

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

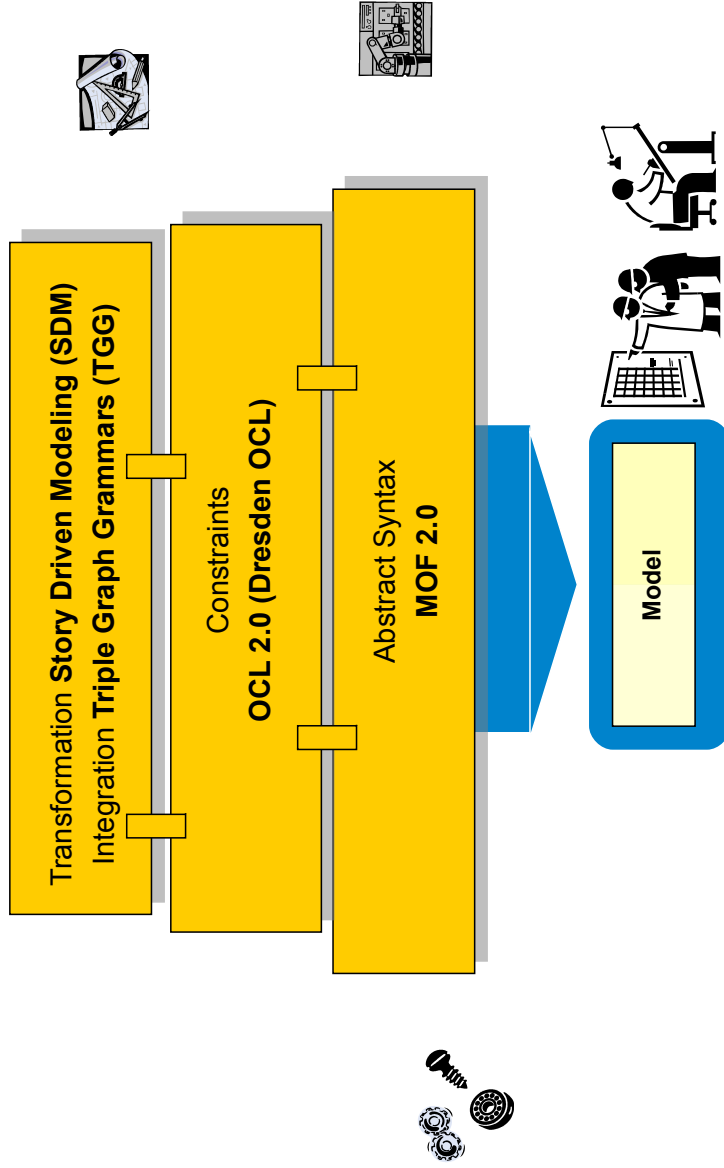
Felix Klar

Felix.Klar@es.tu-darmstadt.de

15.10.2009

© author(s) of these slides 2009 including research results of the research network ES and TU Darmstadt otherwise as specified at the respective slide

Metamodel Architecture of MOFLON

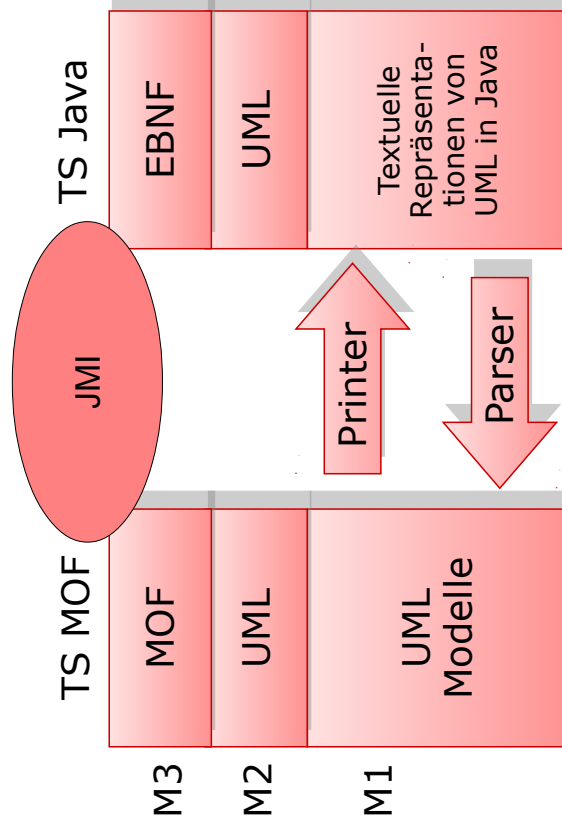


- MOF2.0 editor (draw metamodels that comply to MOF2.0 standard)
 - build Domain Specific Languages (DSLs)
 - based on the CASE-tool framework Fujaba
 - possibility to extend MOFLON by own plugins
 - interoperability (import / export)
- transform metamodel instances with model transformations (SDM, TGG)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



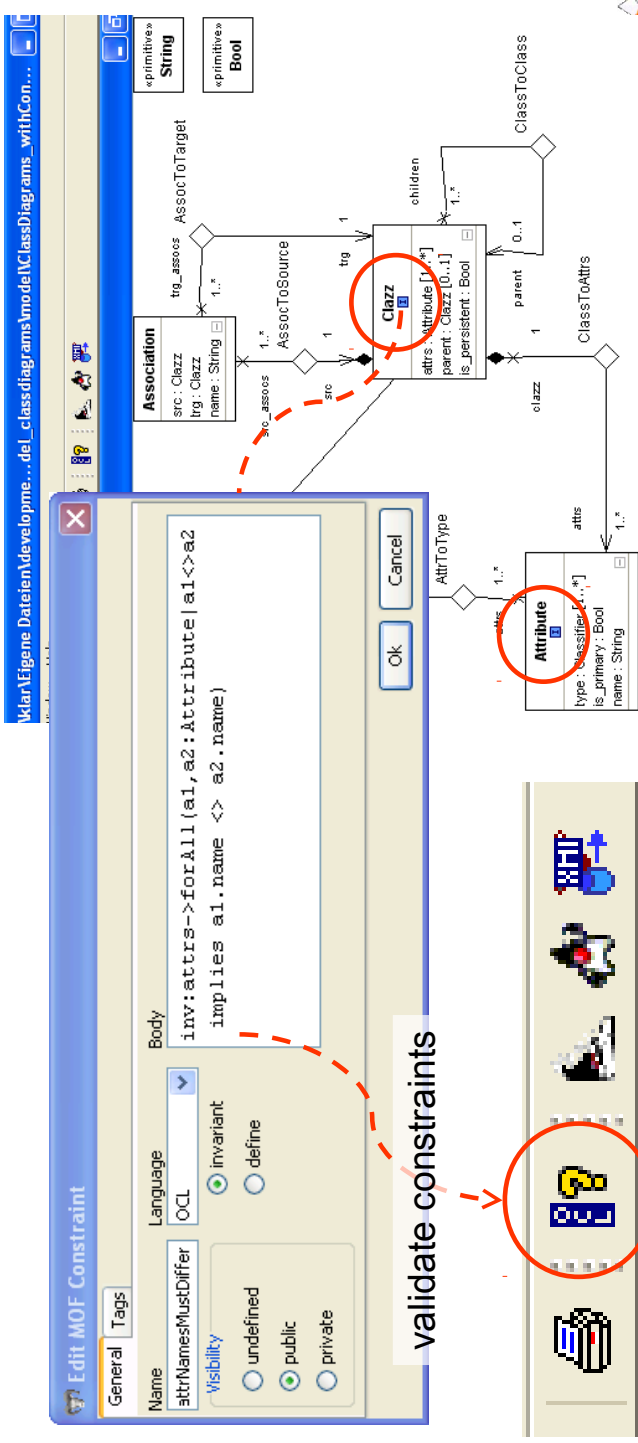
Einschub: JMI: Transformative TS-Bücke für MOF und Java, Sprache UML

