QVT & Co.

Perlen der Informatik

21. November 2007

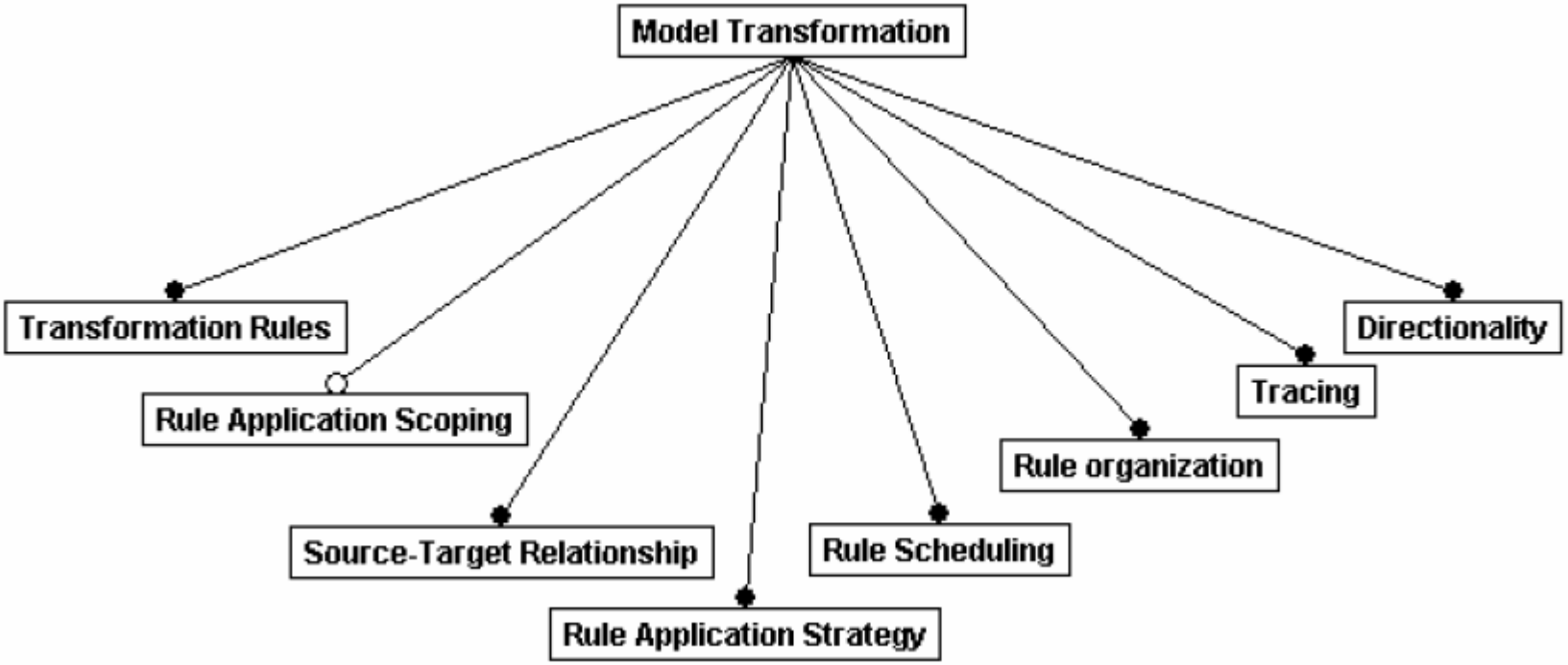
Florian Hölzl

Department of Informatics
Chair IV: Software & Systems Engineering

Agenda

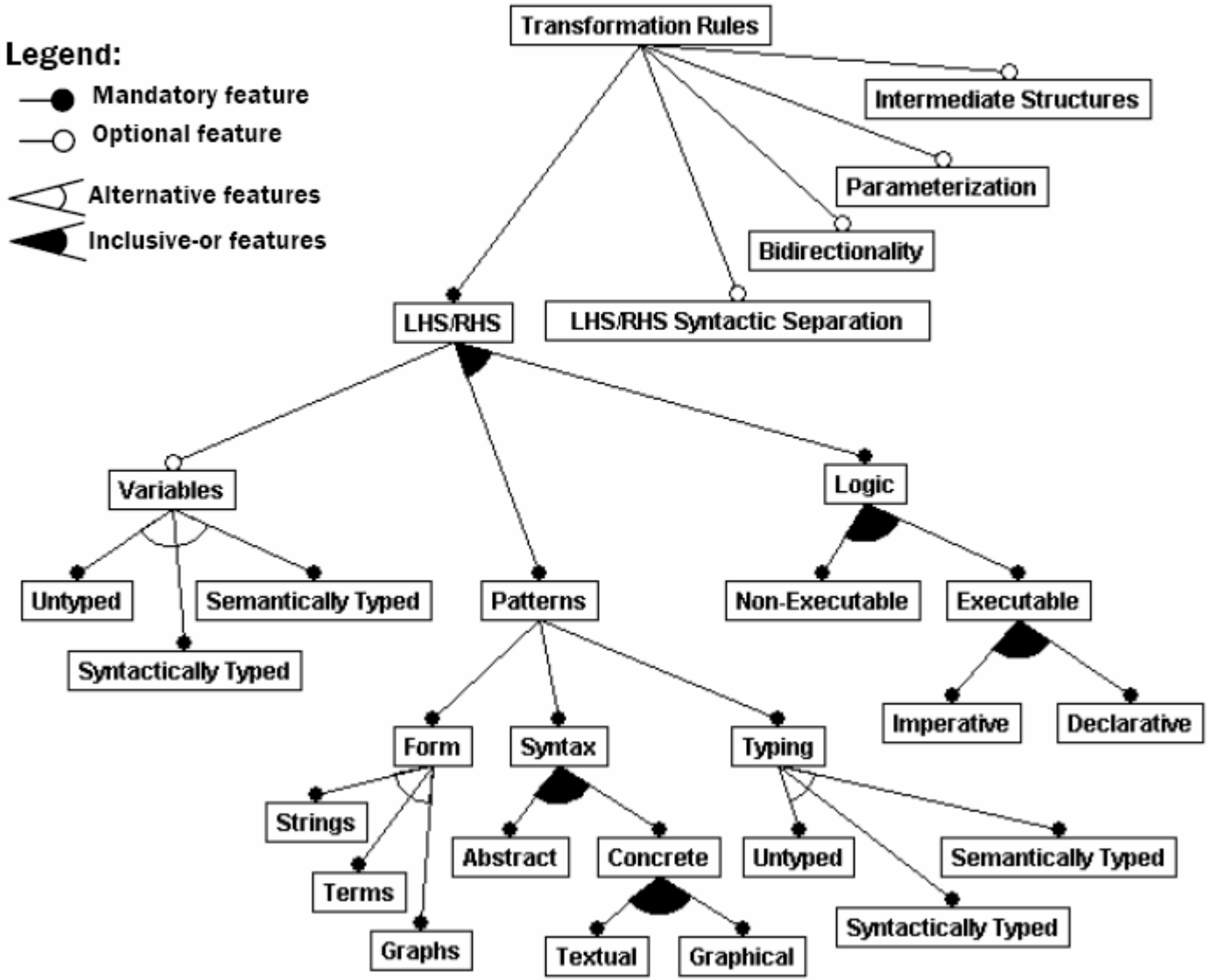
- Model Transformation
- QVT
- TUM-I4 Approaches

Classification of transformation approaches



Czarnecki, Helsen: 2nd OOPSLA Workshop on Generative Techniques in the context of Model Driven Architecture, 2003

