Automatic transformation from QVT-R to C# for Enterprise Architect

CDL-Flex wrap-up @LieberLieber

Erwan Bousse

Business Informatics Group (BIG), TU Wien

December 13, 2016

Outline

Outline

QVT-R model transformation language

- Declarative language by the OMG
- Based on relations between source and target patterns
- Intuitive concrete syntax

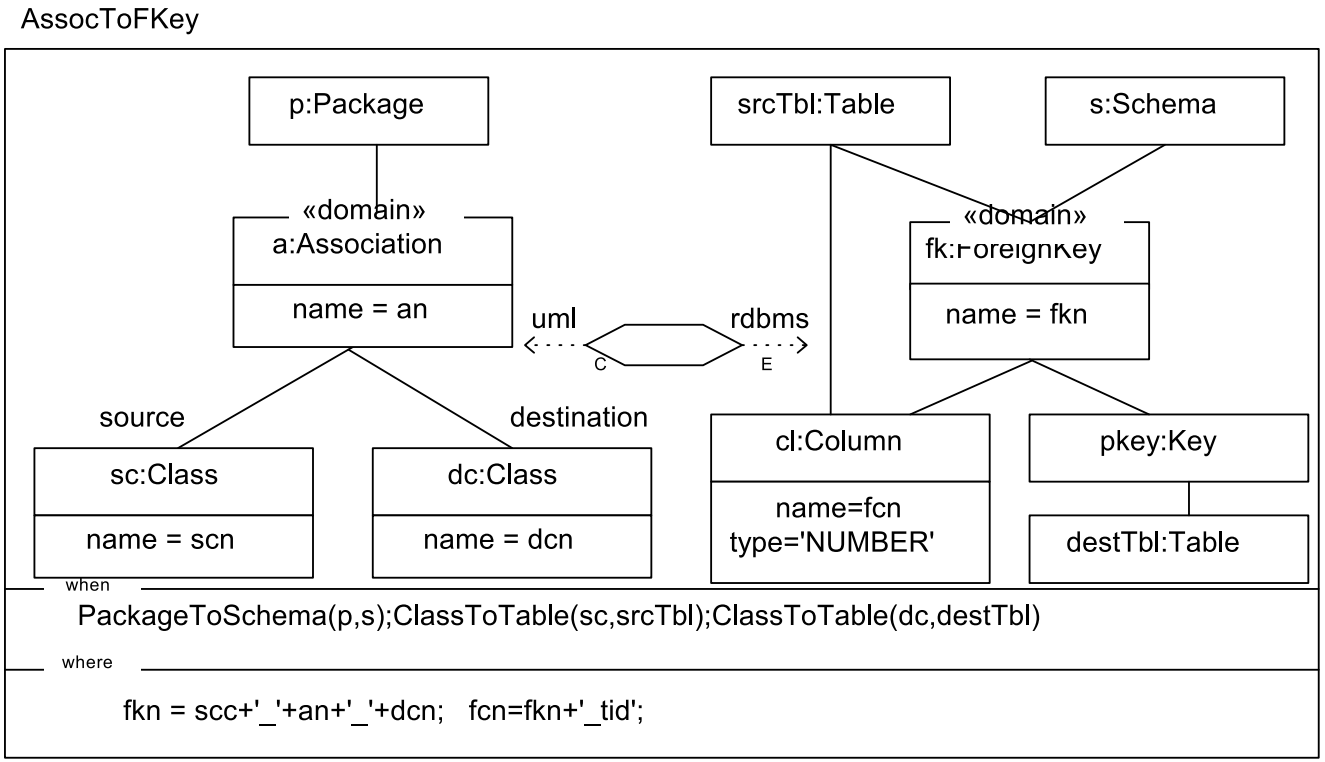

Figure A.8 - AssocToFKey relation

Problem

- QVT-R transformation = a set of declarative constraints, ie. expresses the *what* and not the *how*
- Must be analyzed to determine which sequence of actions (ie. the how) must be done to create the target model
- Such analyses must be done at each execution, ie. significant overhead in the interpreter

Idea

Move from an *interpreter* to a *compiler*, ie. **automatically generate C# code** from QVT-R transformations.