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

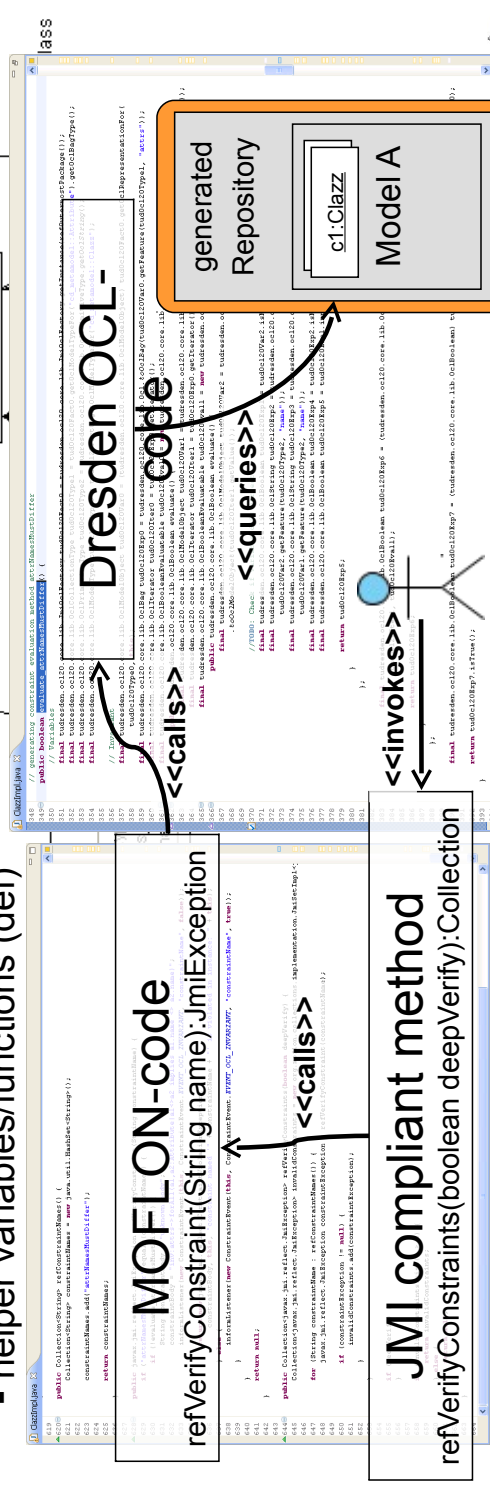


The screenshot shows the 'Edit MOF Constraint' dialog box. The 'Name' field contains 'attrNamesMustDiffer'. The 'Language' is set to 'OCL'. The 'Body' contains the constraint: `inv: attr->forall(a1, a2: Attribute | a1 <> a2 implies a1.name <> a2.name)`. The 'Visibility' is set to 'public'. Below the dialog, a toolbar contains icons for 'validate constraints' (a question mark in a box), 'copy', 'paste', 'undo', and 'redo'. The background shows a metamodel diagram with classes like 'Clazz', 'AssocToSource', and 'AssocToTarget'. A red circle highlights the 'Clazz' class in the diagram, and another red circle highlights the 'validate constraints' icon in the toolbar.

5 | 15.10.2009 | Dresden OCL2 in MOFLON

(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)



The screenshot shows generated Java code for a repository. A blue arrow points from the MOFLON MOF Editor to the code. The code includes a 'Dresden OCL' section with OCL constraints converted to Java. A box labeled 'generated Repository' contains a 'Model A' class with a 'Clazz' attribute. The code includes sections for 'MOFLON-code' (a 'refVerifyConstraint' method), 'JMI compliant method' (a 'refVerifyConstraints' method), and 'Dresden OCL' (the actual constraint logic). A red circle highlights the 'validate constraints' icon from the MOFLON editor, and another red circle highlights the 'validate constraints' icon in the code.

6 | 15.10.2009 | Dresden OCL2 in MOFLON

```

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