Classification of transformation languages

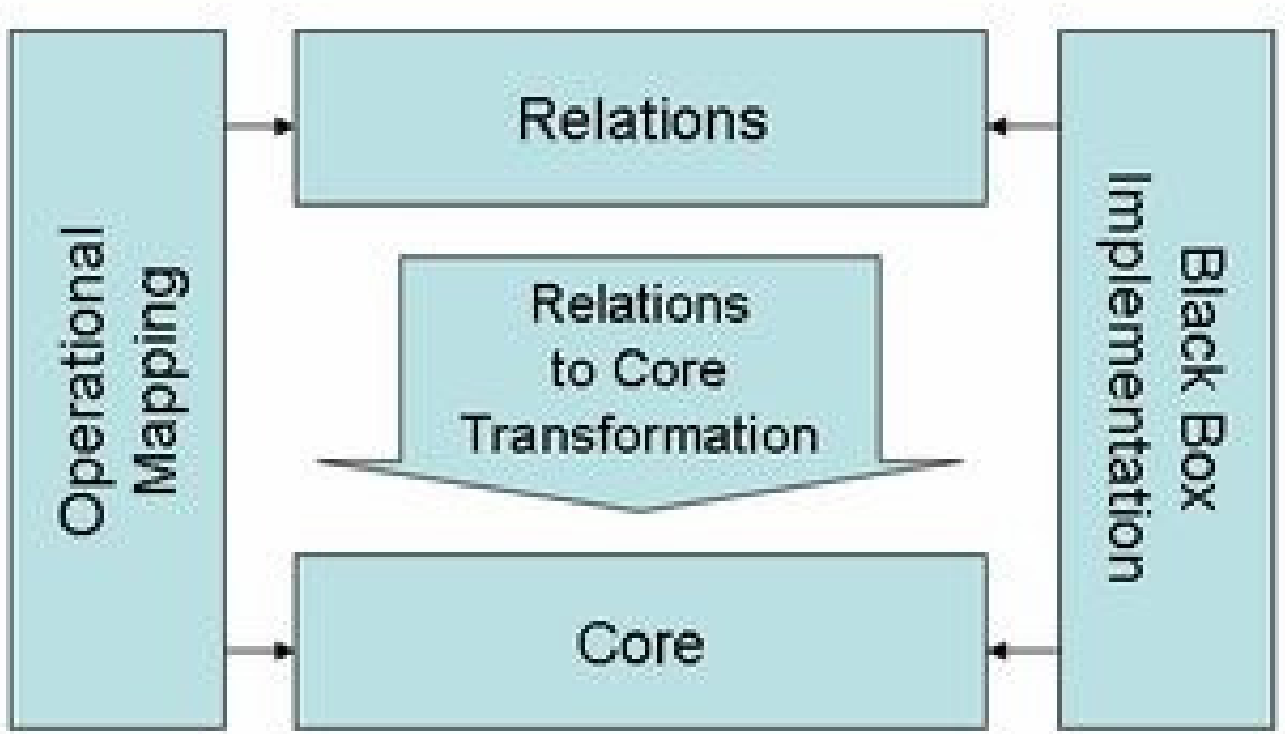


TECHNISCHE UNIVERSITÄT MÜNCHEN

Origin

- OMG RFP April 2002
- Proposals
  - Adaptive Ltd.
  - DSTC/IBM
  - Compuware Corporation / SUN
  - Alcatel / … / Thales
  - Kennedy Carter
  - Artisan, Kings College, Univ. of York
  - Codagen Technology Corporation
  - Interactive Objects Software GmbH
- Final Adopted Specification July 2007

The language world of QVT



QVT Core

- Basic capability to match patterns a bind variables
- Explicit trace model classes
- More or less a unification based evaluation algorithm

- Different semantics for checking or enforcement mode

Core Transformation & Trace Class

```
transformation umlRdbms {
   uml imports SimpleUML;
   rdbms imports SimpleRDBMS;
}
```

Source Domain

Target Domain

```
class PackageToSchema {
   composite classesToTables : Set(ClassToTable)
        opposites owner;
   composite primitivesToNames : Set(PrimitiveToName)
        opposites owner;
   name : String;
   umlPackage : Package;
   rdbms schema : Schema;
}
```

Trace Class

Core Mapping

```
map packageToSchema in umlRdbms {
  uml () { p:Package }
  rdbms () { s:Schema }
  where () {
      p2s:PackageToSchema |
      p2s.umlPackage = p;
      p2s.schema = s;
  }
  map {
       where () {
            p2s.name := p.name;
            p2s.name := s.name;
            p.name := p2s.name;
            s.name := p2s.name;
} } }
```

