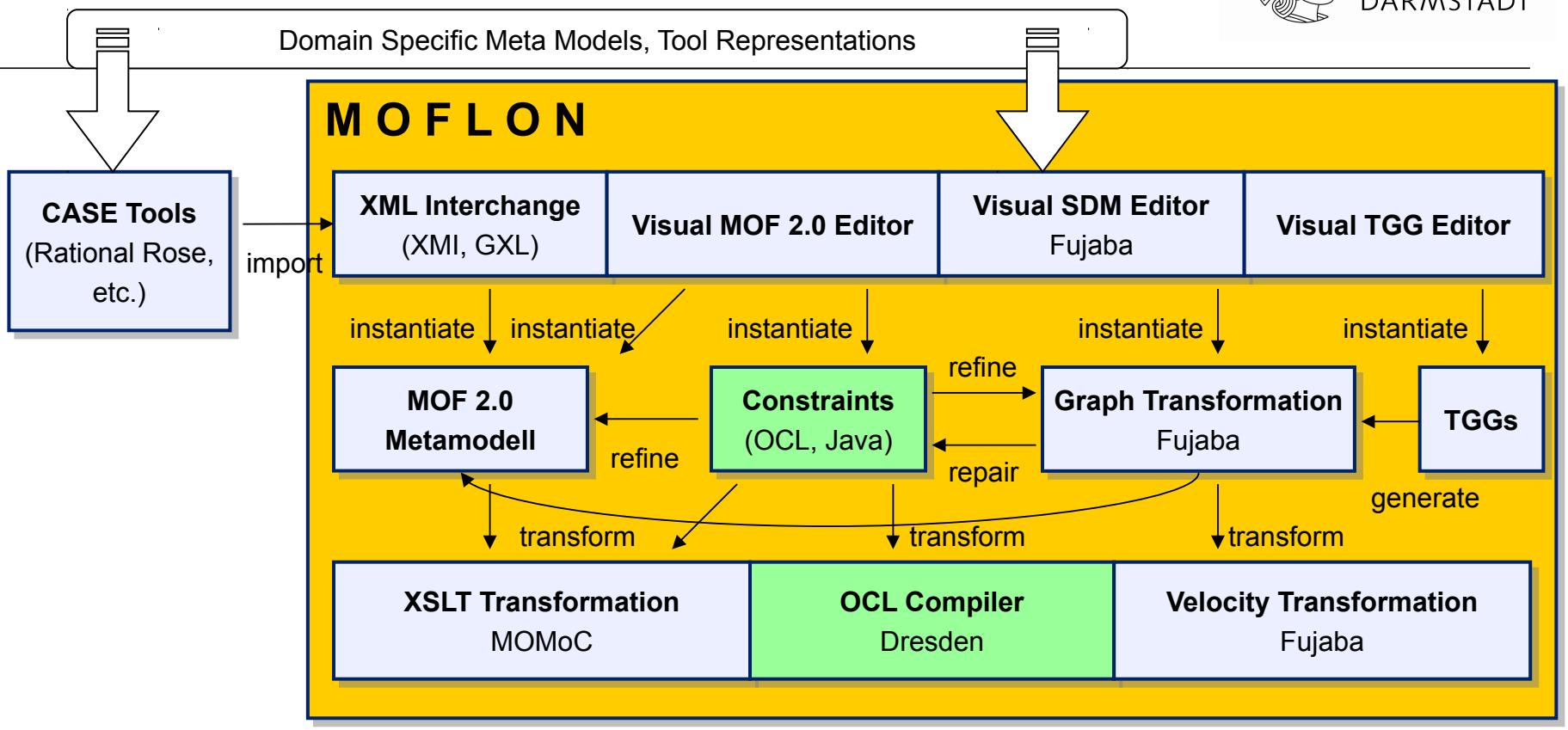
MOFLON – Architecture



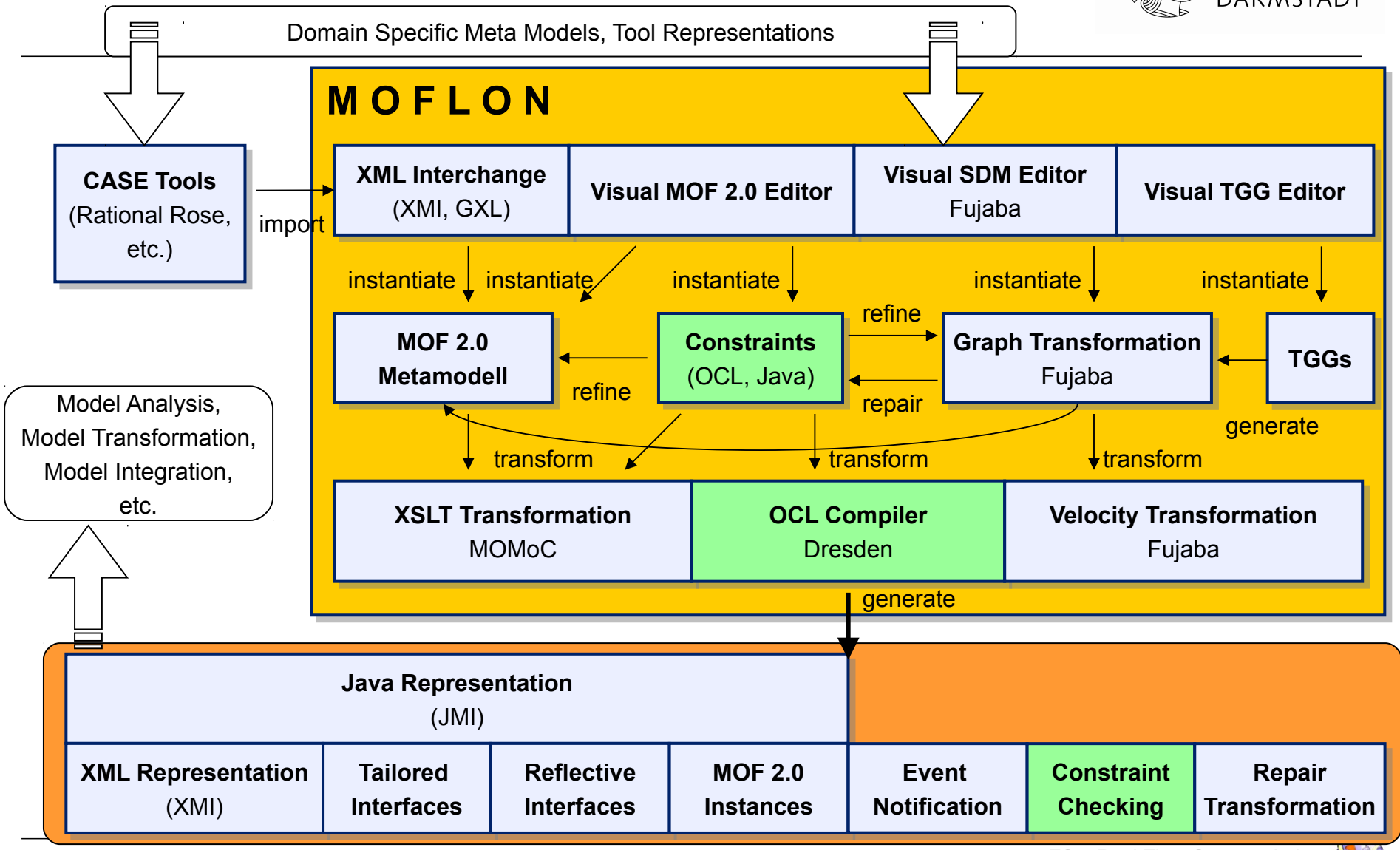
MOFLON – Architecture



MOFLON – Architecture

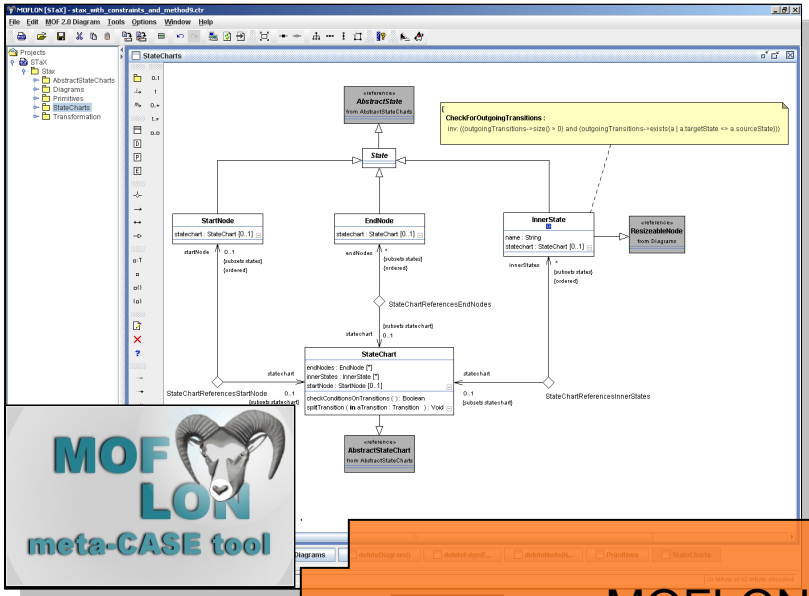


MOFLON – Architecture

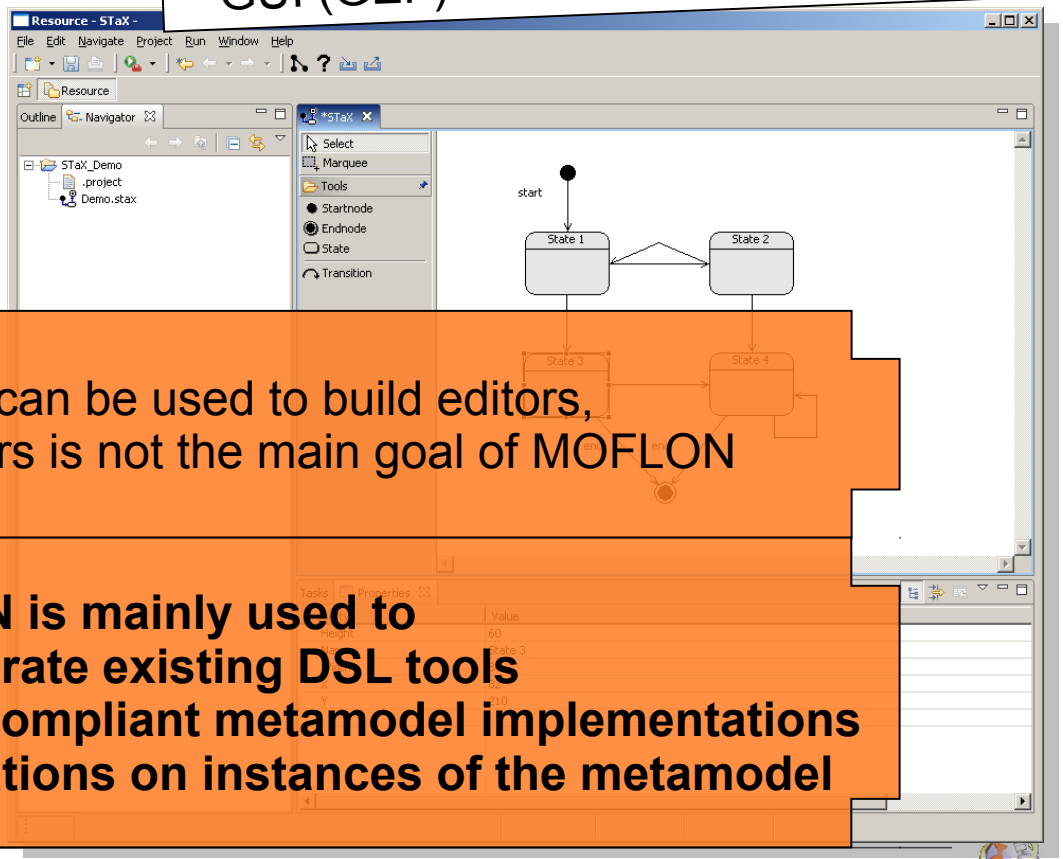