```



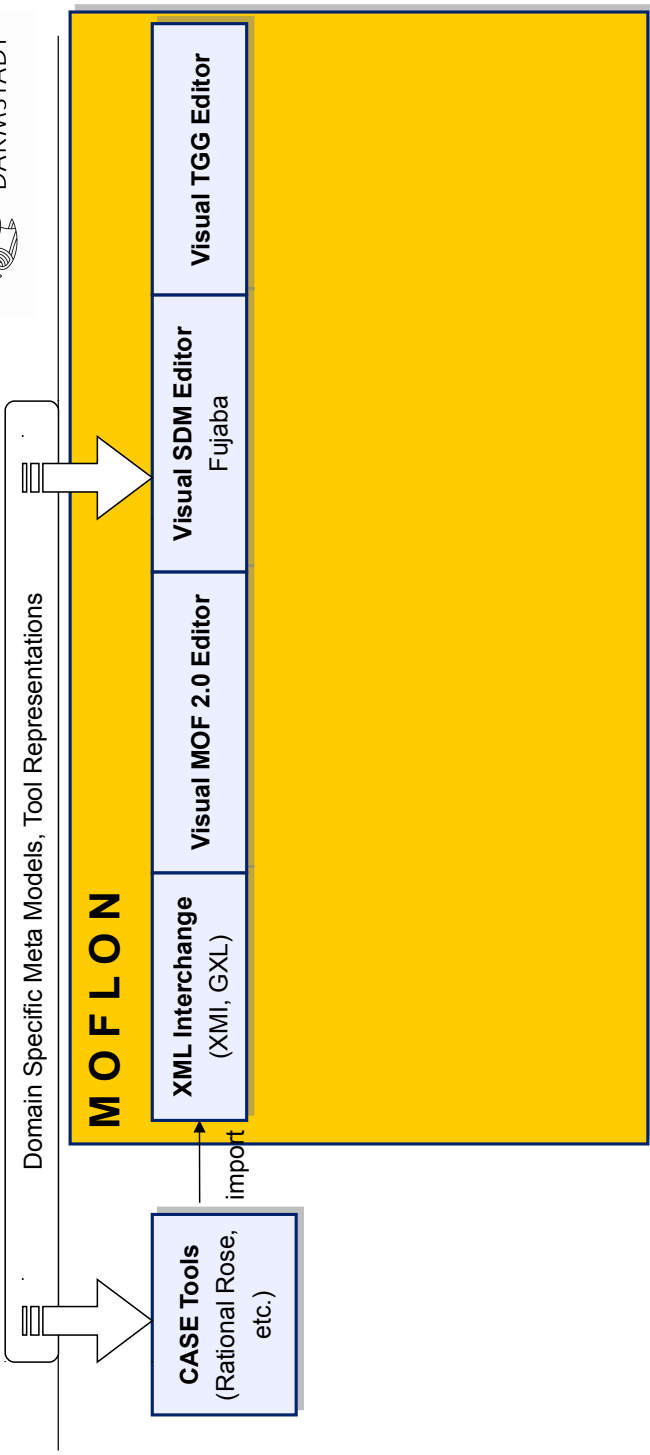
```

ClazImpl.java
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");
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.tOclBag(tudOcl20Var0.getFeatures(tudOcl20Type1, "attrs"));
360     final tudresden.oc120.core.lib.OclIterator tudOcl20Iter0 = tudOcl20Exp0.getIterator();
361     final tudresden.oc120.core.lib.OclBooleanEvaluable tudOcl20Eval0 = new tudresden.oc120.core.lib.OclBooleanEvaluable() {
362         public tudresden.oc120.core.lib.OclBoolean evaluate() {
363             final tudresden.oc120.core.lib.OclModelObject tudOcl20Var1 = tudresden.oc120.core.lib.Ocl.tOclModelObject(tudOcl20Iter0.getValue());
364             final tudresden.oc120.core.lib.OclIterator tudOcl20Iter1 = tudOcl20Exp0.getIterator();
365             final tudresden.oc120.core.lib.OclBooleanEvaluable tudOcl20Eval1 = new tudresden.oc120.core.lib.OclBooleanEvaluable() {
366                 public tudresden.oc120.core.lib.OclBoolean evaluate() {
367                     final tudresden.oc120.core.lib.OclModelObject tudOcl20Var2 = tudresden.oc120.core.lib.Ocl
368                         .tOclModelObject(tudOcl20Iter1.getValue());
369                     //TODO: Check if VariableId is correct
370                     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp1 = tudOcl20Var2.isNotEqualTo(tudOcl20Var1);
371                     final tudresden.oc120.core.lib.OclString tudOcl20Exp2 = tudresden.oc120.core.lib.Ocl.tOclString(
372                         tudOcl20Var2.getFeature(tudOcl20Type2, "name"));
373                     final tudOcl20Var1.tudOclString tudOcl20Exp3 = tudresden.oc120.core.lib.Ocl.tOclString(
374                         tudOcl20Var1.getFeature(tudOcl20Type2, "name"));
375                     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp4 = tudOcl20Exp2.isNotEqualTo(tudOcl20Exp3);
376                     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp5 = tudOcl20Exp1.implies(tudOcl20Exp4);
377                     return tudOcl20Exp5;
378                 }
379             };
380         }
381     };
382     final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp6 = (tudresden.oc120.core.lib.OclBoolean) tudOcl20Exp0.forAll(
383         tudOcl20Iter1, tudOcl20Eval1);
384     return tudOcl20Exp6;
385 }
386 }
387 }
388 }
389 }
390 final tudresden.oc120.core.lib.OclBoolean tudOcl20Exp7 = (tudresden.oc120.core.lib.OclBoolean) tudOcl20Exp0.forAll(tudOcl20Iter0, tudOcl20Eval0);
391 return tudOcl20Exp7.isTrue();
392 }
393 }

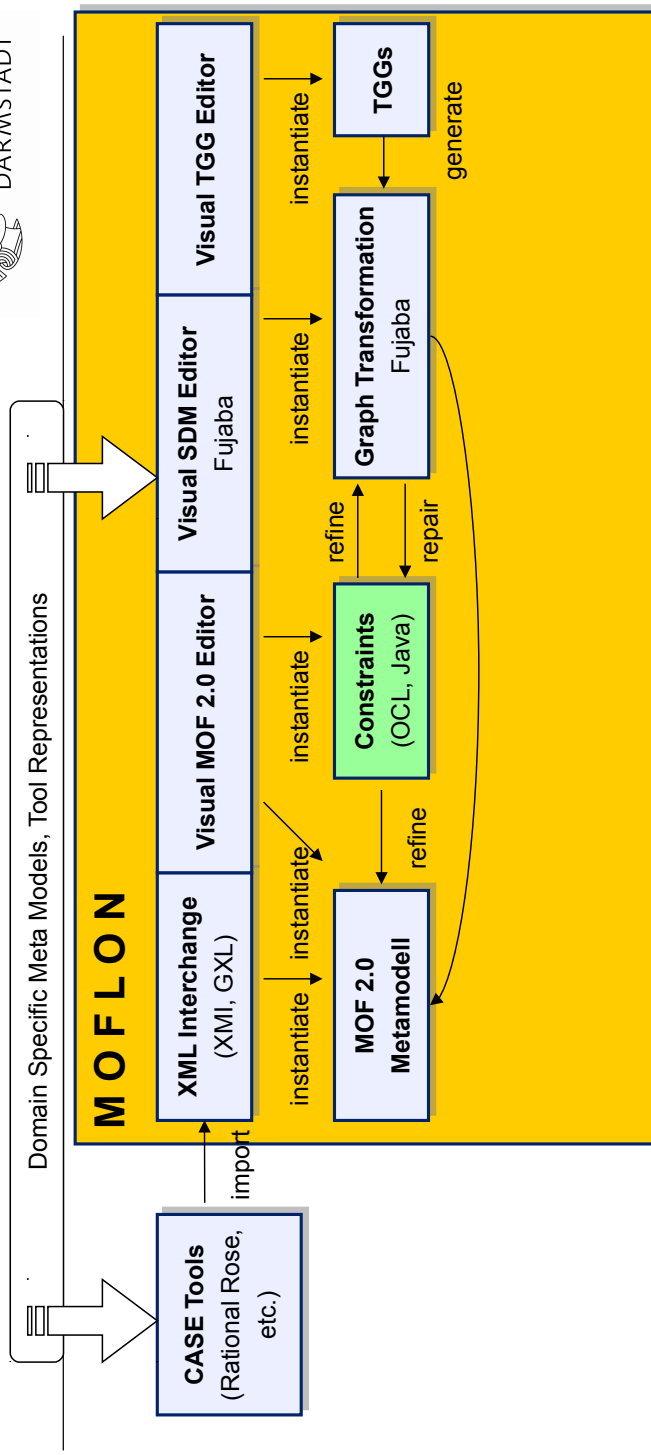
```