Variable Binding

Trace Class Instantiation

Unification Condition

QVT Relations Transformations

- Transformation = set of relations
  - Top-level relations hold for transformation
  - Non-top-level hold indirectly (`where`)
- Relations over two or more domains (=meta models)
- Execution selects one domain as target
- Relations are checked, and enforced on failure
- Relations are constrained by `when` and `where` terms

```
A(s,t) when B(u,v) where C(w, x)
```

`B(u,v)` holds => `A(s,t)` holds => `C(w,x)` holds

Relations Transformation

```
transformation umlToRdbms(uml:SimpleUML, rdbms:SimpleRDBMS) {
   key Table (name, schema);
   key Column (name, owner);
   key Key (name, owner);

top relation PackageToSchema {
   pn: String;
   checkonly domain uml p:Package {
        name = pn
   };
   enforce domain rdbms s:Schema {
        name = pn
   };
}
```

Object Identification

Source Domain

Target Domain

Relations Transformation

```
top relation ClassToTable {
    cn, prefix: String;
    checkonly domain uml c:Class {
          namespace=p:Package {},
          kind='Persistent',
          name=cn
    };
    enforce domain rdbms t:Table {
          schema=s:Schema {},
          name=cn,
          column=cl:Column {
                  name=cn+'_tid',
                  type='NUMBER,
          },
          key=k:Key { name=cn+'_pk', column=cl }
    };
    when { PackageToSchema(p, s); }
    where { prefix = ''; AttributeToColumn(c, t, prefix); }
}
```

QVT Operational Mappings

- Uni-directional transformations
- Signature of participating models
- Entry point `main()`
- Mapping operations define the actual transformation
  - Operational Syntax (including imperative OCL)
  - Implicit Relations (`when` & `where` terms)
  - Additionally explicit object creation / update
    - Constructor concept
  - Helper Operations (`query` & `helper`)
- Library concept
- „The execution semantics of an operational transformation is described below through comparisons with the Java language.“

Operational Transformation

```
transformation Uml2Rdb in srcModel:UML,out dest:RDBMS);
```

Source & Target Model

```
main() {
   srcModel.objects()[Class]->map class2table();
   srcModel->objectsOfType(Association)->map asso2table();
}
```

Sub Mapping Calls

TECHNISCHE UNIVERSITÄT MÜNCHEN

Operational Mapping

```
mapping Class::class2table () : Table
when {self.kind='persistent';} {
    init {                                    implicit object creation
        self.leafAttributes := self.attribute
                ->map attr2LeafAttrs("","");
    }
    name := 't_' + self.name;
    column := self.leafAttributes
        ->map leafAttr2OrdinaryColumn("");
    key_ := object Key {                      explicit object creation
        name := 'k_'+ self.name;
        column := result.column[kind='primary'];
    };
}
```

Implementations

- QVT Operational
  - Borland Together Architect (M2M Eclipse project)
  - SmartQVT: Eclipse OS implementation
- QVT Relations
  - ModelMorf: proprietary, closed-source
  - MediniQVT (ikv++ technologies): Eclipse based
- QVT Core
  - OptimalJ (Compuware): targets at industrial use
  - MTF (IBM): partially QVT-compliant
- QVT-like
  - ATL: also part of Eclipse M2M
  - MOFLON: Triple Graph Grammars based on Fujaba
  - Tefkat: F-Logic based transformation (SQL-like syntax)

Literature

- MOF 2.0 Query / View / Transform RFP
  http://www.omg.org/cgi-bin/doc?ad/2002-4-10
- Review of the QVT Proposals
  http://ww.omg.org/docs/ad/03-08-02.pdf
- QVT Final Adopted Specification
  http://www.omg.org/cgi-bin/doc?ptc/2007-07-07
- Classification of Transformation Approaches
  www.swen.uwaterloo.ca/~kczarnec/ECE750T7/czarnecki_helsen.pdf

TUM-I4 Approaches

- Bi-directional Object Transformation Language (BOTL)
- Consistency Constraint Language (CCL)
  - Part of AF1
- Operation Definition Language (ODL)
  - Part of AF2
- Rulebased Interactive Transformation Language (RITL)
  - Part of AF3
  - See Summer Hut 2007 for Foundations

Thank you very much for your attention !

Department of Informatics
Chair IV: Software & Systems Engineering