

# **Modelltransformation mit der QVT Relationssprache - Fallstudie einer werkzeugspezifischen Realisierung**

---

## **Model transformations with QVT relational language - Case study of a tool specific realization**

Oliver Alt

Oliver.Alt2@de.bosch.com



## Introduction



- Development and test of car multimedia devices at Bosch/Blaupunkt
  - Car Radio Systems
  - Navigation Systems
  - Car Multimedia devices and components



- Increasing part of software to realize new functions

- Internet connection
- Rear seat entertainment
- ...



- Increasing complexity

- To ensure the quality and handle the complexity, model-driven development and model-based testing concepts are developed



## Tools and languages in model-driven system development

- For the model-driven development (MDD)
  - UML2
  - SysML (Systems Modeling Language)
  - further domain specific extensions (UML2 profiles)are used.



- As modeling tool Sparx Systems Enterprise Architect is used.



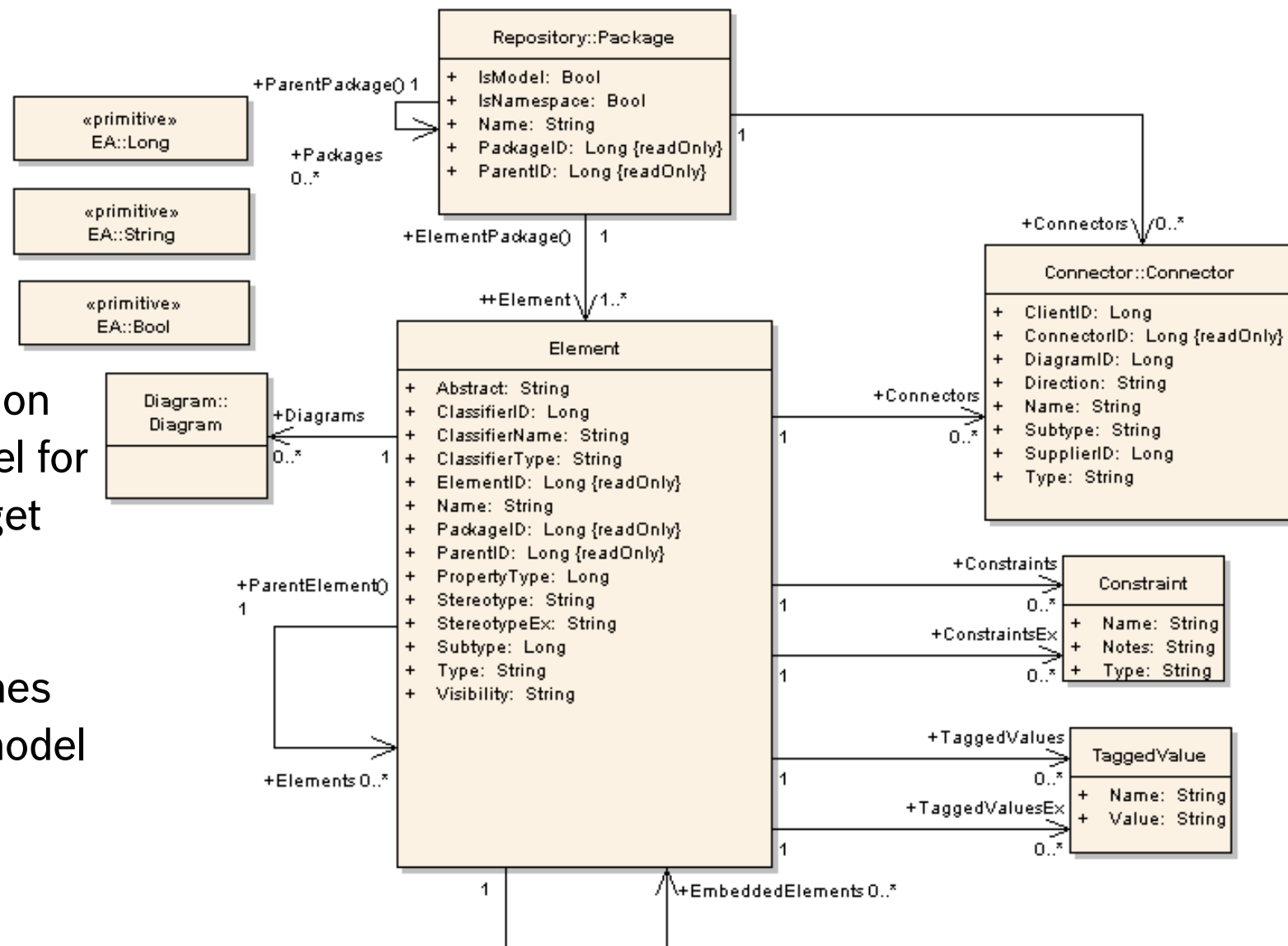
- Model transformations are required to transform the models to other artifacts
  - Other UML-models, XML, Code, ...

