An Overview of QVT

Dániel Varró

Budapest Univ. of Technology and Economics
Dept. of Measurement and Information Systems

QVT = Queries, Views, Transformations

- Outline of the talk:

July 26th, 2005

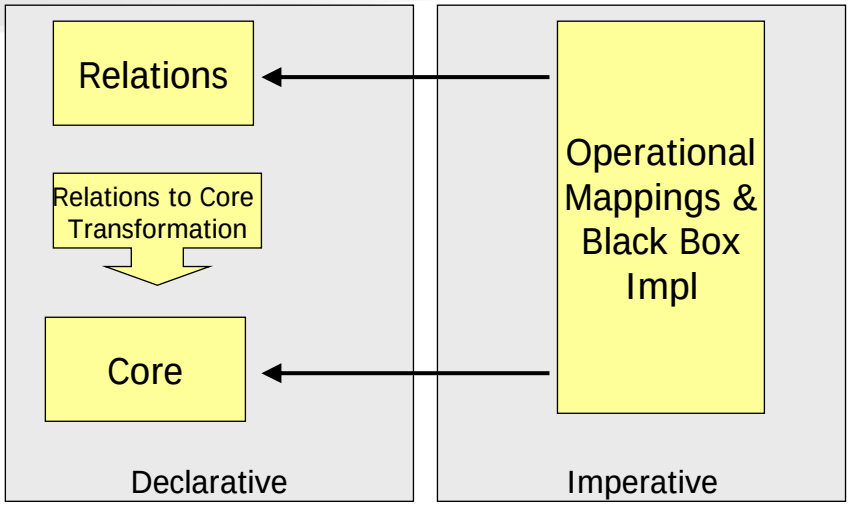
Objectives of QVT

- Mandatory requirements:
  - Query language
  - Transformation language from language S to language T
  - Defined over MOF metamodels
  - Executable / Portable vs. Black-Box XForm
  - Definition of Views
  - Declarative / Incremental transformations

Objectives of QVT

- Optional requirements:
  - Bidirectional transformations
  - Traceability of transformations
  - Reuse and extend generic transformations
  - Transformations as transactions: Commit and Rollback
  - Transformations within a Language

Language Overview



July 26th, 2005

Declarative spec: Relations

- Relations: declarative specification of the relationships between MOF models.
- Supports:
  - complex object pattern matching,
  - implicitly creates trace classes and their instances
  - to record what occurred during a transformation execution.
- Relations can assert that other relations also hold between particular model elements matched by their patterns

July 26th, 2005

Declarative spec: Core

- Core: small model/language
  - only supports pattern matching over a flat set of variables
  - by evaluating conditions over those variables against a set of models.
- It treats all of the model elements (source, target and trace) symmetrically.
- Virtual machine of transformations

Imperative spec: Mappings

- Mappings: standard way of providing imperative implementations to populate trace models
- Provides:
  - OCL extensions with side effects that allow a more procedural style
  - a concrete syntax that looks familiar to imperative programmers.
- Use:
  - implement a Relation (from a Relations spec)
  - when it is difficult to provide a purely declarative specification of how a Relation is to be populated.

Imperative Spec: Black-Box

- Black Box: MOF Operations may be derived from Relations making it possible to "plug-in" any implementation of a MOF Operation with the same signature.
- Goals:
  - It allows complex algorithms to be coded in any programming language with a MOF binding (or that can be executed from a language with a MOF binding).
  - It allows the use of domain specific libraries to calculate model property values (instead of OCL).
  - It allows implementations of some parts of a transformation to be opaque.

Execution scenarios for Core

- Check-only transformations to verify that models are related in a specified way
- Single direction transformations
- Bi-directional transformations
- Establishing relationships between pre-existing models
- Incremental updates (in any direction) when one related model is changed after an initial execution.
- Create and delete objects and values,
  - or specify which objects and values must not be modified.

