Visual QVT/R
A concrete graphical syntax for QVT/R
Content

- Who I am
- Motivation
- What is QVT?
- QVT/R Engines
- Graphical Syntax of QVT/R
- Problems
- Outlook
Who I am

Max Bureck

Student of Freie Universität Berlin

Employed at qme Software GmbH for work on Diploma Thesis:
"Development of a visual transformation-framework for QVT"
Motivation (I / II)

- **Model-to-Model (M2M) transformations**
  - Convert models of meta-model A into models of meta-model B (A may equal B)
  - Use Cases:
    - Model refinements
    - Keep models in sync (e.g. different views on same issue)
    - Transformation to a more concrete model
  - Advantages:
    - Separation of concerns
    - Can be checked statically (will target be valid?)
Motivation (II/II)

• **M2M-transformation DSLs**
  - Define transformations in an expressive way → less to write
  - Easy to understand due to concise syntax

• **Graphical syntax**
  - Often easier to conceive than textual representation
What is QVT? ( I / III )

- Standard of the Object Management Group
- Stands for Meta Object Facility (MOF) 2.0
  Query/View/Transformation
- Defines languages for M2M-transformations
- Consists of three languages
  - Core: declarative language, simple syntax → rules are complex to write
  - Relations: declarative language, has textual and graphical syntax
  - Operational Mappings: imperative language
What is QVT? ( II / III )

Taken from the QVT standard
What is QVT? ( III / III )

QVT Relations

- Advantages:
  - Standardized
  - On retransformation:
    - Modifies target-model, doesn't create it from scratch
    - Can keep changes in existing target-model
  - Multidirectional transformations are possible
  - Graphical syntax

- Disadvantages:
  - Source code a bit bloated
  - Declarative syntax needs to get used to
QVT/R Engines (I / II)

Eclipse QVT Declarative

- Scope: QVT/R and QVT/C interpreter, Eclipse integration
- Developed by: Obeo and Ed Willink
- State: Early in development
- License: EPL
QVT/R Engines ( II / II )

Medini QVT

- Scope: QVT/R interpreter, Eclipse integration
- Developed by: ikv++ technologies ag
- State: Some language features are missing (e.g. Collection Templates)
- License:
  - EPL for the Interpreter
  - Eclipse plug-ins are closed source, but freely available for non-commercial use only
Graphical Syntax of QVT/R (I / XI)

- Similar to UML Object Diagrams
- Idea:
  - Define object patterns (a set of templates)
  - Relation between patterns for different models
  - A transformation consists of one or more relations
  - Toplevel relations are entry-points for transformation
Graphical Syntax of QVT/R (II / XI)

Meta-models for example

```
Directory
  File
    name : EString
  0..* content
  0..1 directory

Tree
  Vertex
    name : EString
    value : EInt
  1..* children
  Leaf
  InnerVertex
    parent 0..1
```
Graphical Syntax of QVT/R ( III / XI )

- **Domain (Relation Parameter)**
  - InnerDirectoryToInnerVertex

- **Relation Needs to Hold if True**
  - When
    - id.content->size()>0;

- **Object Template**
  - id : Directory
    - name = n

- **Check / Enforce**
  - when
    - id.content->size()>0;

- **Typed Model (Transformation Parameter)**
  - InnerDirectoryToInnerVertex(id,iv);
  - EmptyDirectoryToLeaf(id,iv);
  - FileToLeaf(id,iv);
Equivalent in textual syntax

```plaintext
relation InnerDirectoryToInnerVertex {

  n : String;

  checkonly domain src d : dir::Directory {
    content = id : dir::Directory {
      name = n
    }
  };

  enforce domain dst v : tree::InnerVertex {
    children = iv : tree::InnerVertex {
      name = n
    }
  };

  when {
    id.content->size() > 0;
  }

  where {
    InnerDirectoryToInnerVertex(id, iv);
    EmptyDirectoryToLeaf(id, iv);
    FileToLeaf(id, iv);
  }
}
```
Graphical Syntax of QVT/R (V / XI)

RootDirectoryToInnerVertex

- «domain»
  - d : Directory
    - src : dir
    - name = n
  - iv : InnerVertex
    - dst : tree
    - name = n

when
- d.content->size()>0;
- d.directory.oclIsUndefined();
when
- InnerDirectoryToInnerVertex(d, iv);
- EmptyDirectoryToLeaf(d, iv);
- FileToLeaf(d, iv);