26.2 MOFLON Case Study – Statechart Editor (STaX)

TECHNISCHE UNIVERSITÄT DARMSTADT



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



MOF LON meta-CASE tool

+

eclipse 3.1

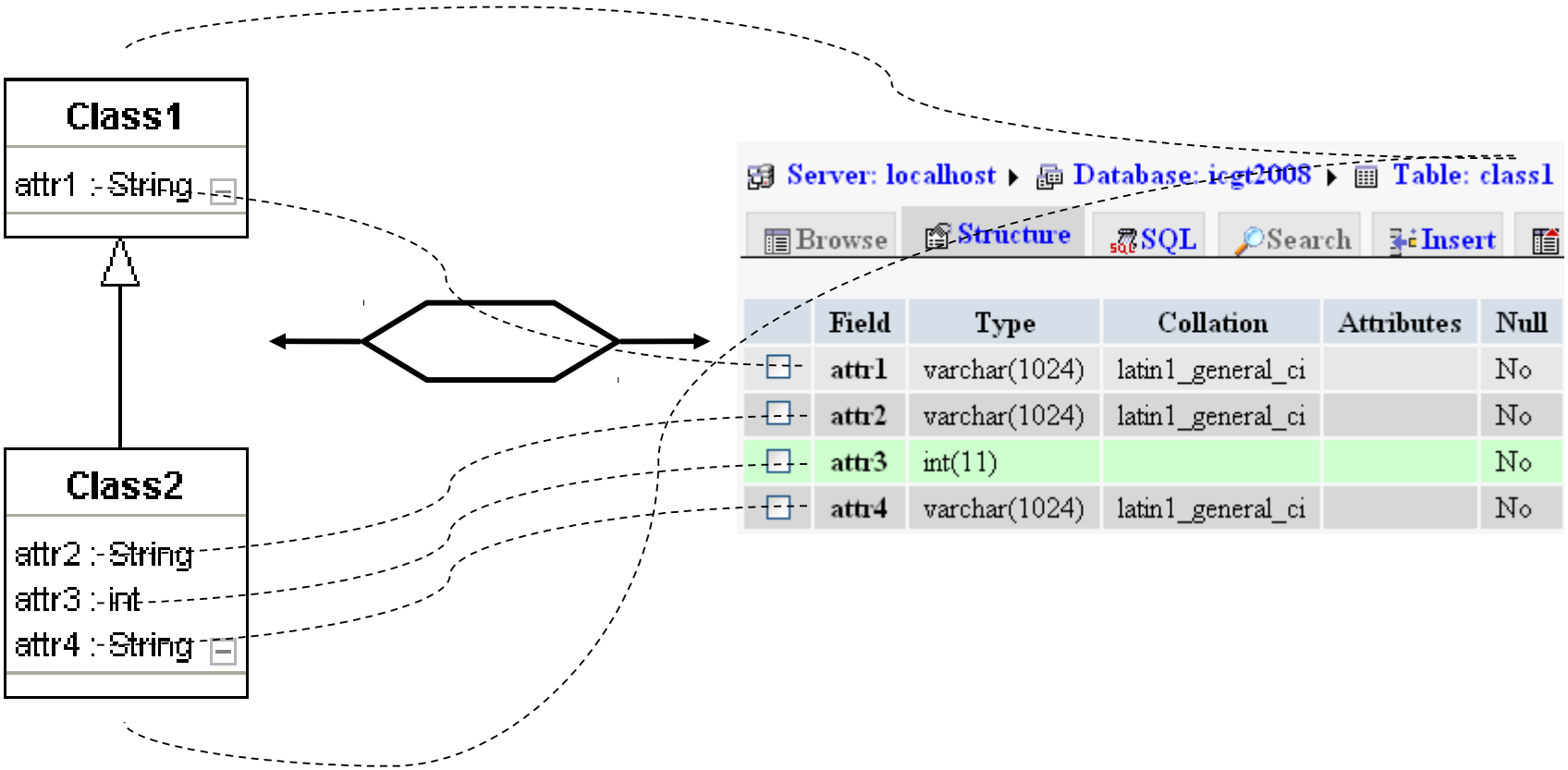
GEF

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

TECHNISCHE UNIVERSITÄT DARMSTADT



domain specific language, e.g. Class Diagrams

domain specific language, e.g. Database Schemata

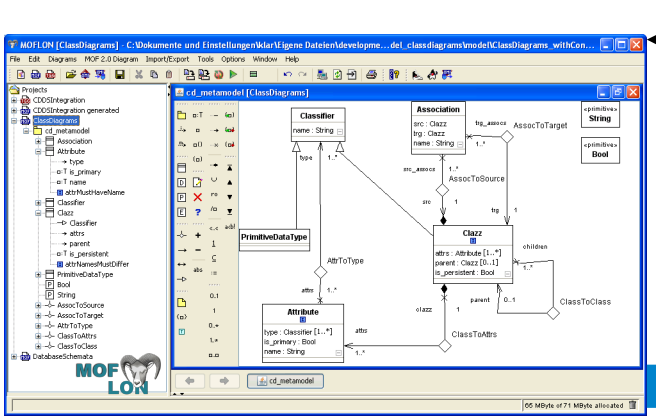
Case Study 2: Tool Integration Scenario TiECDDS: (ClassD / DatabaseSchema)