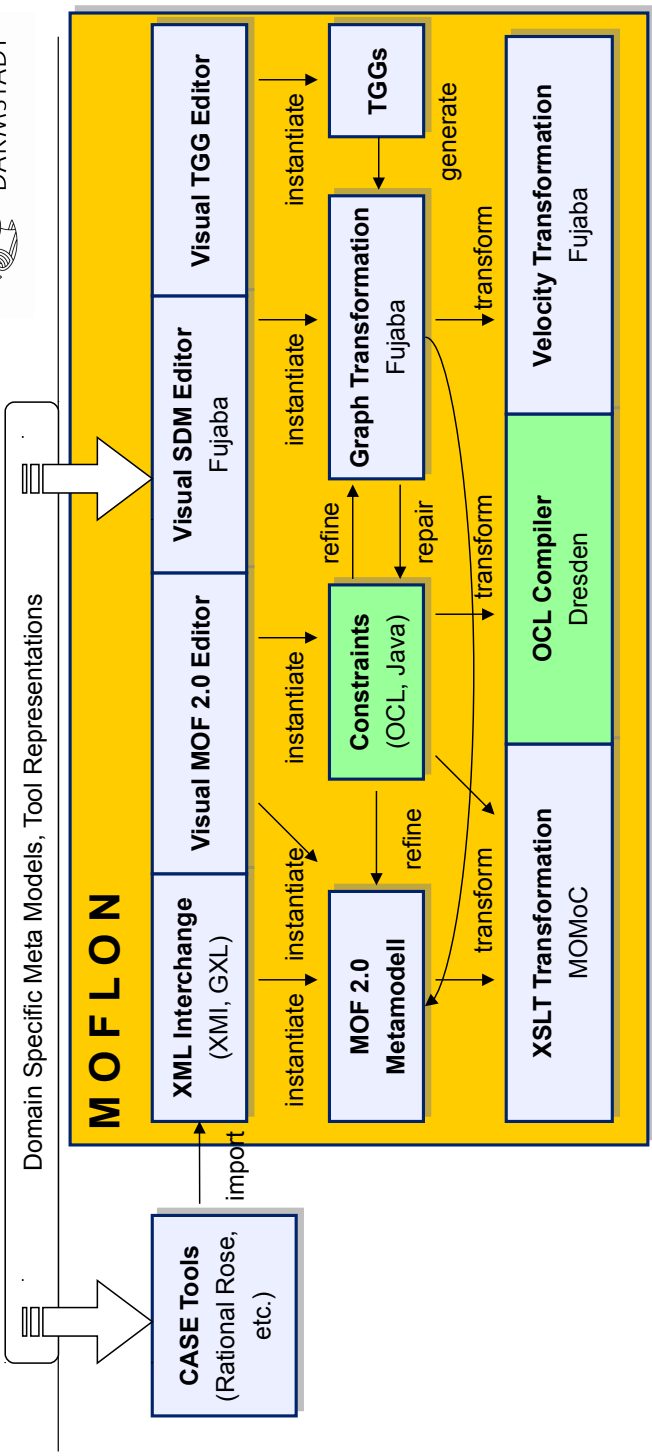
MOFLON – Architecture



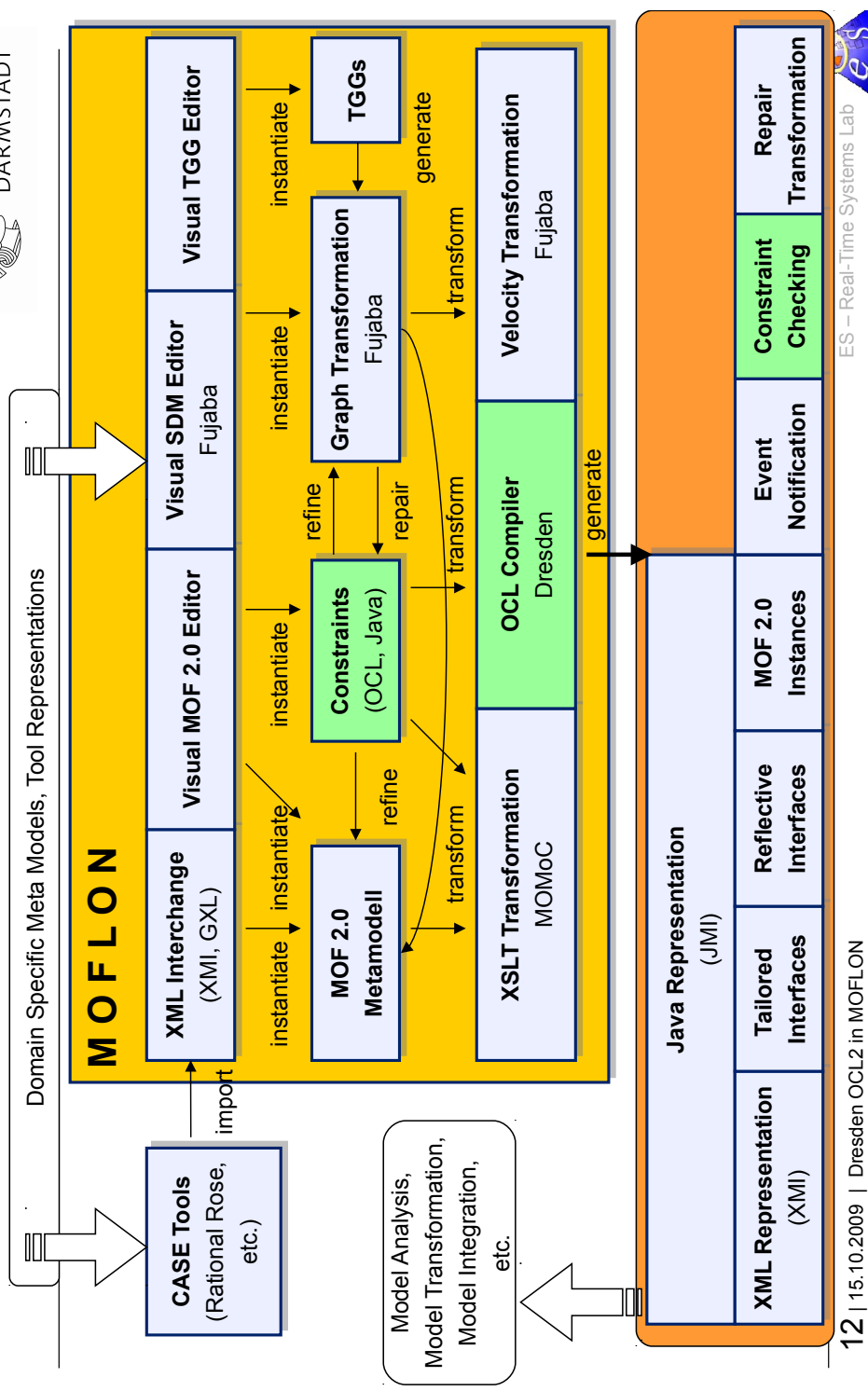
MOFLON – Architecture



MOFLON – Architecture



MOFLON – Architecture



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

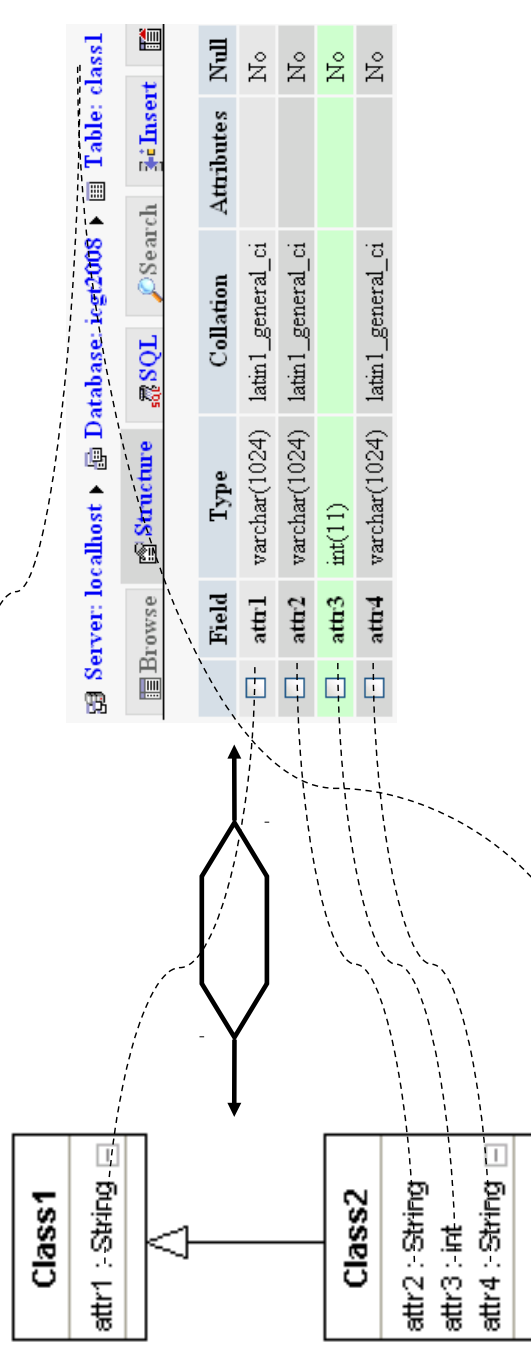
MOFLON is mainly used to

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

ES – Real-Time Systems Lab

13 | 15.10.2009 | Dresden OCL2 in MOFLON

Integration Example with TGG – Class diagrams / database schemata