RootDirectoryToLeaf

- «domain»
  - d : Directory
    - src : dir
    - name = n
  - l : Leaf
    - dst : tree
    - name = n

when
- d.content->size()==0;
- d.directory.oclIsUndefined();
when
- EmptyDirectoryToLeaf(d, iv);

EmptyDirectoryToLeaf

- «domain»
  - d : Directory
    - src : dir
    - name = n
  - ed : Directory
{...}
when
- ed.content->size()==0;

FileToLeaf

- «domain»
  - d : Directory
    - src : dir
    - name = n
  - v : InnerVertex
    - dst : tree
    - name = n

when
- f.oclIsTypeOf(dir::File);
Graphical Syntax of QVT/R (VI / XI)

- **Example:**
  - Run a transformation from directory to tree with an existing target-model
  - We observe what relation "InnerDirectoryToInnerVertex" does
Called from relation 'RootDirectoryToInnerVertex' with Directory 'C:' and InnerVertex 'C:' All further calls will be recursive.
Graphical Syntax of QVT/R (VIII / XI)

**src**
- Directory
  - name = 'C:'
- Directory
  - name = 'prog'
- Directory
  - name = 'max'
- Directory
  - name = 'eclipse'
- File
  - name = 'eclipse.exe'

**dst**
- InnerVertex
  - name = 'C:'
  - value = 0
- InnerVertex
  - name = 'prog'
  - value = 1
- InnerVertex
  - name = 'max'
  - value = 1
Graphical Syntax of QVT/R (X/XI)

src

: Directory
  name = 'C:'

: Directory
  name = 'prog'

: Directory
  name = 'max'

: Directory
  name = 'eclipse'

: File
  name = 'eclipse.exe'

dst

: InnerVertex
  name = 'C:'
  value = 0

: InnerVertex
  name = 'prog'
  value = 1

: InnerVertex
  name = 'max'
  value = 1

: InnerVertex
  name = 'eclipse'
  value = 0

when
  id.content ->size()>0;
  where
  InnerDirectoryToInnerVertex(id,v);
  EmptyDirectoryToLeaf(id,v);
  FileToLeaf(id,v);
Graphical Syntax of QVT/R ( XI / XI )

- Visual QVT/R provides an editor for the syntax (and more)
- More information soon to come
Problems ( I / II )

- For several elements in abstract syntax there is no graphical Syntax
  - Key definitions
  - Queries
  - No indication for toplevel relations
  - ...

- Some Elements of the graphical syntax have unclear semantics

```
object template or collection template?
```

```
\begin{tikzpicture}
  \node (f) {f : File};
  \node (d) [below of=f] {\texttt{d : Directory}};
  \node (n) [below of=d] {\texttt{name = n}};
  \node (f2) [right of=n] {\texttt{\{not\}:File}};
  \node (c) [right of=f2] {\texttt{oclTypeOf(dir::File)}};
  \draw (f) -- (d);
  \draw (d) -- (n);
  \draw (n) -- (f2);
  \draw (f2) -- (c);
\end{tikzpicture}
```
Problems ( II / II )

- Problem for Implementation: Some syntax elements have no correspondence in abstract or concrete textual syntax
  - Not template
  - Relation middle

```
\begin{center}
\begin{tikzpicture}
\node[draw, rounded corners] (file) at (0,0) {\texttt{(not):File}};
\node[draw, diamond] (c) at (1,0) {C};
\node[draw, diamond] (e) at (2,0) {E};
\draw[->] (file) -- (c) node[midway, above] {src : dir};
\draw[->] (e) -- (file) node[midway, above] {dst : tree};
\end{tikzpicture}
\end{center}
```
Outlook

• Visualization of trace-data is interesting
  (→ last Eclipse DemoCamp)
  • Show transformation on 3D-trace visualization to show cause of errors
  • Or simply dye transformation-model according to selection in source
    or target-model

• **Graphical debuggers possible**
  • Add breakpoint-conditions to model-elements as OCL-constraints
  • Mark currently processed template and show match for template
Thank you for your attention

Questions?

max.bureck@fu-berlin.de

qme Software
Gustav-Meyer-Allee 25
13355 Berlin

Telefon 030/46307-230
Telefax 030/46307-649
Email info@qme-software.de
Web www.qme-software.de