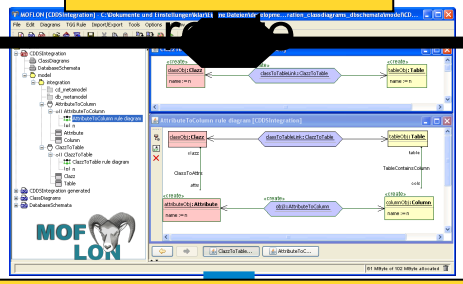
TECHNISCHE UNIVERSITÄT DARMSTADT

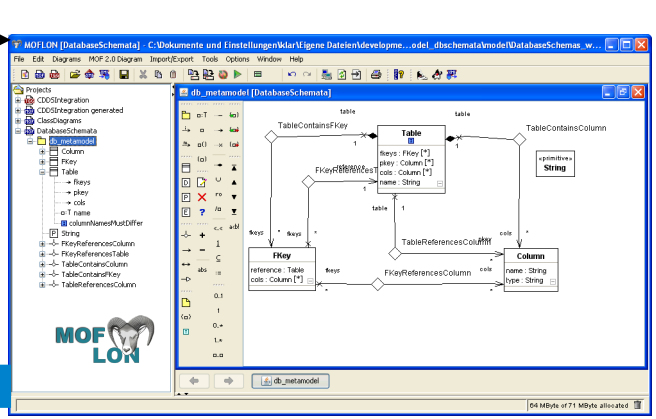
Class Diagrams Metamodel

TGGs

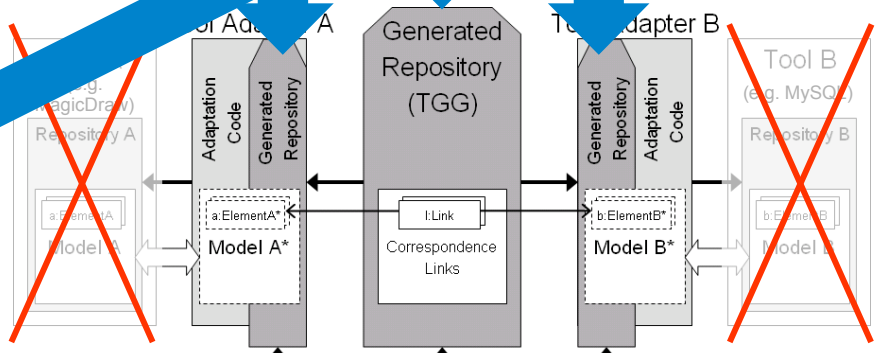
Database Schemata Metamodel







MOFLON generates