domain specific language,
e.g. Class Diagrams

domain specific language,
e.g. Database Schemata

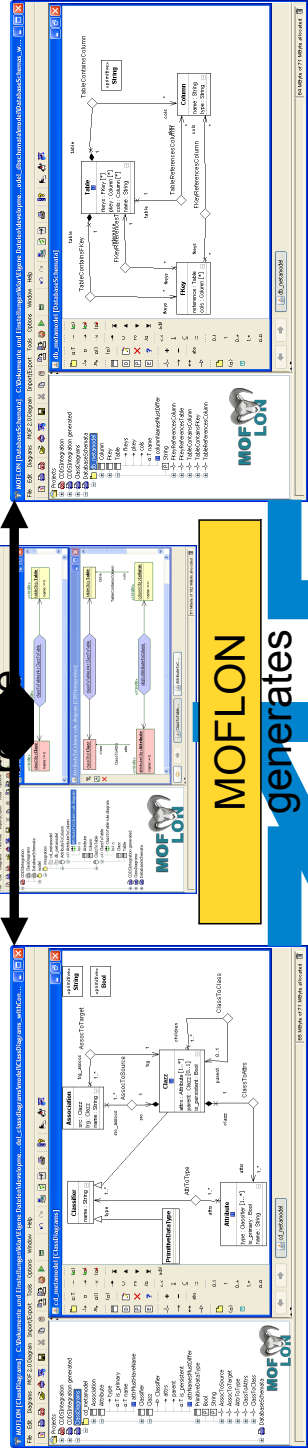
14 | 15.10.2009 | Dresden OCL2 in MOFLON

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

Class Diagrams Metamodel

TGGs

Database Schemata Metamodel



MOFLON
generates

integration rule code

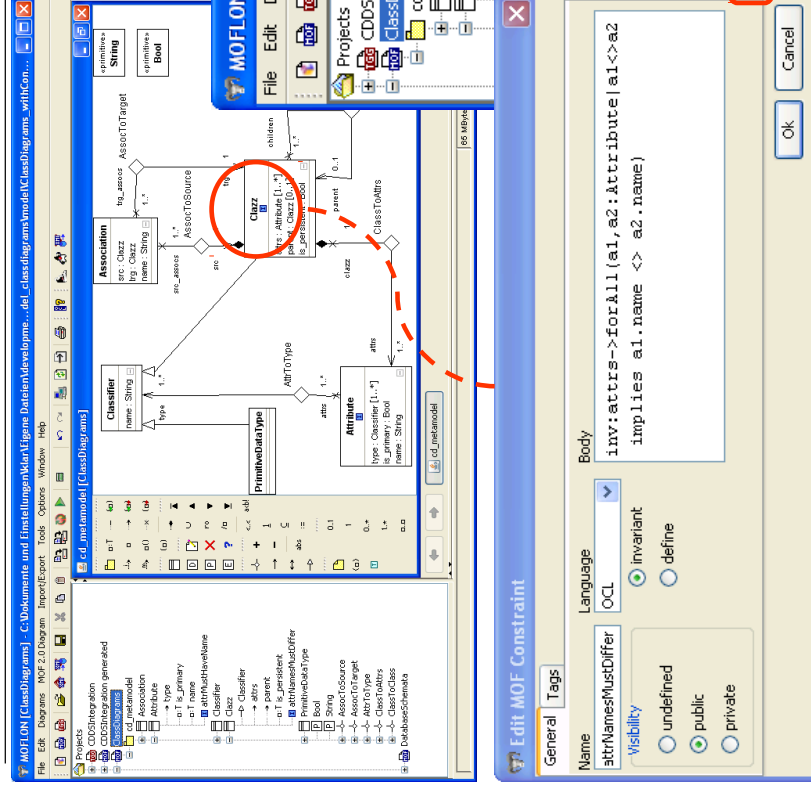
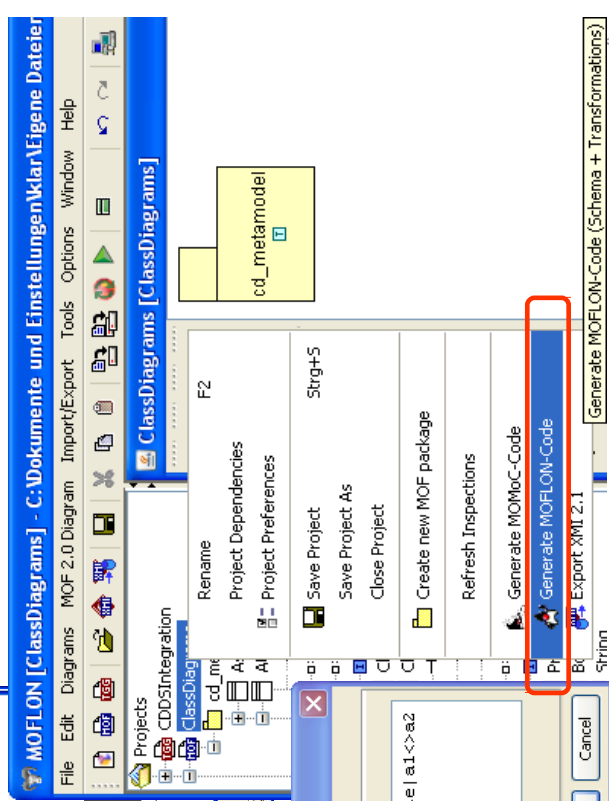
Integration Framework