# Ways to describe model transformations

- By program code (VB, C#, Java,...)
- By using specific model transformation languages
  - ATL (Atlas Transformation Language)
  - QVT (Query/View/Transformation)
- QVT is a new OMG standard to describe model transformations
- Works with MOF meta-models
- QVT brings a declarative way to describe transformations – the relational language
  - Graphical syntax based on object diagrams
- QVT relational language was chosen as model transformation language

## How QVT works: Meta-models

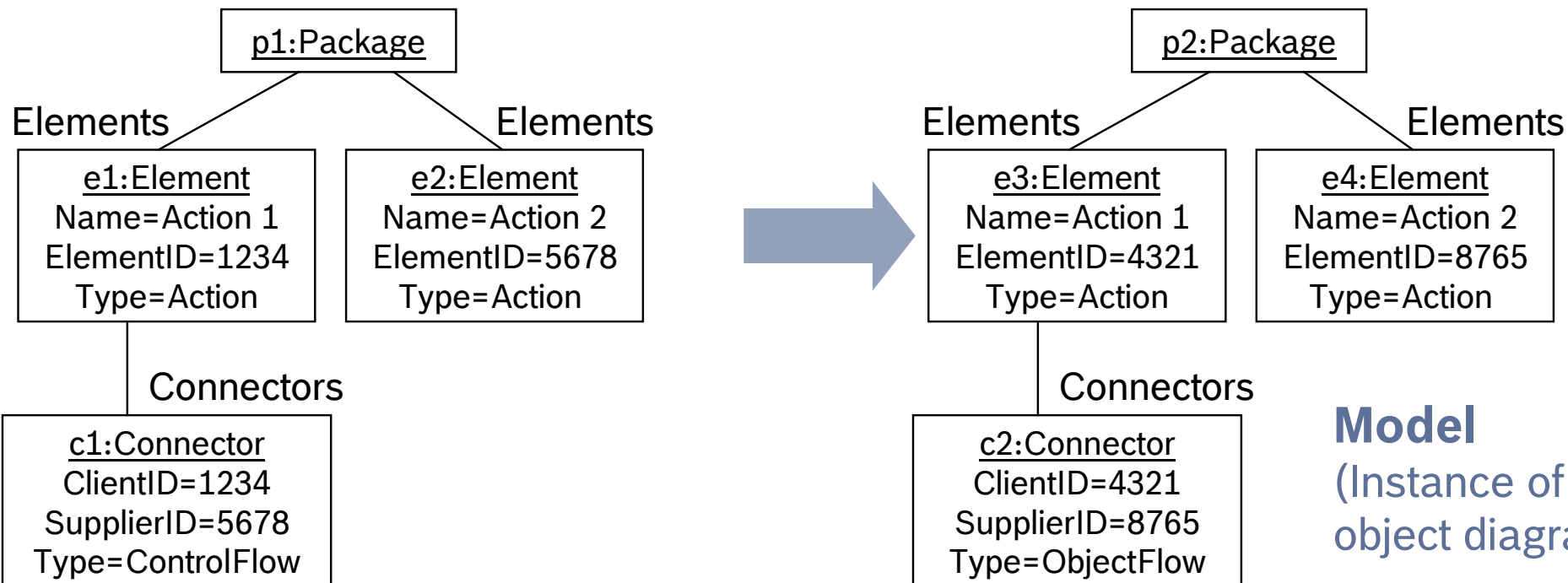
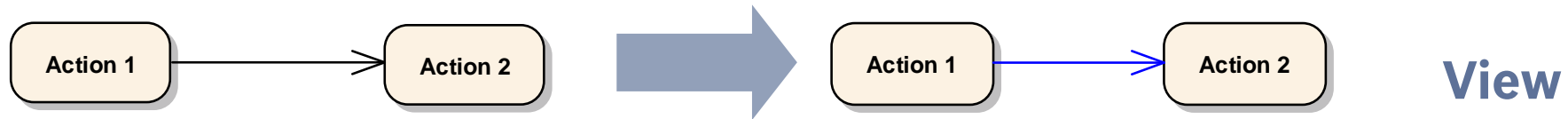
- A QVT transformation needs a meta-model for the source and target model
- A meta-model defines the structure of a model (model of a model)