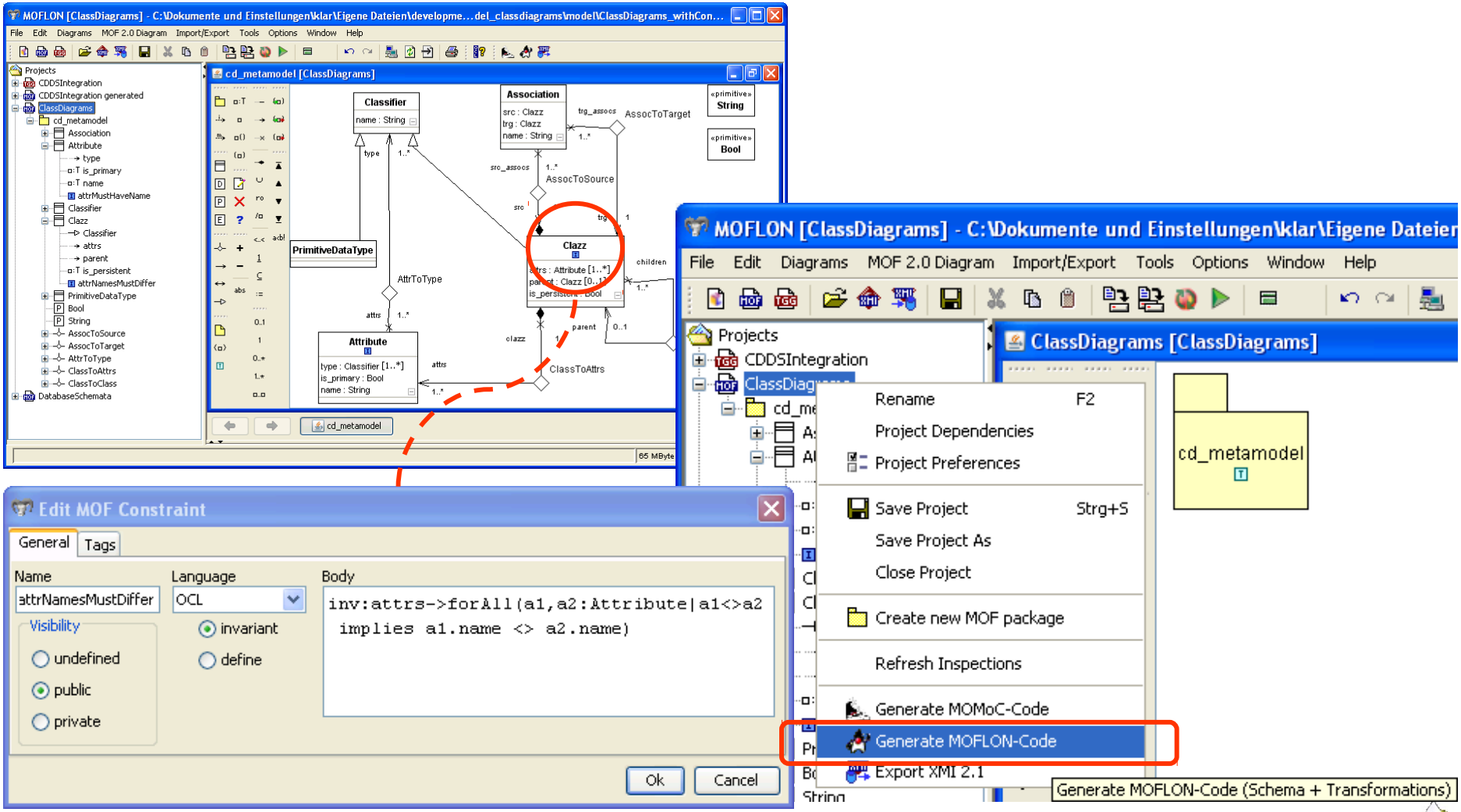


integration rule code

Run-Time Verification of Constraints

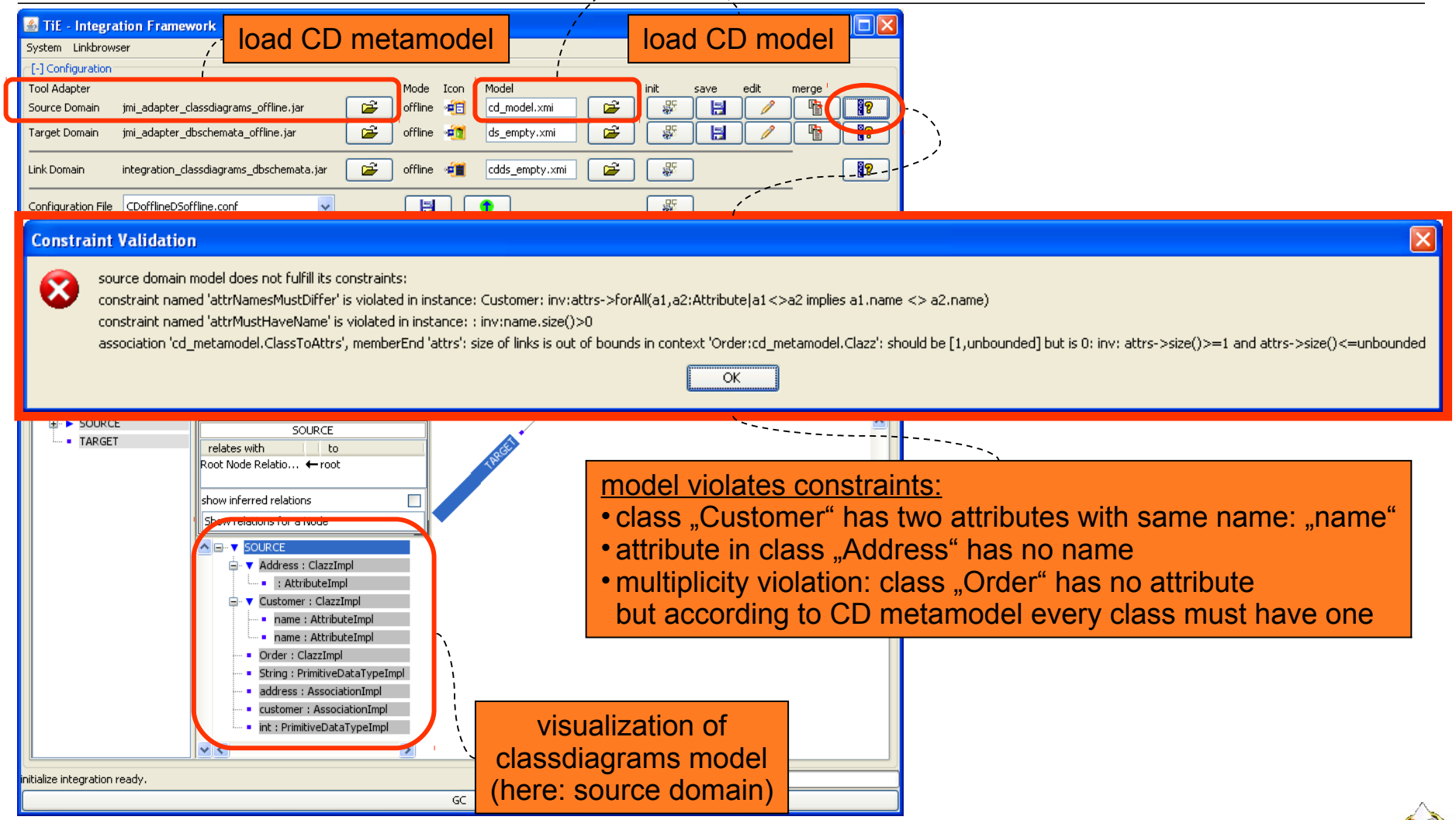
TiE-CDDS – Focus on Constraints in CD (1)
Generate Code from MOF model (CD metamodel)