Run-Time Verification
of Constraints

15 | 15.10.2009 | Dresden OCL2 in MOFLON

ES – Real-Time Systems Lab

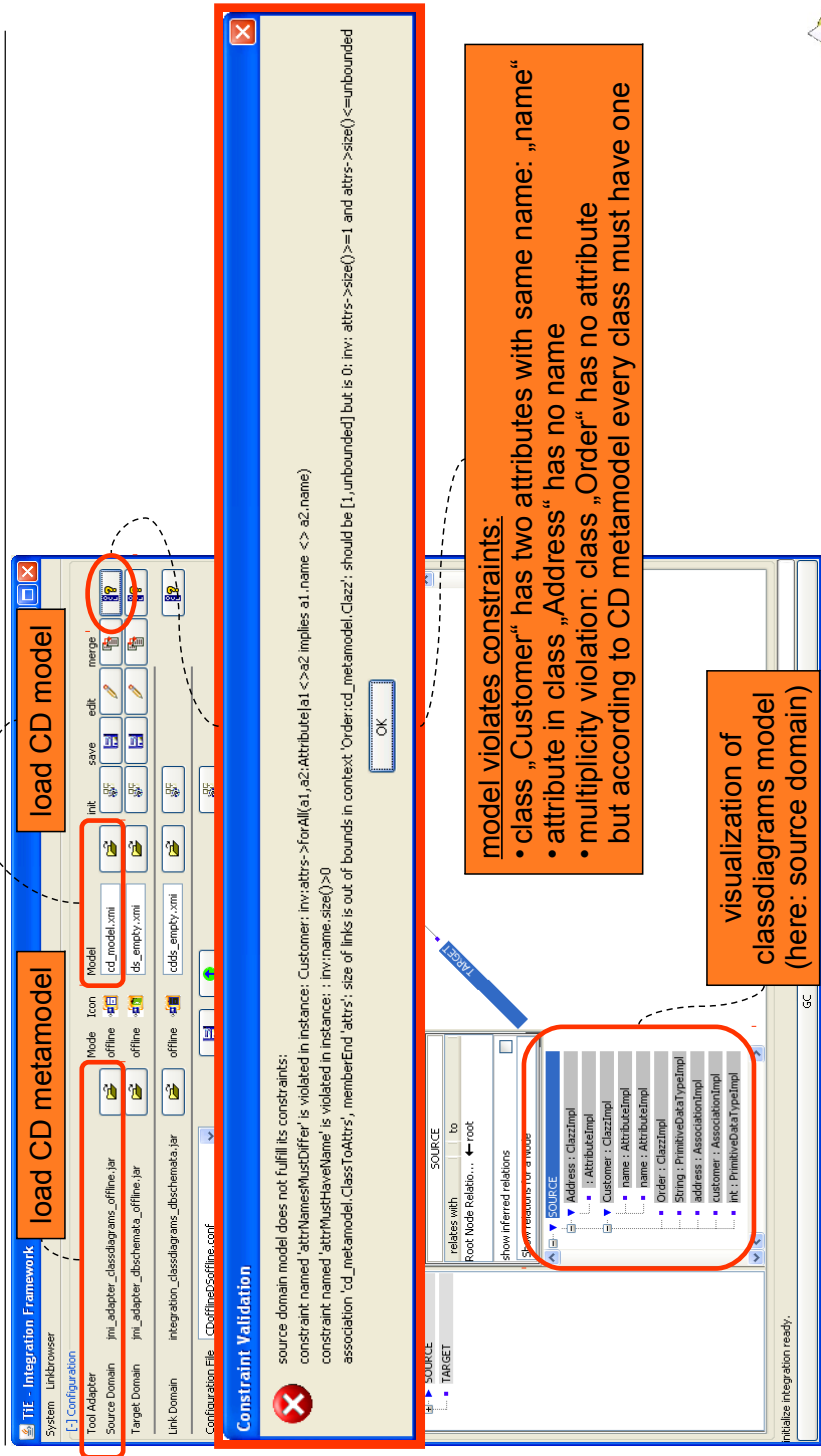
TiE-CDDS – Focus on Constraints in CD (1) Generate Code from MOF model (CD metamodel)

16 | 15.10.2009 | Dresden OCL2 in MOFLON

ES – Real-Time Systems Lab

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



load CD metamodel

load CD model

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.Class': should be [1,unbounded]] but is 0; inv. attrs->size()>=1 and attrs->size()<=unbounded

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

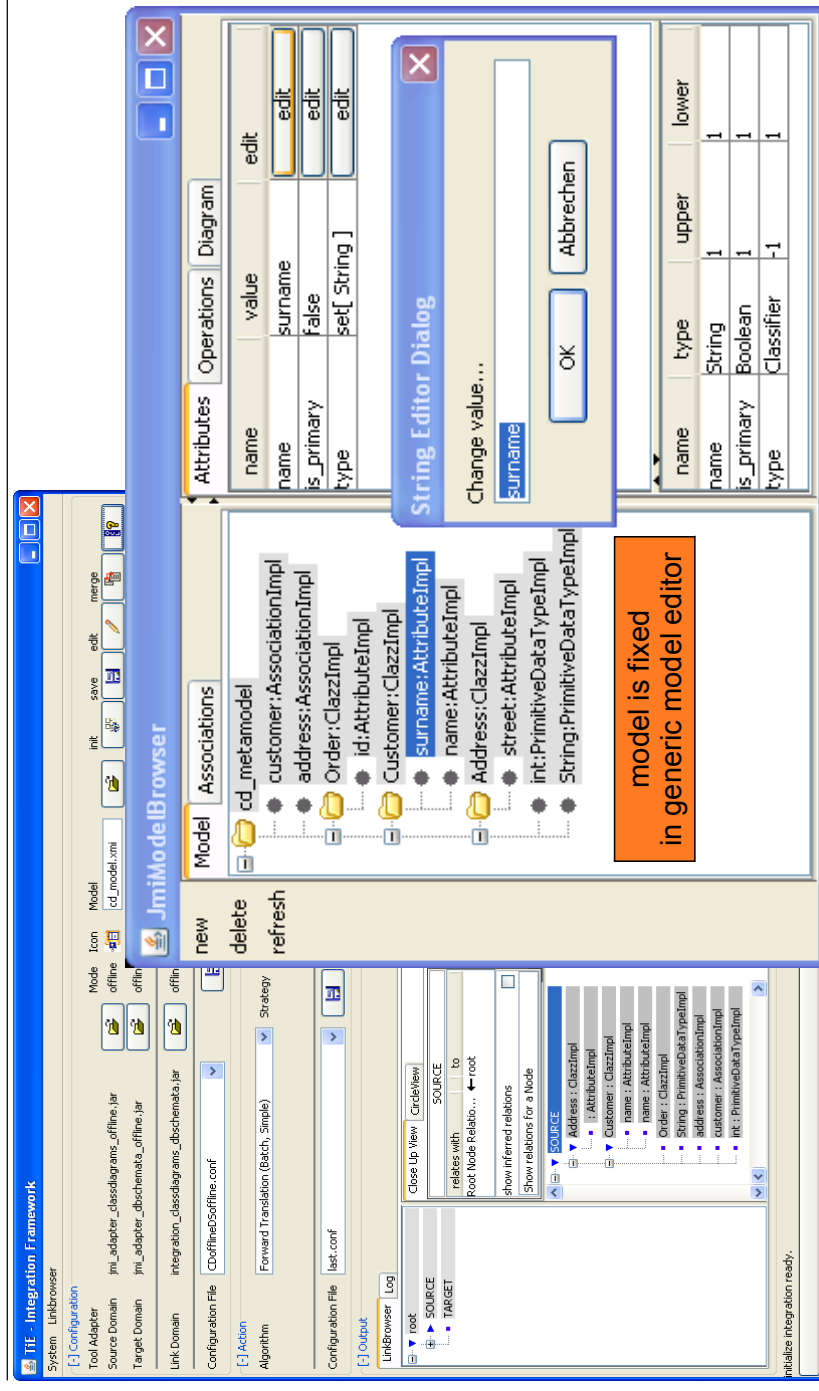
visualization of classdiagrams model (here: source domain)

```

classDiagram
    class Address {
        name : ClassImpl
    }
    class Customer {
        name : ClassImpl
    }
    class Order {
        name : AttributeImpl
    }
    class String {
        name : PrimitiveDataTypesImpl
    }
    class AddressImpl {
        name : AssociationImpl
    }
    class CustomerImpl {
        name : AssociationImpl
    }
    class OrderImpl {
        name : PrimitiveDataTypesImpl
    }
    class StringImpl {
        name : PrimitiveDataTypesImpl
    }
    AddressImpl --> CustomerImpl
    OrderImpl --> StringImpl
    
```

17 | 15.10.2009 | Dresden OCL2 in MOFLON

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



JmiModelBrowser

Associations

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

Attributes

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

String Editor Dialog

Change value...

