Alignment of ATL and QVT

Ivan Kurtev

ATLAS group, INRIA & University of Nantes, France
http://www.sciences.univ-nantes.fr/lina/atl/

Context of this work

- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

Prerequisites

To be able to understand this lecture, a reader should be familiar with the following concepts, languages, and standards:

- Model Driven Engineering (MDE)
- The role of model transformations in MDE
- QVT
- ATL
- MOF

Outline

- Need for ATL and QVT alignment
- Requirements alignment of ATL and QVT
- Architectural alignment of ATL and QVT
  - QVT architecture
  - ATL architecture
  - Aligning the architectural components
- Achieving interoperability between ATL and QVT
- Conclusions

A DSL Perspective on MDE

- MDE allows definition of small (and large) languages, focused on specific problems known as Domain Specific Languages (DSLs)
- Typical tasks in MDE also require DSLs: for example, model transformations and language definition
- A set of transformation languages exists

Problems

- Language fragmentation (Babylon tower)
  - Need for managing DSLs
- Multiple languages per problem domain
  - Problem domains often overlap
  - Need for choosing among DSLs
- Need for clear guidelines and knowledge for:
  - Matching the problem domain and the available solutions (DSLs)
  - Explicit comparison among languages to allow selection of the right tool

Aligning ATL and QVT

- Two model transformation languages for MDE
- Solving the same problem (at first glance)
  - Is that true?

- Comparing ATL and QVT:
  - At problem domain level: what problems can be solved, what are the requirements per language
  - At architectural level: components and major capabilities

Requirements Alignment of ATL and QVT

- What are the problem domains of ATL and QVT?
- QVT is proposed in the context of OMG MDA approach
  - Targeted at software development
- ATL requirements evolved towards solving interoperability problems in data engineering
  - May solve software development problems
  - Deals with heterogeneous data

Characteristics of Software Development

- Transformations are mainly refinement of models;
- Models are expressed in a limited set of languages (e.g. UML)
- Languages often share the same conceptual foundation: Object-Oriented Principles
- Transformations should be semantic preserving
- Software development is iterative:
  - Change propagation should be supported
- Separation of concerns:
  - Need for model composition
- Reverse Engineering
- Refactoring

Characteristics of Data Engineering

- Heterogeneity of data and schemas (modeling languages):
  - RDBMS;
  - XML;
  - ORDBMS;
  - ER;
  - Many others;
- Need for data translation between heterogeneous data
- Need for declarative mappings for schema integration and query answering and translation
- Data migration (kind of data translation)

Requirements for QVT

An incomplete list (consult the QVT specification):

- Operates on MOF models (basically XMI-to-XMI transformations)
- Supports bidirectional transformations
- Declarative language (satisfied by Core and Relations languages)
- Checking the presence of certain relations among models

Requirements for ATL

Major requirement: ability to deal with various models expressed in different languages and technologies

Unification concepts:

- Everything is a model!
- Technologies are unified by the concept of Technical Space
- Heterogeneity is handled by the notion of Technical Projector
- Data translation between Technical Spaces is regarded as Bridging

Requirements alignment for ATL and QVT

- From the ATL perspective QVT solves transformational problems within the OMG/MDA Technical Space
- Without the concepts of Technical Projector and Bridging real interoperability problems cannot be solved (misses in QVT)
- ATL as a part of AMMA supports these concepts