TECHNISCHE UNIVERSITÄT DARMSTADT



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

TECHNISCHE UNIVERSITÄT DARMSTADT

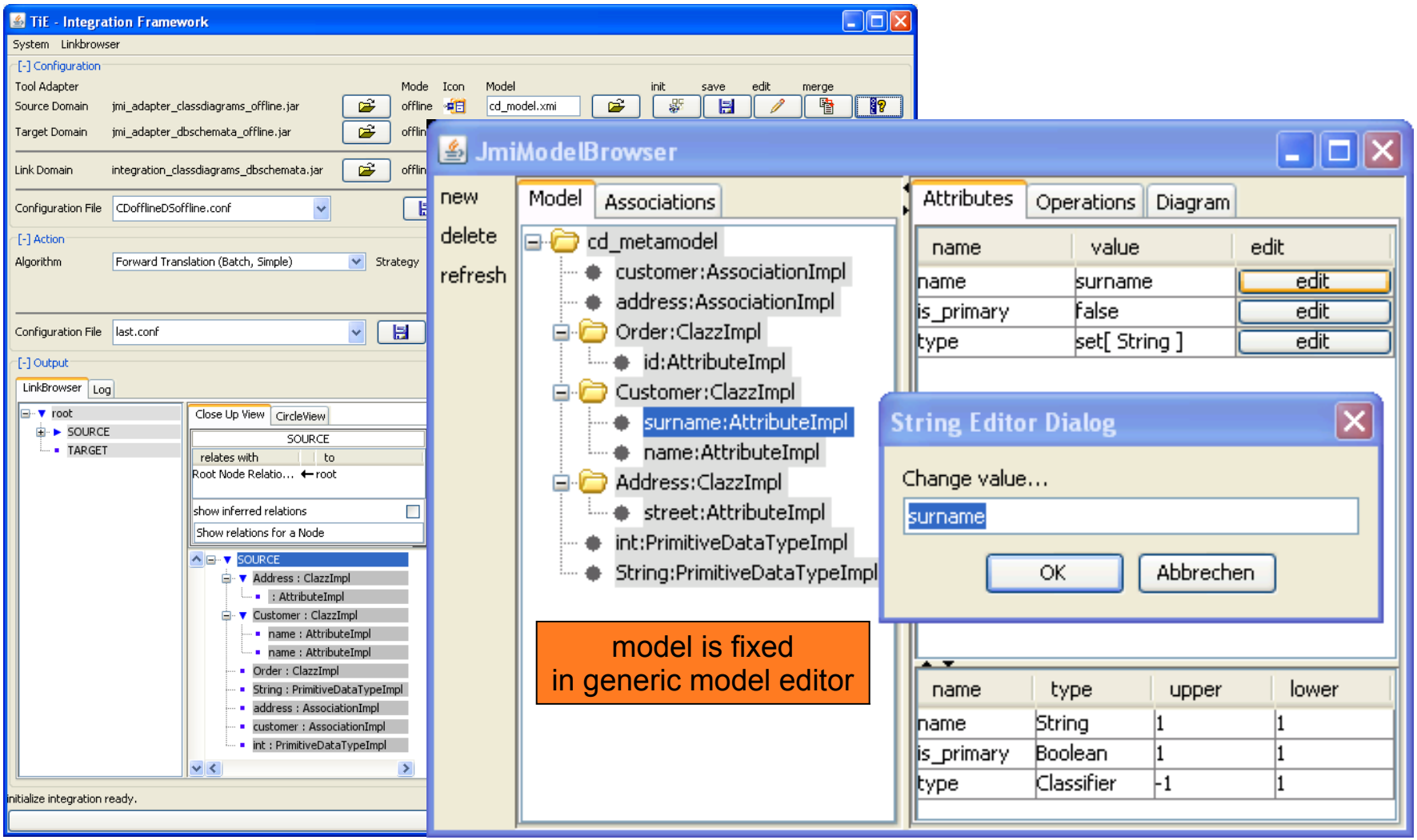


model violates constraints:
- class „Customer“ has two attributes with same name: „name“
- attribute in class „Address“ has no name
- multiplicity violation: class „Order“ has no attribute but according to CD metamodel every class must have one

visualization of classdiagrams model (here: source domain)

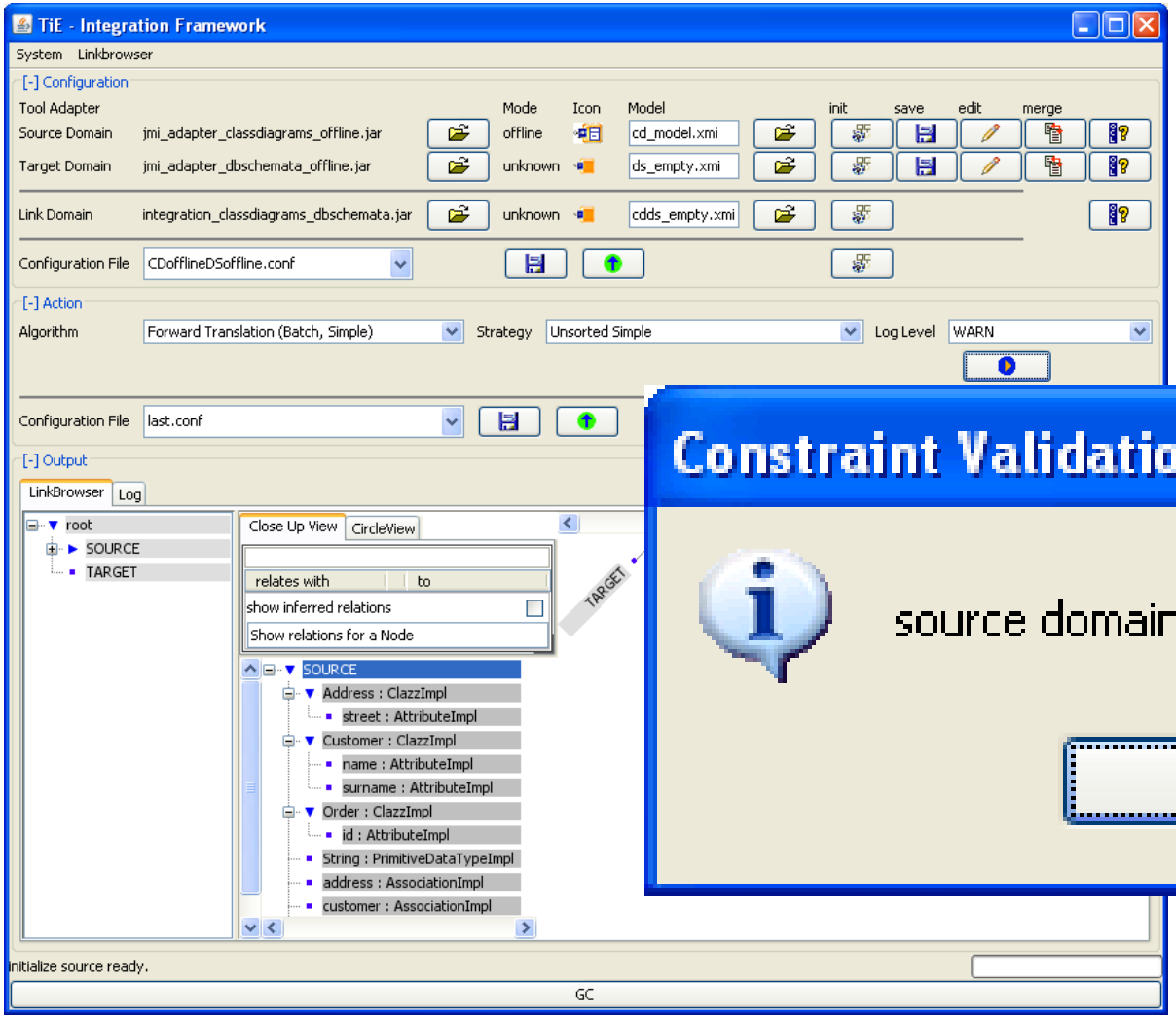
TiE-CDDS – Focus on Constraints in CD (3)
Model Browser