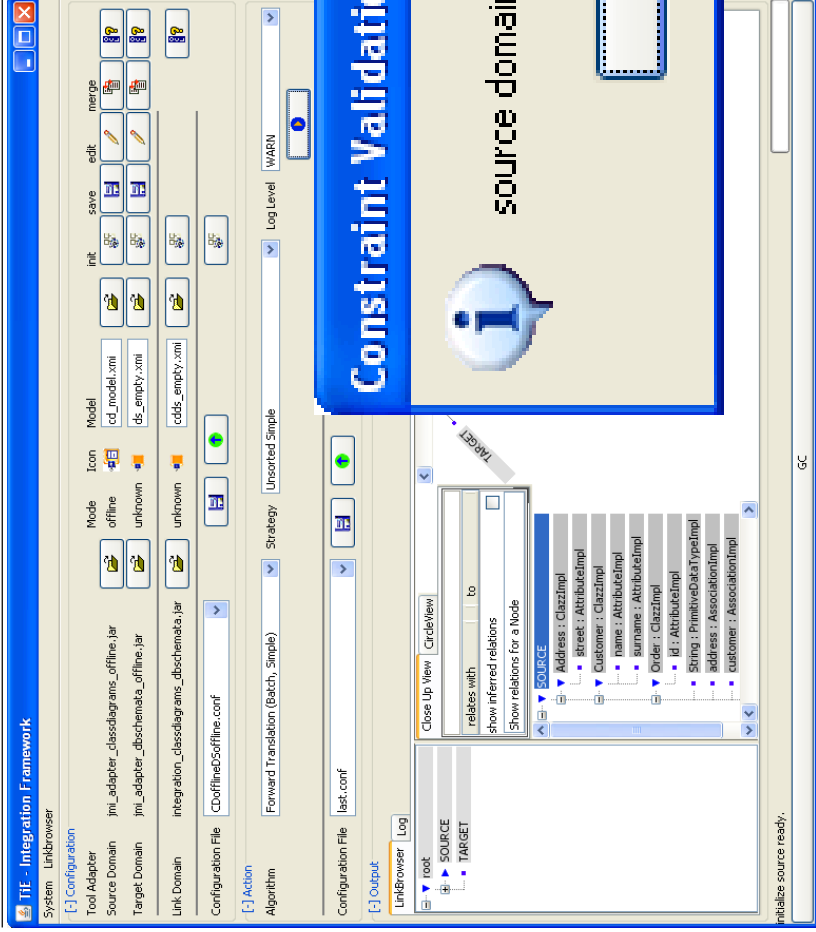
surname

OK Abbrechen

model is fixed in generic model editor

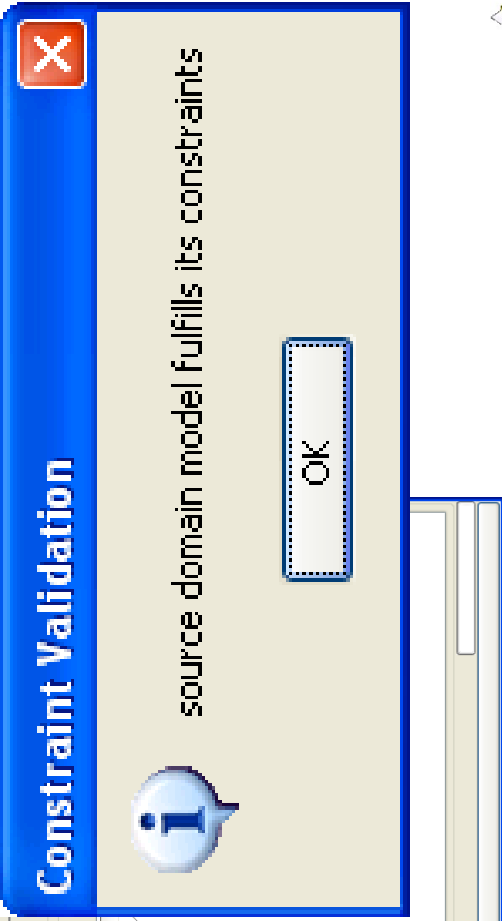
18 | 15.10.2009 | Dresden OCL2 in MOFLON

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



The screenshot shows the TiE-Integration Framework interface. The 'Tool Adapter' section is active, displaying settings for 'ini_adapter_classdiagrams_offline.jar' and 'ini_adapter_observable_offline.jar'. The 'Link Domain' is set to 'integration_classdiagrams_observable.jar'. The 'Configuration File' is 'C:\offlineds\offline.conf'. The 'Action' is 'Forward Translation (Batch, Simple)' and the 'Strategy' is 'Unsorted Simple'. The 'Log Level' is set to 'WARN'. The 'Output' window shows a tree view with 'root' expanded to show 'SOURCE' and 'TARGET' nodes.

translation process
may start now ...



