Towards a Standard for MOF
Queries, Views, and Transformations

Jana Koehler, Rainer Hauser, Tracy Gardner, Catherine Griffin
Why a Standard?

- new role of models in MDA
  - not only an aid for understanding and communication
  - models as key parts of a *software system specification*
  - models enable *automatic code generation* via transformations

- UML2/MOF 2.0 as major representations

- unique language to query, view, and transform models becomes possible
Why an IBM Position Paper?

- OMG Request for Proposal for MOF 2.0 Query, Views, and Transformations (QVT)
  - IBM’s name is on the proposal by DSTC
  - 8 submissions were made (several hundred pages)

- Our goals
  - define **terminology** based on submissions, but edited for consistency
  - define **common transformation scenarios** based on our experience with MDA/QVT
  - **compare the submissions** based on the RFP + additional benchmarks
  - derive **recommendations**
About the authors

- we are ‘potential QVT users’ having *experience* in designing and implementing transformations
  - Hursley: MDBI Project
    - UML Profile for BPEL4WS to BPEL4WS, WSDL, XSD
  - Zurich: BPIA Project
    - transform business process models (Holosofx, UML2, OpS-BOpS-BOM) to BPEL4WS, WSDL, SACL

- ensure that QVT is *useable* for the kinds of MDA transformations that we want to implement

- help QVT to become a *widely adopted* standard

- make sure IBM supports the *best* proposal
Consistent Terminology I

- **query**: an expression evaluated over a model

- **view**: a model completely derived from another model

- **transformation**: generates a target model from a source model
  - may lead to independent or dependent models
  - **unidirectional**: changes are always made to the source model and propagated to the target model
  - **bidirectional**: each model may be modified, changes must be propagated in either direction
  - **update**: target model replaces the source model
Consistent Terminology II

- **declarative**: relationships between variables in terms of functions or inference rules + execution engine

- **imperative**: programming-language like transformations

- **hybrid**: combination of declarative and imperative constructs to define transformations
  - declarative approach is used to select rules for application
  - imperative approach is used to implement the detail of rules that are not completely expressed declaratively
Consistent Terminology III

- source-driven: the source model drives the build of the target model

- target-driven: the target metamodel drives the analysis of the source model

- M-N-transformations: it cannot be assumed that there is a one-to-one correspondence between source and target models
Common Scenarios

- simple vs. complex transformations
  - many submissions showed only simple transformations
    - source and target models have essentially the same structure
  - we faced complex transformations in our projects
    - structures in the target model have no simple correspondence in the source model

- regeneration and reconciliation
  - deal with user-modified transformation output
  - keep evolving models synchronized, deal with conflicts

- transformation from partial source models
  - resilience to errors, transactional behavior
  - generate valid, but incomplete models
The submissions

- Adaptive Ltd. (abbreviated with ADAPTIVE)
- DSTC/IBM (DSTC)
- Compuware Corporation/Sun Microsystems (SUN)
- Alcatel/Softeam/TNI-Valiosys/Thales (THALES)
- Kennedy Carter (KC)
- TCS, which comprises Artisan Software, Kinetum, King’s College, and the University of York (TCS)
- Codagen Technologies Corporation (CODA)
- Interactive Objects Software GmbH/Project Technology (IO)
Positioning the Submissions
- Query Languages -

- OCL 2.0 is the preferred language of choice
Positioning the Submissions
- Transformation Rules -

- guaranteeing the semantic correctness of a transformation will require a declarative solution
Positioning the Submissions - Transformation Execution -

- we will need the most flexible solution possible
Further Recommendations ...

- composition of transformations is important

1. Chained

   Input → Transform A → Output & Input → Transform B → Output

2. Merged

   Input
   
   Transform A
   
   Transform B
   
   Output

3. Embedded

   Input
   
   Transform A
   
   Output
   
   Input
   
   Transform B
   
   Output
Further Information

- QVT review+ position paper at OMG Website:
  - http://www.omg.org/docs/ad/03-08-02.pdf

- Business Process Integration and Automation Project at ZRL