Additional requirements/constraints

Requirements stated by Lieberlieber:

- Top relations are called manually, by the modeler (ie. no pattern matching or scheduling required)
- Metamodel interface: possibility to choose how the output model is constructed (eg. as EA elements, or pure C# objects)

Outline

Technologies used

- **Enterprise Architect (EA)**: modeling workbench
- **C#**: .NET programming language
- **T4**: .NET template language
- **Roslyn**: C# parser, to obtain an AST from C# code
- **.NET Modeling Framework (NMF)**: powerful library to manipulate models in C# and to interoperate with the Eclipse Modeling Framework (EMF), ie with the Ecore language
- **Ecore**: metamodeling language of the EMF

Challenge: OCL

- OCL hard to manage (large, complex semantics)
- Used in:
  - Helper functions
  - When and where clauses

Solution

- Use of C# intead of OCL, as expressions in the transformation
- Use of Roslyn to obtain a C# AST

Overview (diagram)

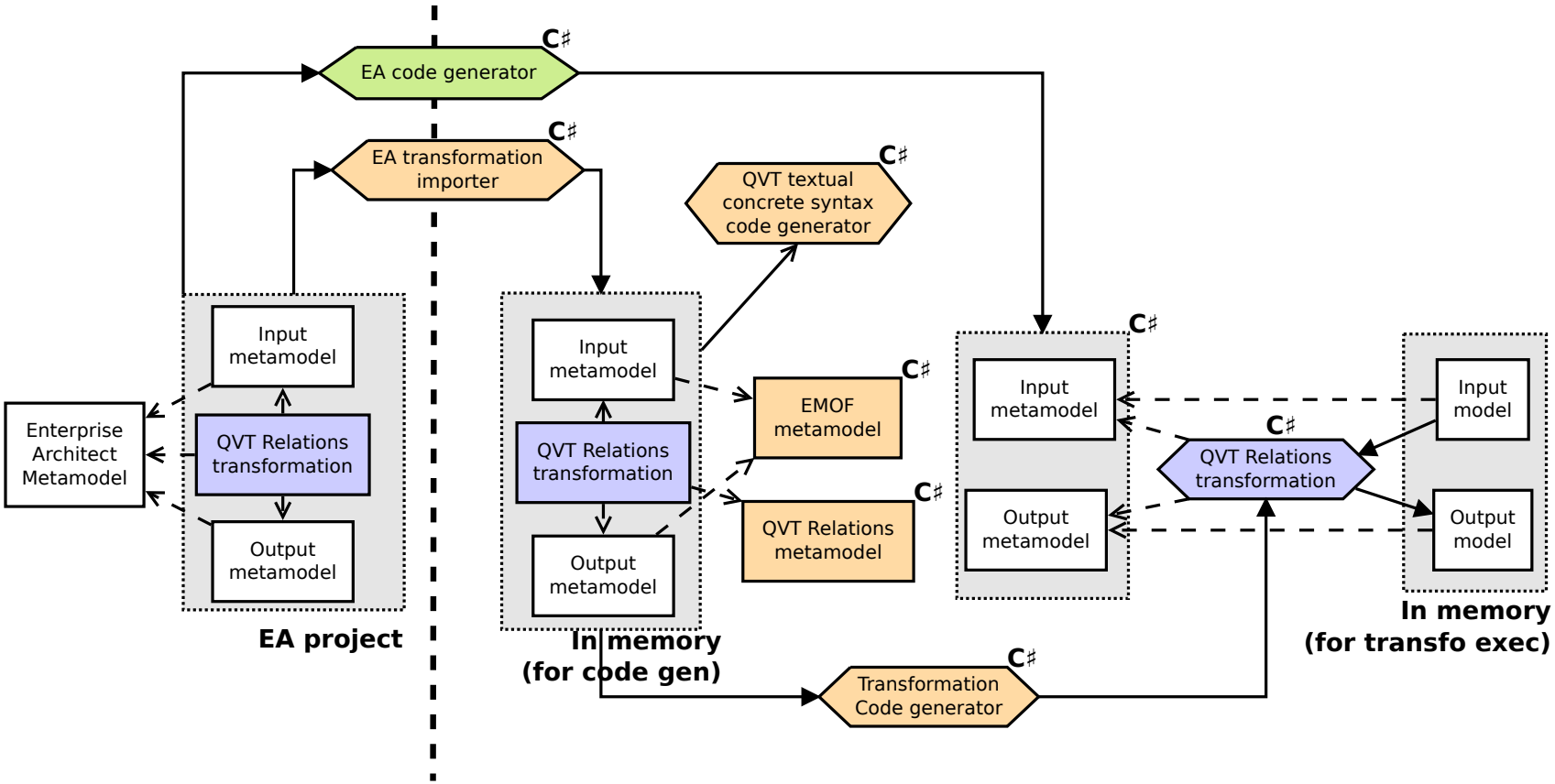

Figure 1: Architecture

Overview (steps)

- Architecture based on independent and reusable components
- Main steps:

0. Translation of `qvt.ecore` to C#, using NMF
1. Model-to-model transformation from Enterprise Architect class diagram to EMOF, for both source and target metamodels
2. Model-to-model transformation from QVT-R model in Enterprise Architect to QVT-R model conforming to the official QVT-R standard (slightly extended)
3. Model-to-text transformation from QVT-R to C#, ie. code generation