QVT Core

Core Concepts

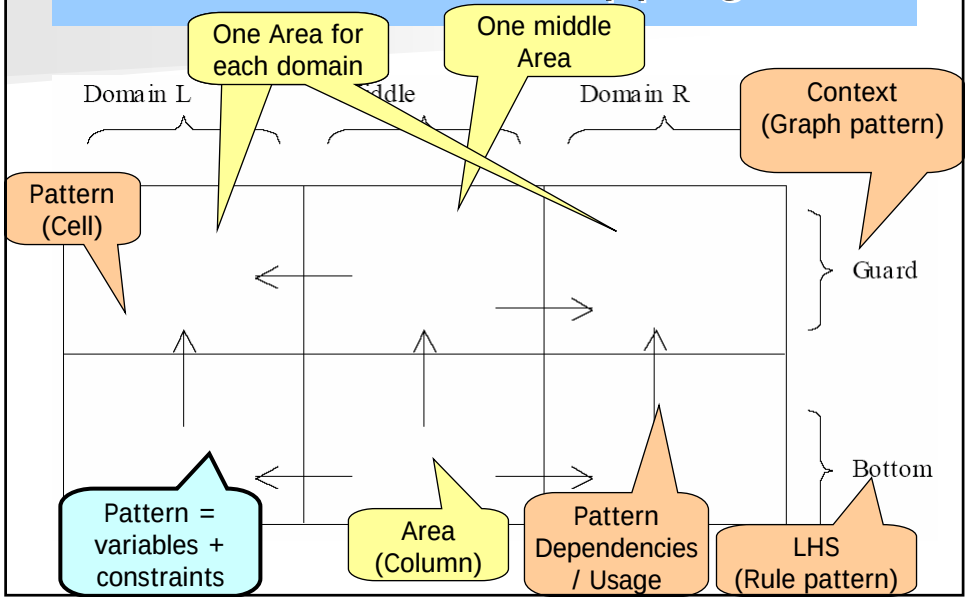
- Transformations are defined between models by sets of constraints and derivations.
- Transformations have directions (zero or more).
  - Each direction defines a source and/or target of a transformation.
- A user who executes a transformation can
  - check all constraints (no side effect), or
  - enforce (apply) all derivations of that transformation in one chosen direction.
- Package imports

Mapping

- A mapping
  - is part of one transformation
  - has zero or more domains.
- Each domain (of a mapping)
  - has one direction of the transformation.
  - Domains do not have names:
    the name of the direction identifies a domain uniquely in one mapping.
- When a transformation is executed
  - all mappings of that transformation are executed,
  - domains are derived in one chosen direction.

July 26th, 2005

Structure of mappings



Restrictions on patterns

- Each variable should be well-typed (e.g. package imports)
- If pattern C depends on pattern S, then C may use variables declared in S in declarations of constraints in C.
- Unclear to me:
  - When a direction A uses (?) another direction B then each pattern of domain dA that has direction A depends on the corresponding pattern of domain dB that has direction B.
  - There are never any dependencies between patterns of domains which do not use each others directions.

QVT: The Upcoming MT Standard

Binding (Matching)

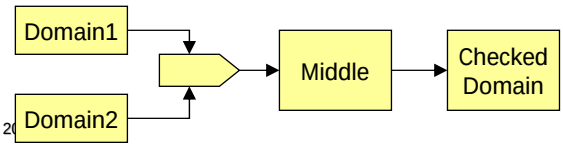
- A valid binding (when matching a pattern) is a binding where
  - all variables are bound to a value other than *undefined*, and
  - all constraints evaluate to *true*.
- Unclear: A binding is a unique set of values (???) for all variables of the pattern.
- A partial-valid binding is a binding
  - where one or more variables are bound to a value other than *undefined* or one or more constraints evaluate to *true,* and
  - no constraints evaluate to *false*.
- An invalid binding is a binding where at least one of the constraints evaluates to *false*.
- Consequence:
  - any valid binding (of a non-empty pattern) is a partial-valid binding,
  - partial-valid bindings of empty patterns do not exist.

Binding dependencies

- If a pattern C depends on patterns S1...Sn, then a valid binding of C needs one valid binding for each S1...Sn.
- Guards
  - Define the context of a matching
  - The matching of all bottom patterns only takes place in the context of a valid binding of all guard patterns.

Checking

- Each domain of a mapping can be checked
- Rule:
  - There must be (exactly) one valid combination of a valid-binding of the bottom-middle pattern and
  - one valid binding of the bottom-domain pattern of the checked domain,
  - for each valid combination of valid bindings of all bottom-domain-patterns of all domains not equal to the checked domain.



Enforcement

- The enforcement
  - Alters bottom-pattern bindings to fulfill the checking constraints.
  - Pattern bindings are only changed when the checking constraints of a mapping are not fulfilled.
  - At evaluation time, one direction can be chosen as the ***enforcement direction***.

Enforcement (cont.)

- Constraints (in the context of a binding) can be
  - predicates
  - assignments.
- A predicate evaluates to *true*, *false* or *undefined*.
  - Predicates are only used for matching.
- Assignments assign values to properties.
  - Assignments are used to provide values to variables,
  - properties of object-valued variables
  - in order to make a check of the patterns evaluate to *true*.

Enforcement (Default assignments)

- Unclear:
  Assignments may be default assignments
  - An assignment that is not a default assignment also has the meaning of a predicate, where the assignment operator is replaced by the equality operator.
  - Default assignments only assign values to satisfy the checking semantics, they do not have meaning as predicates.