Meta-model for Enterprise Architect UML tool

## How QVT works: Example

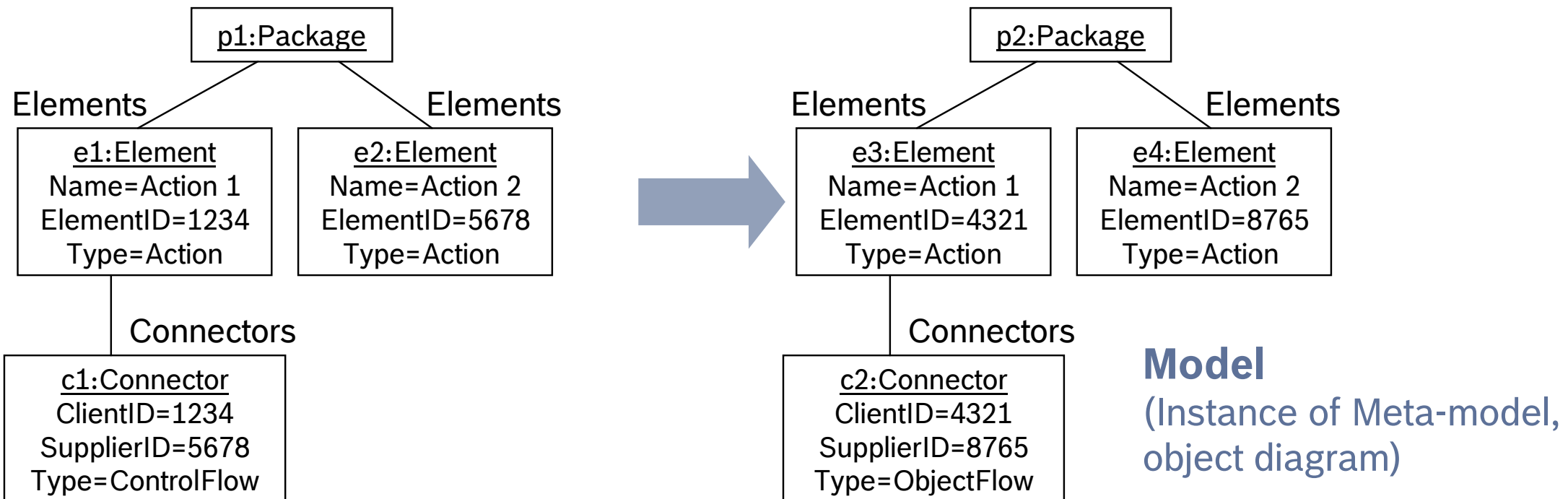
- Control flows between actions in source model should be transformed to object flows and actions with same names in target model