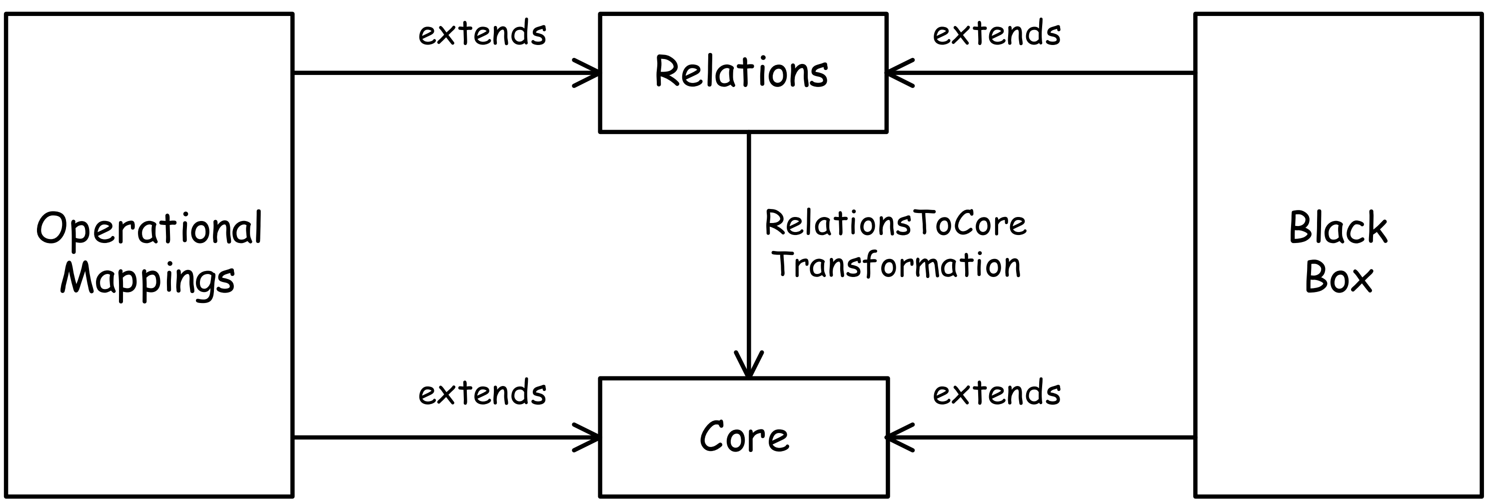
Architectural Alignment of ATL and QVT

- Architecture: set of concepts and relations
- Alignment for the major components of ATL and QVT;
- Major goal:
  - Achieving interoperability between ATL and QVT

QVT Architecture

Contains three DSLs that form layers:

- Relations (declarative)
- Core (declarative, simpler than Relations)
- Operational Mappings (imperative)



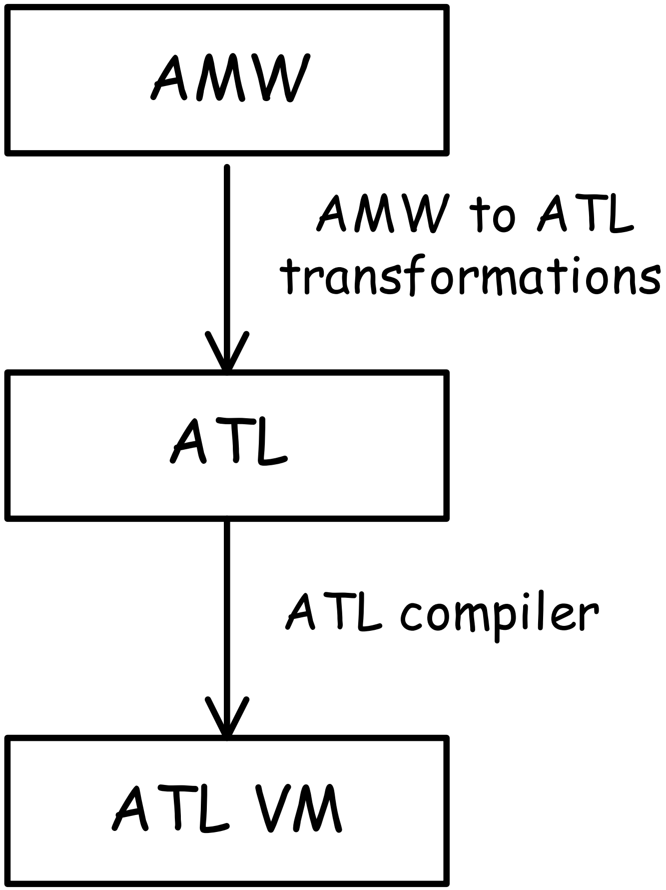
Conformance Points for QVT Tools

	Interoperability Dimension				
Language Dimension		Syntax Executable	XMI Executable	Syntax Exportable	XMI Exportable
	Core				
	Relations				
	Operational Mappings				

Note that QVT conformance is defined for tools. The term "QVT compliant language" is not defined in the spec.

ATL Architecture

Three-level architecture:



ATL Components

- AMW (ATLAS Model Weaver): Generic metamodel for establishing links among model elements (metamodel for model weaving)
  - Tool for defining domain specific transformation languages
- ATL: hybrid transformation language
  - Semantics of AMW extensions may be defined by ATL transformations
- ATL VM: virtual machine for executing model transformations

Abstractness

ATL and QVT Component Alignment (1)

Category		ATL	QVT
Abstraction Level of Transformation Specification		AMW	
			Relations
		ATL	Core, Operational Mappings
		VM	
Transformation Scenarios	Model synchronization	Via a separate transformation	Relations, Core
	Conformance checking	Via a separate transformation	Relations, Core
	Model transformation	AMW, ATL, VM	Relations, Core, Operational Mappings

ATL and QVT Alignment (2)

Category		ATL	QVT
Paradigm	Declarative	AMW	Relations, Core
	Hybrid	ATL	
	Imperative	VM	Operational Mappings
Directionality	Multidirectional	AMW	Relations, Core
	Unidirectional	ATL, VM	Operational Mappings
Cardinality	M-to-N	ATL, AMW, VM	Operational Mappings, Relations and Core (in checkonly mode)
	M-to-1		Relations and Core (in enforce mode)
Traceability	Automatic	ATL	Relations, Operational Mappings
	User-specified	VM	Core
In-place Update		ATL (in refining mode), VM	Relations, Core, Operational Mappings

ATL and QVT Interoperability (1)