Enforcement (Creation&Deletion)

- Variables may be realized
- When a variable is realized,
  - a new instance of the type of that variable may be created or
  - an existing value of that variable may be deleted.
- Role of realized variables:
  - Non-realized variables are only used for matching.
  - Realized variables are used for both matching and enforcement.
- Assignments and realized variables may only be defined in
  - bottom-domain patterns that are enforced and
  - in the bottom-middle pattern.

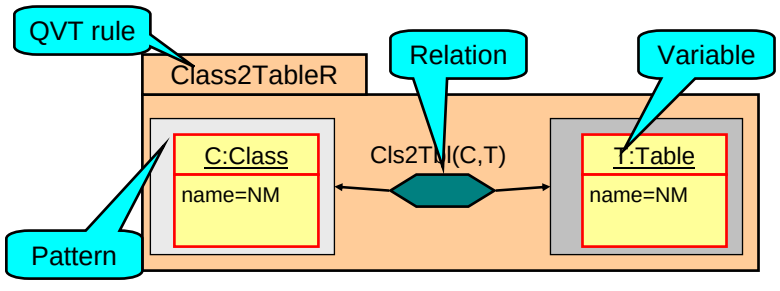
- Creation Rule:
  - If no valid binding of a certain pattern exists and
  - a valid binding of that pattern is required by the checking semantics,
  - then new instances of all unbound realized variable types are created and bound as values of the variables and all assignments are executed.
- Deletion Rule:
  - If a valid binding of a pattern exists and
  - it is required by the checking not to have a valid binding for that pattern,
  - then all assignments are nullified and all realized variables values are deleted.

Status of QVT

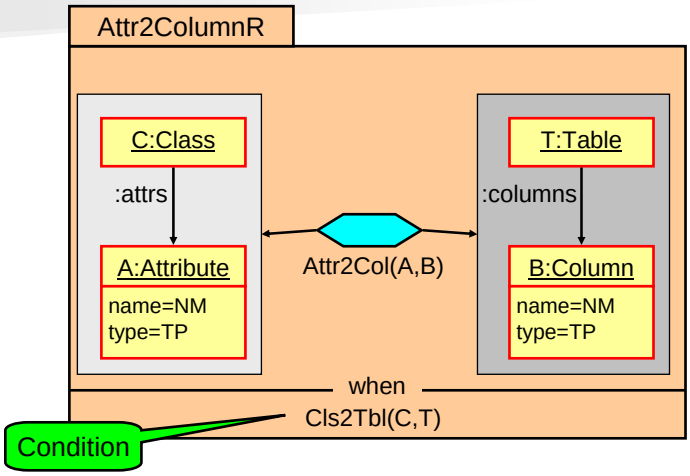
- **QVT = Revised submission for MOF 2.0 Query / View / Transformation RFP**
  - Final voting is scheduled to September 2005 by OMG
- Is QVT mature enough?
  - No proper tool support currently, BUT
  - Sooner or later there will be

- If QVT becomes a standard, will there be life for other MT approaches?

QVT example

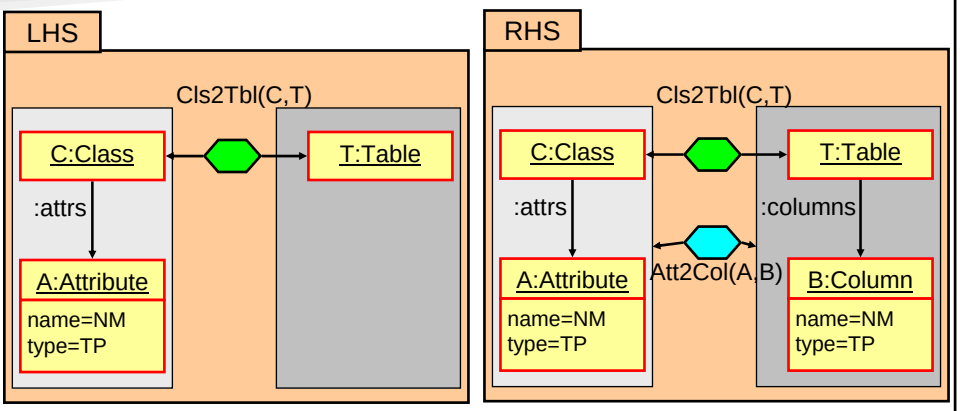


QVT example



July 26th, 2005

GT rule for QVT rule



July 26th, 2005

Final Questions / Challenges

- Will QVT be mature enough?
  - Sooner or later it will be
- What is the role of graph transformation?
  - Specification formalism: Maybe NOT
  - Execution engine: QVT2GT mapping?
  - Analysis: Reuse of GT results on QVT level?

Thanks for Your Kind Attention!

July 26th, 2005