**Model**  
(Instance of Meta-model,  
object diagram)

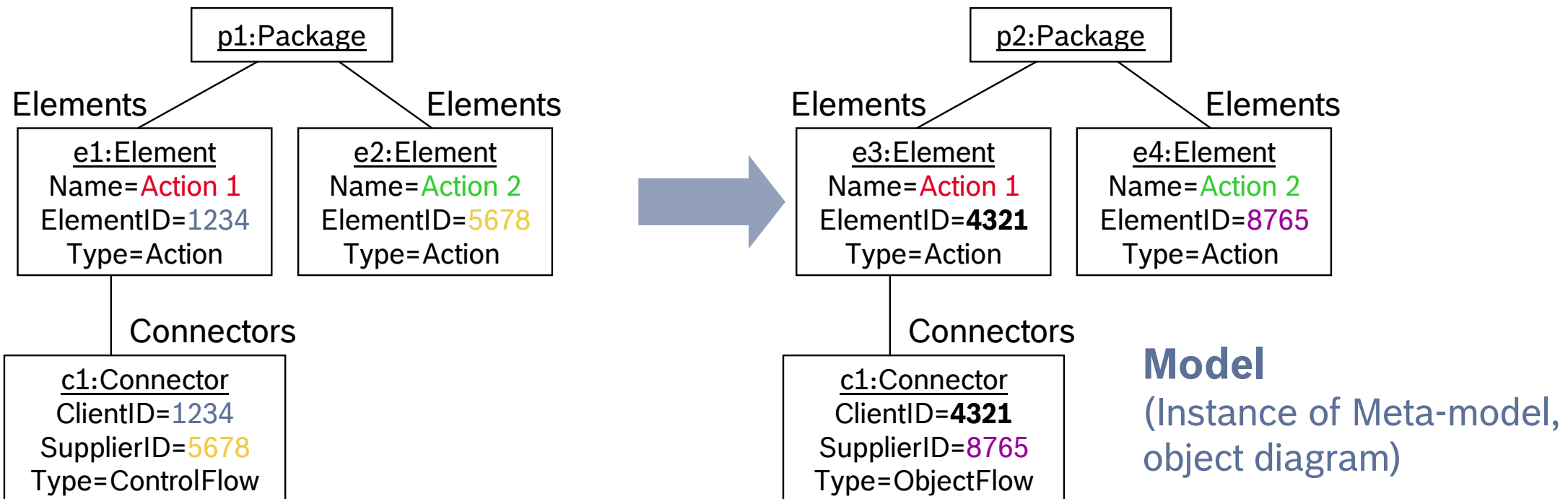
## How QVT works: Example

- QVT uses these object diagrams as templates and sets source and target model in relation
- Concrete values of objects are replaced by constants and variables to transform values between source and target model