TECHNISCHE UNIVERSITÄT DARMSTADT

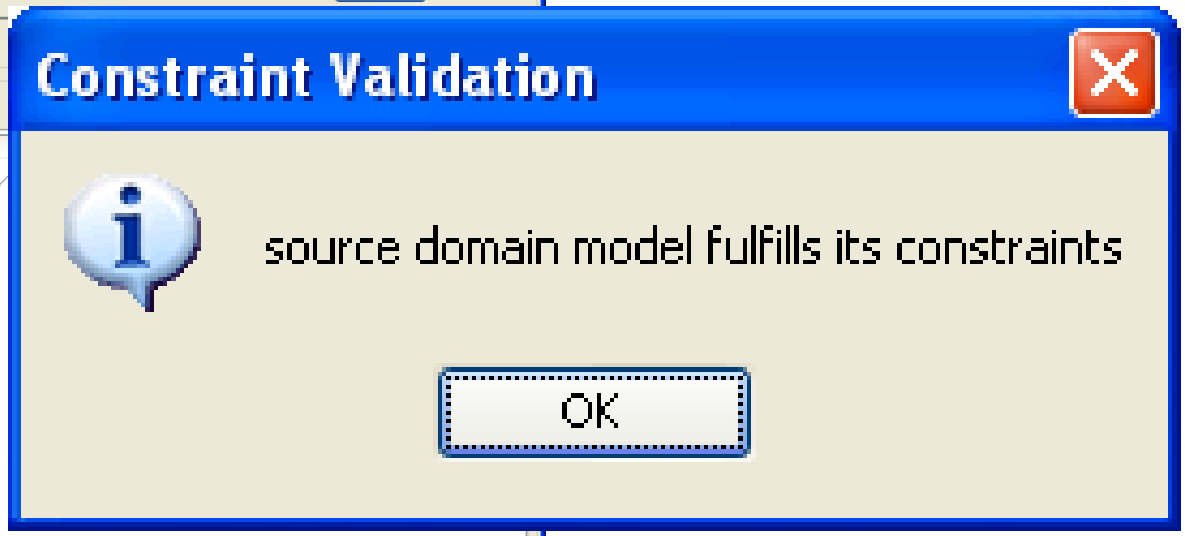


TiE-CDDS – Focus on Constraints in CD (4)
Integration Framework

TECHNISCHE UNIVERSITÄT DARMSTADT

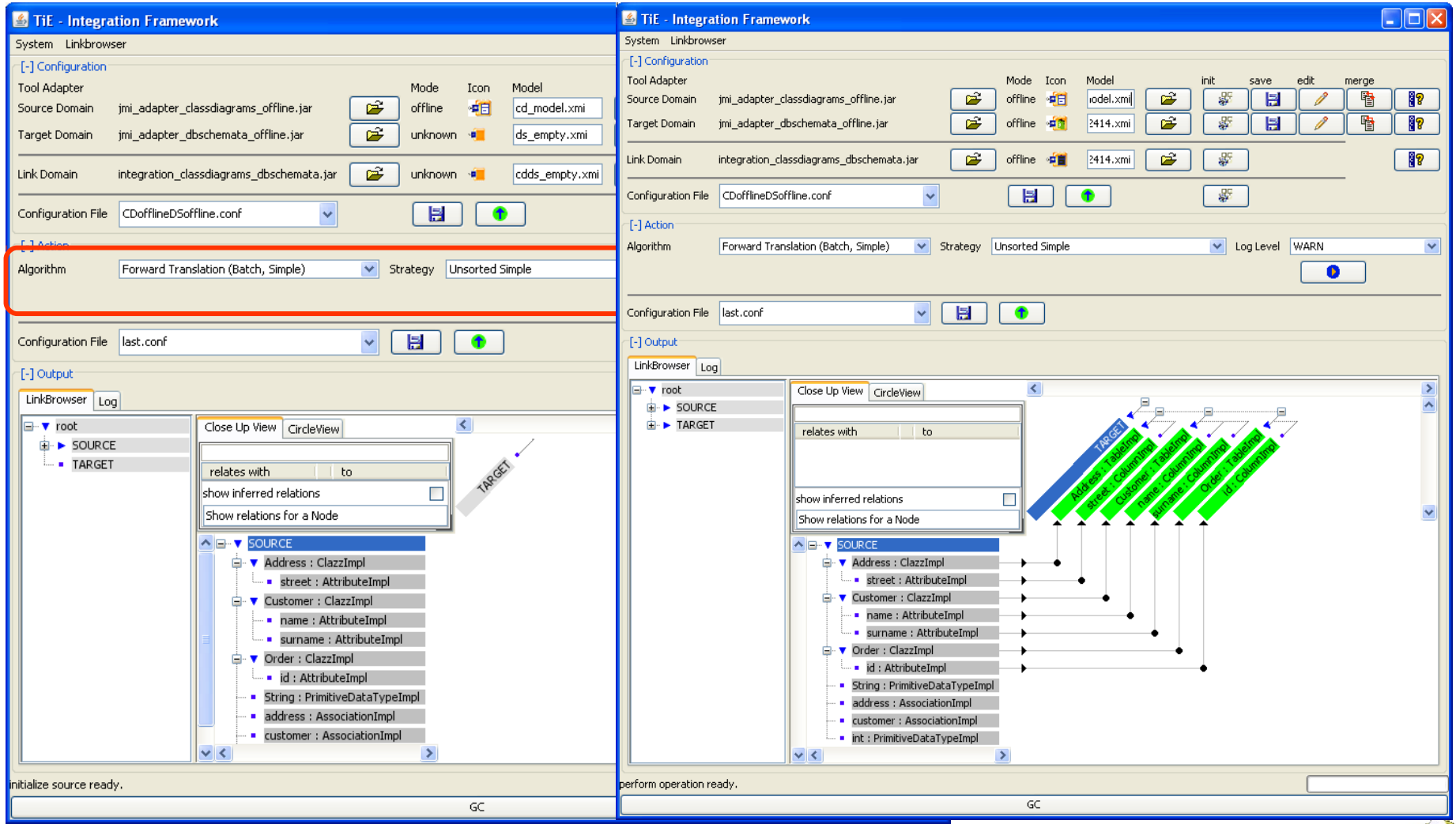


translation process may start now…



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

TECHNISCHE UNIVERSITÄT DARMSTADT



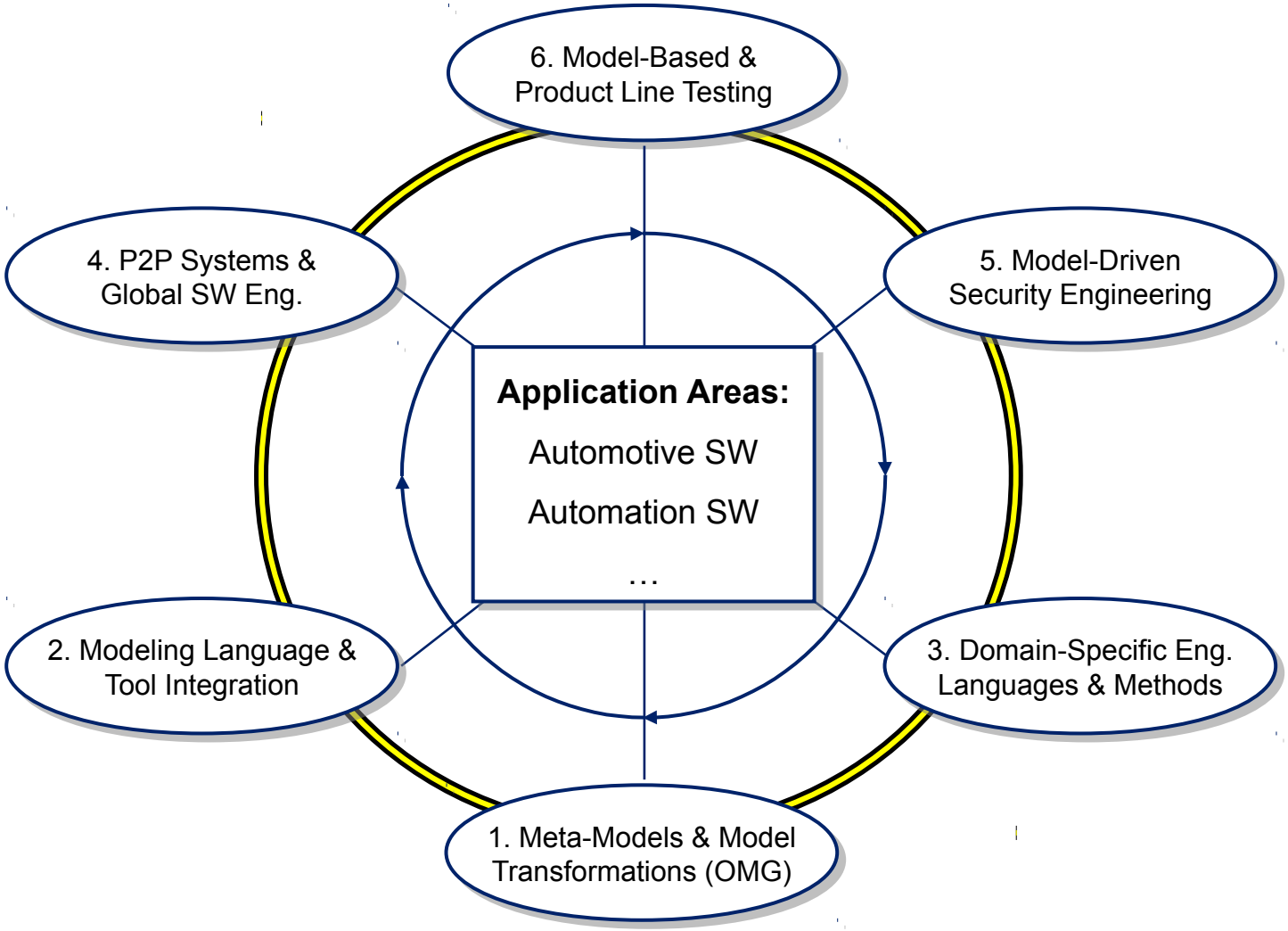
Future Work – OCL

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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)

TECHNISCHE UNIVERSITÄT DARMSTADT



Related Approaches

TECHNISCHE UNIVERSITÄT DARMSTADT

	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

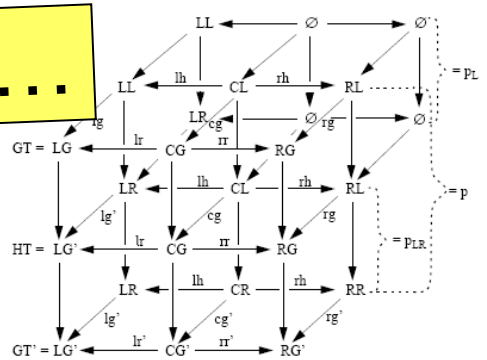
TECHNISCHE UNIVERSITÄT DARMSTADT

- **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

TECHNISCHE UNIVERSITÄT DARMSTADT

Thank you for your attention…



MOFLON

http://www.moflon.org