

Model Transformation goes declarative



QVT Relations in practice

Hajo Eichler

[eichler@ikv.de]

who we are

ikv++ technologies

- located in Berlin (de) and Yokohama (jp)
- developing model-driven solutions
- focus on automotive domain
- eclipse user

medini

- umbrella of our base technology
- integration and customization

medini QVT

- OMG's QVT Relations implementation
- just another part of our tech pool
- started 1,5 years ago

Consulting

**model-based
process automation
and optimization**

medini solutions

**customized tool and
development
environments for
our customers**

medini base technology

**meta modeling
model transformation
domain specific modeling
methods and
tool adapters**



medini QVT engine - facts

OMG's QVT Relations implementation with compliance level:
syntax execution compliance

- pattern matching of object templates
- multiple domains – multiple models
- relation calls via *when* and *where* clauses
- OCL queries
- key concept
- in-place transformations
- black-box through user operations
- relation import
- (no object templates)

trace management

- enables incremental updates on re-transformation
- enables backward transformation (depends on rule)

direct implementation (no QVT Core mapping)



medini QVT editor

The screenshot displays the medini QVT editor interface with the following components:

- Class Diagram (Top Left):** A UML class diagram showing relationships between classes:
 - Block** (base class) has subclasses **RootBlock**, **Shape**, and **SimpleShape**.
 - Shape** has subclasses **Circle**, **Triangle**, and **Square**.
 - ModelElement** (base class) has subclasses **Arrow** and **SimpleShape**.
 - ModelElement** has an attribute **name**.
 - Block** has a directed association to **ModelElement** with multiplicity **0..*** and role **modelElement**.
 - Shape** has a directed association to **Arrow** with multiplicity **0..1** and role **source**.
 - Arrow** has a directed association to **Shape** with multiplicity **0..1** and role **target**.
 - Arrow** has a directed association to **Block** with multiplicity **0..*** and role **outArrow**.
 - Block** has a directed association to **Arrow** with multiplicity **0..*** and role **inArrow**.
- Project Tree (Bottom Left):** A tree view showing the project structure:
 - Transformation r1
 - Model source
 - Model target
 - Relation ArrowSource2ArrowSource
 - Relation ArrowTarget2ArrowTarget
 - Relation ModelElementName2ModelElementName (selected)
 - Relation Show variable dependencies
 - Show pseudo code (highlighted)
 - Relation DomainImpl
 - DomainPatternImpl
 - ObjectTemplateExpImpl (type=Square)
 - VariableDeclarationImpl
 - VariableDeclarationImpl
 - Relation ModelElementParent2ModelElementParent
 - Relation Source2Target
 - Relation MapRootBlock
 - Relation MapBlock
 - Relation MapTriangle
 - Relation MapSquare
 - Relation MapArrow
- Code Editor (Bottom Right):** Displays the QVT transformation code for the selected relation:


```

57 top rela
58
59 info
60 );
61 enforce domain target t : ShapeLanguageMetamodel::Square {
62     inArrow - OrderedSet(ShapeLanguageMetamodel::Arrow)
63     name - String
64     outArrow - OrderedSet(ShapeLanguageMetamodel::Arrow)
65     parent - ShapeLanguageMetamodel::Block
66 }
67 top relat
68
69 enfor
70 p
71 );
72
73 enforce domain target targetModelElement : ShapeLanguageMetamodel::Model
74 parent = targetParent : ShapeLanguageMetamodel::Block {
75
      
```
- Dialog Box (Center):** Titled "Ordered clauses of relation ModelElementName2ModelElementName", it contains the following text:


```

FOR ALL (sourceModelElement) ∈ ModelElement DO IF
sourceModelElement.name = >varName<
AND
Source2Target(sourceModelElement, >targetModelElement<)
THEN
>targetModelElement< := BIND BY TRACE OR CREATE ModelElement;
targetModelElement.name := varName;
END IF
      
```

medini QVT debugger

like the Java debugger

- step into, step over, step return and resume
- breakpoints (with conditions)
- call stack view
- variable view (changing variable bindings)
- expressions view (watch points with code assistant)

let's have a look...



last slide

source code available under the Eclipse Public License (EPL) at

<http://projects.ikv.de/qvt>

we welcome your feedback...

TRAC/Subversion system for

- downloads
- help and QA
- discussion forum
- **tutorial**
- **“how can I use the engine in my application?”**
- browse & checkout source code
- ticket system (bugs & features)



Hajo Eichler
[eichler@ikv.de]



ikv++ technologies ag
Bernburger Strasse 24-25
D-10963 Berlin
www.ikv.de