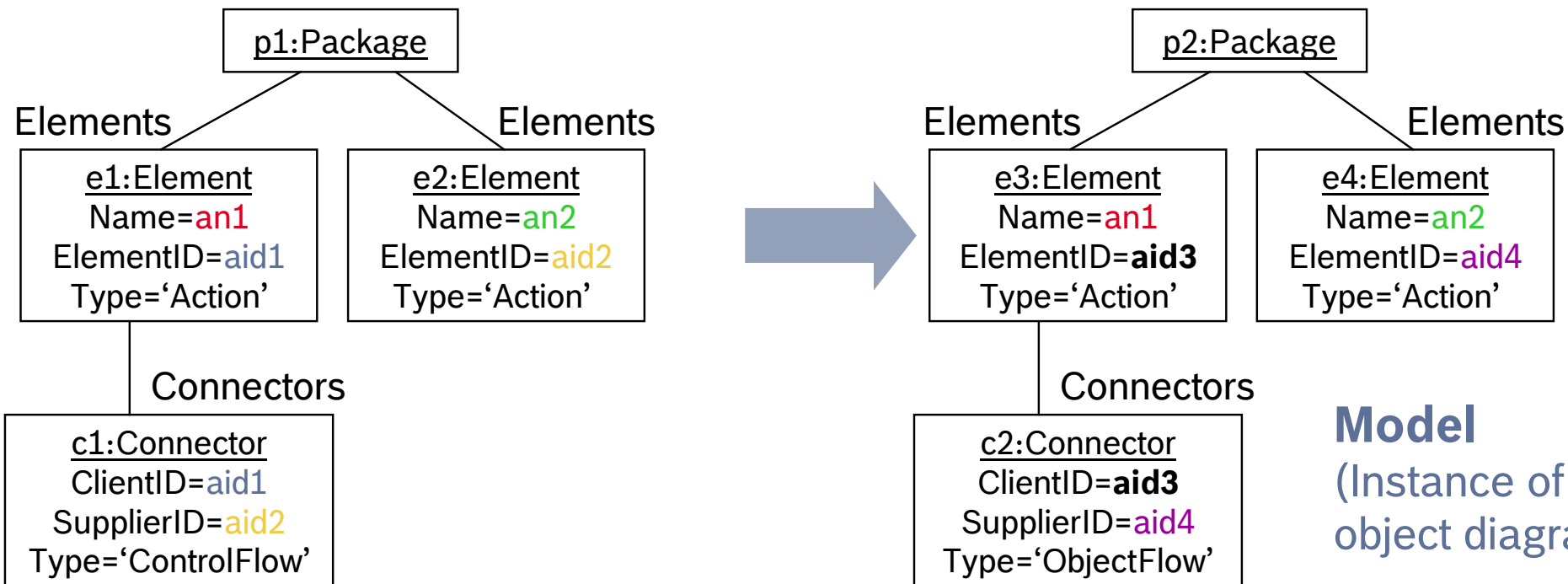
## How QVT works: Example

- QVT uses these object diagrams as templates and sets source and target model in relation
- Concrete values of objects are replaced by constants and variables to transform values between source and target model



## How QVT works: Example

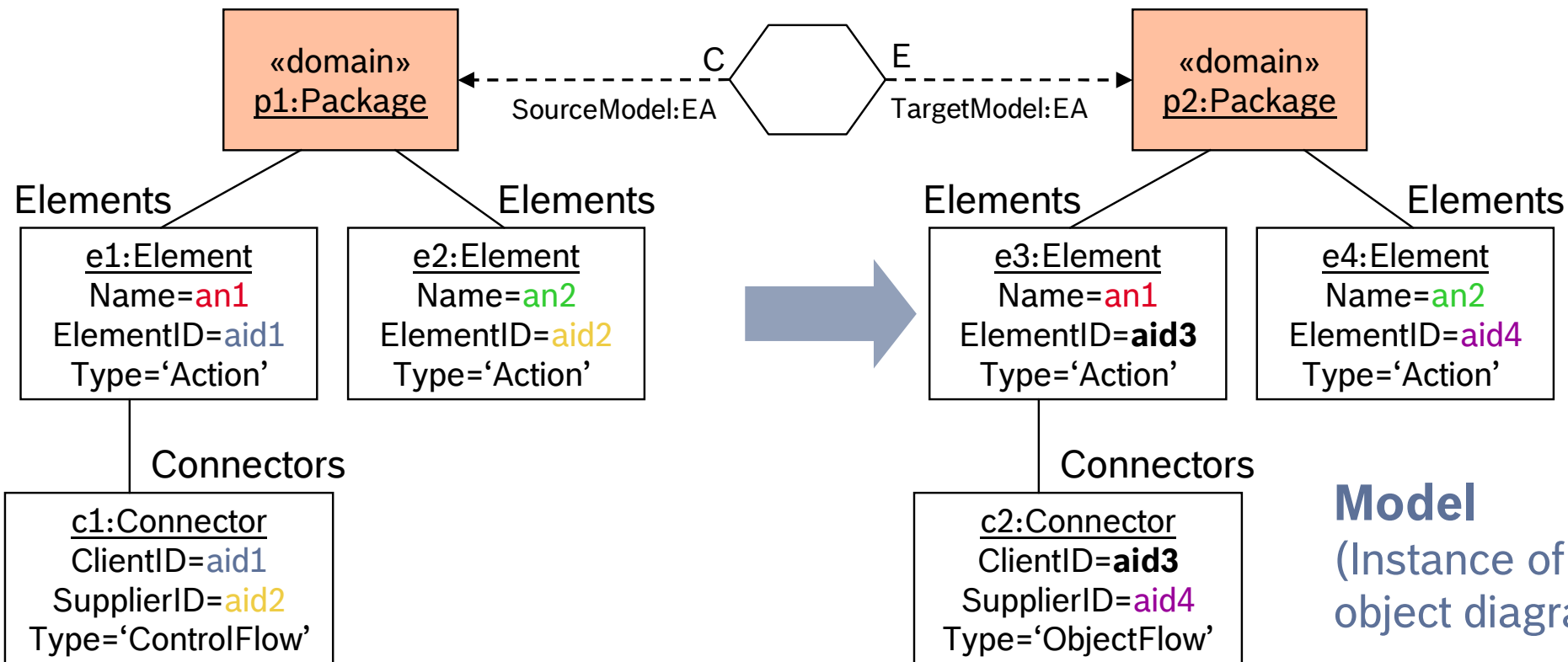
- QVT starts the mapping using **templates** and sets source and target model in relation
- The starting objects of objects are replaced by **constants** and variables to transform values between source and target model
- A new element is used as a relation symbol (⬡)