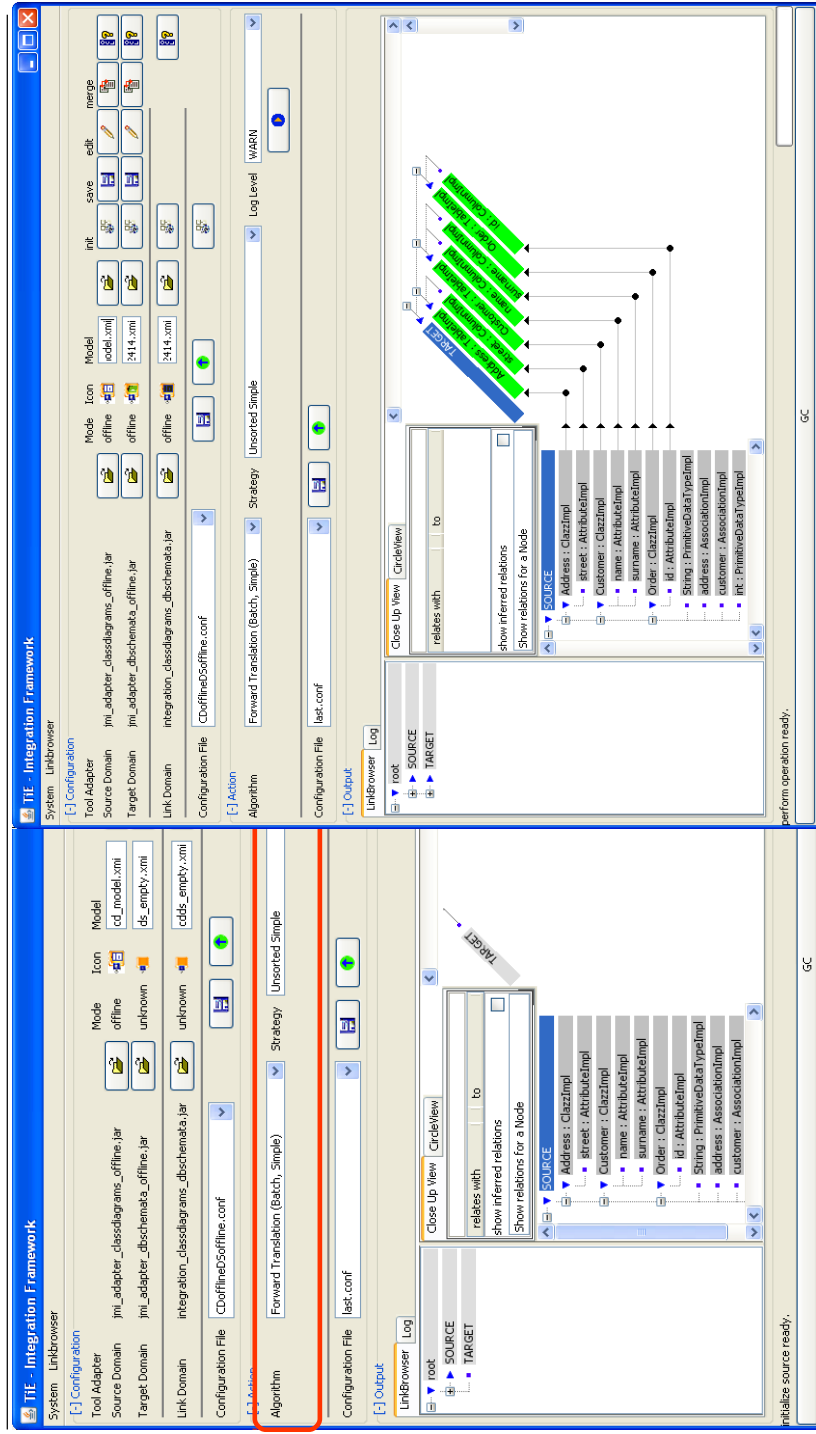
Constraint Validation

source domain model fulfills its constraints

OK

19 | 15.10.2009 | Dresden OCL2 in MOFLON

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



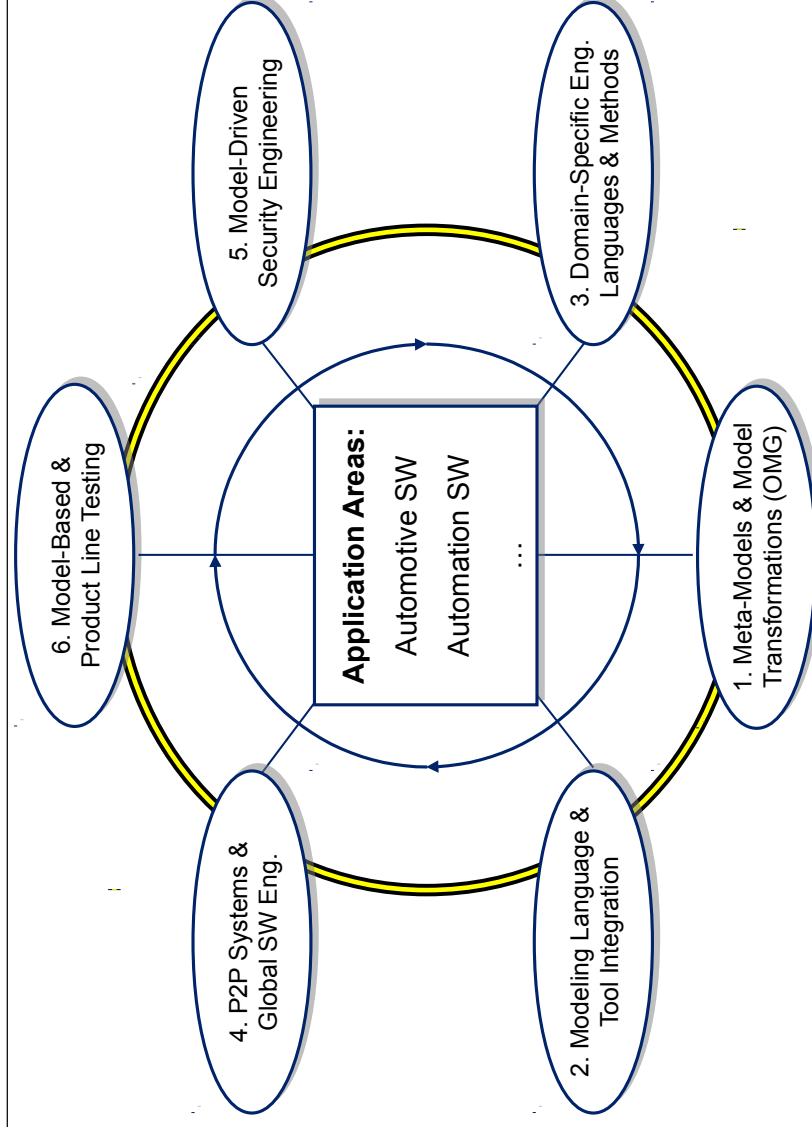
The screenshot shows the TiE-Integration Framework interface with the 'Forward Translation (Batch, Simple)' action selected. The 'Output' window shows a tree view with 'root' expanded to show 'SOURCE' and 'TARGET' nodes. The 'Target Domain' is 'integration_classdiagrams_observable.jar'. The 'Configuration File' is 'C:\offlineds\offline.conf'. The 'Action' is 'Forward Translation (Batch, Simple)' and the 'Strategy' is 'Unsorted Simple'. The 'Log Level' is set to 'WARN'. The 'Output' window shows a tree view with 'root' expanded to show 'SOURCE' and 'TARGET' nodes. The 'Target Domain' is 'integration_classdiagrams_observable.jar'. The 'Configuration File' is 'C:\offlineds\offline.conf'. The 'Action' is 'Forward Translation (Batch, Simple)' and the 'Strategy' is 'Unsorted Simple'. The 'Log Level' is set to 'WARN'. The 'Output' window shows a tree view with 'root' expanded to show 'SOURCE' and 'TARGET' nodes. The 'Target Domain' is 'integration_classdiagrams_observable.jar'. The 'Configuration File' is 'C:\offlineds\offline.conf'. The 'Action' is 'Forward Translation (Batch, Simple)' and the 'Strategy' is 'Unsorted Simple'. The 'Log Level' is set to 'WARN'. The 'Output' window shows a tree view with 'root' expanded to show 'SOURCE' and 'TARGET' nodes.

20 | 15.10.2009 | Dresden OCL2 in MOFLON

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)



Related Approaches

	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	Poumanu	DiaGen	EBNF & TXL	SQL	XML
Abstract syntax	+	+	+	+	+	+	0	0	+	+	0	+	+	0	+
Concrete syntax	--	--	--	+	+	+	--	+	+	+	+	+	--	--	--
Static semantics	+	+	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
Acceptability	+	+	0	--	0	0	+	+	+	--	0	+	0	+	+
Scaleability	+	+	--	0	--	0	--	0	--	--	0	--	--	--	0
Tool availability	--	0	0	+	+	+	+	+	+	0	0	+	+	+	0
Expressiveness	+	+	0	+	+	+	0	0	0	0	0	0	+	0	0

from Amelunxen, Königs, Röttschke, 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.**