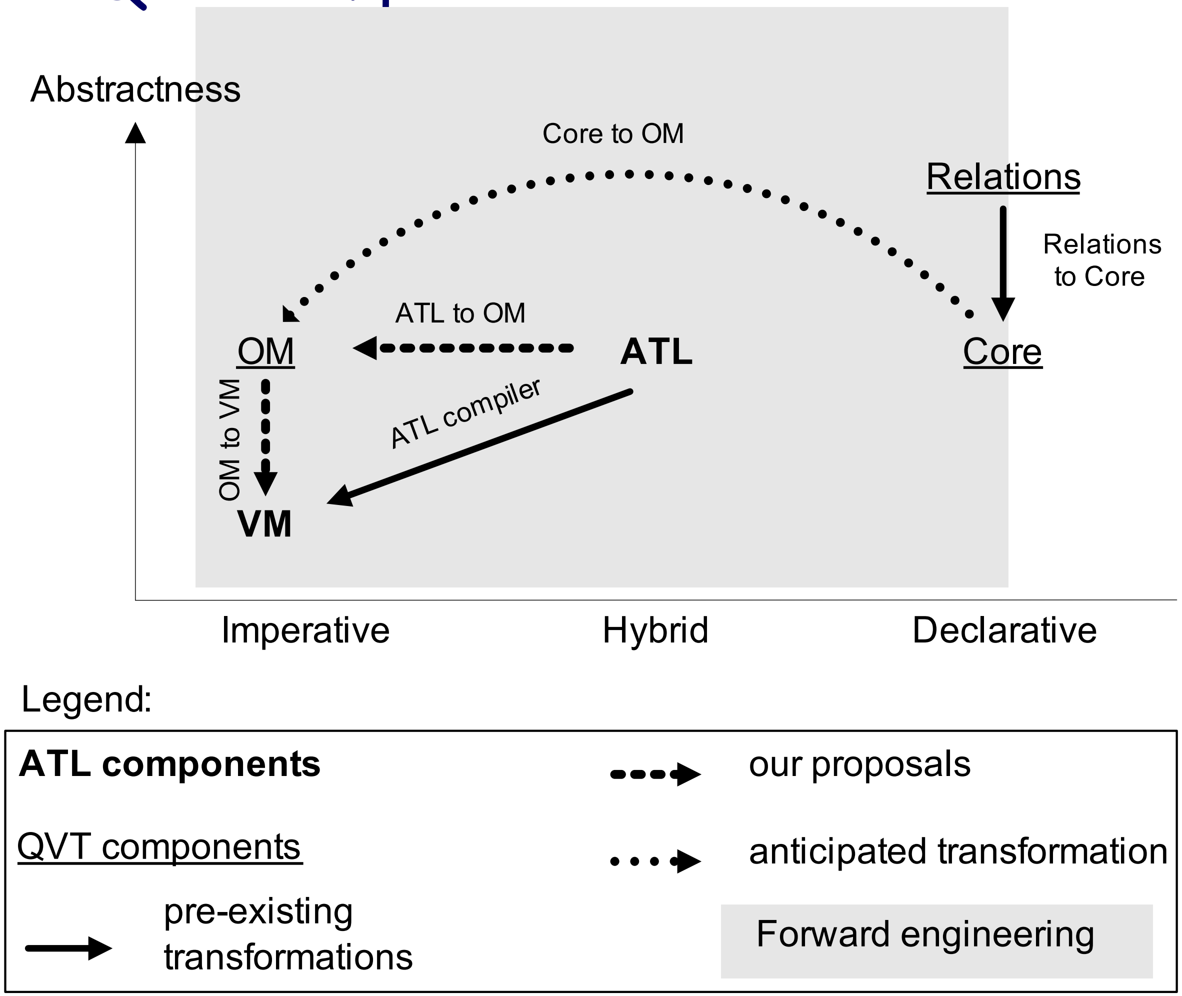
Interoperability: executing programs written in one language with the tools designed for another language.

Motivations:

- Execution and support tools
  - Assume you have a wonderful language without engine and a not so nice language with execution engine;
- Compliance to standards
  - If QVT programs run on the ATL VM then the ATL VM is QVT conformant!

ATL and QVT Interoperability (2)

Framework for reasoning about interoperability among ATL and QVT components



ATL and QVT Interoperability (3)

to / from	**VM**	OM	**ATL**	Core	Relations
VM	N/A	Reverse engineering			
OM	OM-to-VM	N/A	issues		
ATL	ATL compiler	ATL-to-OM	N/A		
Core		Core-to-OM		N/A	
Relations				Relations-to-Core	N/A

Conclusions

- ATL and QVT have common features but their problem domains are different:
  - QVT is mainly for software development;
  - ATL aims at solving data engineering transformation problems
- Architectural alignment between ATL and QVT shows that language interoperability is feasible via model transformations
- ATL tools may be called QVT conformant after the required transformations are provided
- It is interesting to generalize the framework for reasoning on ATL and QVT to other languages