**Model**  
(Instance of Meta-model,  
object diagram)

## How QVT works: Example

- Now we have a declarative, graphical transformation rule in QVT relation language



**Model**  
(Instance of Meta-model,  
object diagram)

# Problems using QVT relations

- No existing implementation of QVT relations
- No existing editor for the graphical representation
- **So a case study was done to examine if Enterprise Architect (EA) can be extended with QVT relational support:**
  1. Is it possible to realize user friendly editing support in EA for QVT relations by using the meta-model information ?
  2. Is the generation of the textual QVT representation from graphical diagrams possible ?
  3. How hard is it to realize a QVT rule interpreter to execute the transformations within EA ?

## Results 1: QVT editing support (1)

- Enterprise Architect supports UML2 profiles
  - A QVT editor could be realized as such a profile
- New model elements, like the hexagonal relation node are realized using the EA shape script language

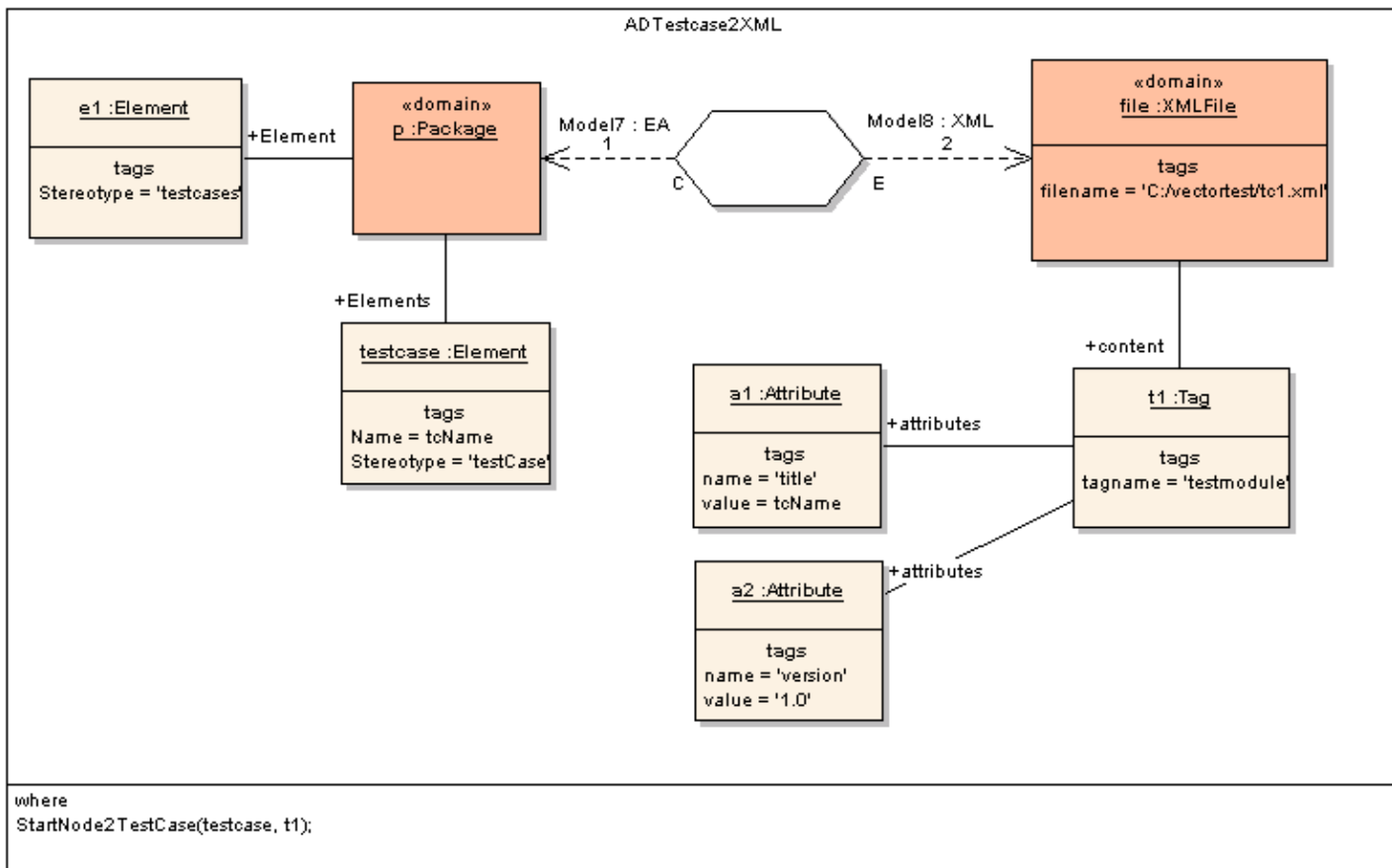
```
...
startpath();
  moveto(0, 50);
  lineto(20, 0);
  lineto(80, 0);
  lineto(100, 50);
  lineto(80, 100);
  lineto(20, 100);
  lineto(0, 50);
endpath();
setfillcolor(255,255,255);
fillandstrokepath();
...
```

Example EA shape script



Resulting  
shape

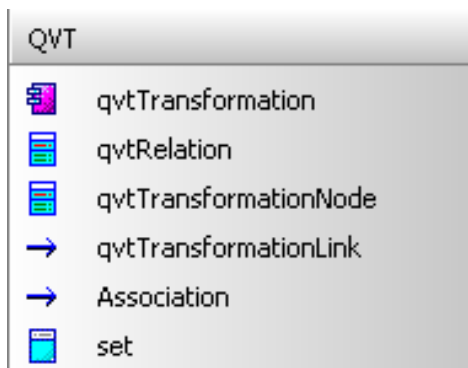
## Results 1: QVT editing support (2)



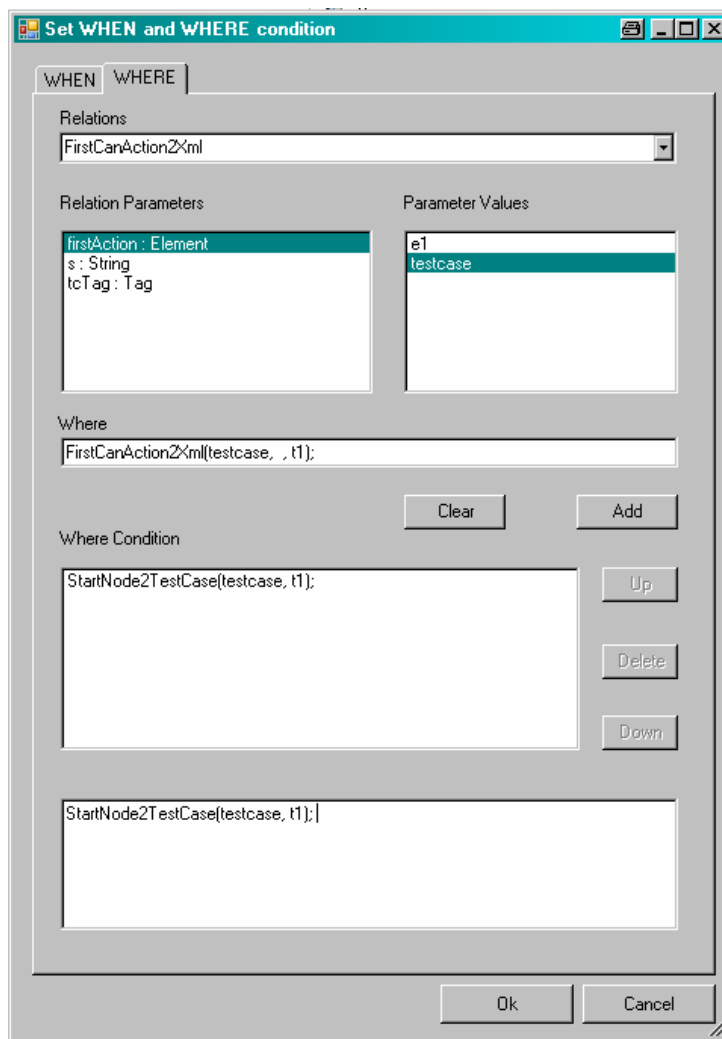
Example QVT Relation

- The QVT profile enables EA for QVT editing
- By using the EA automation interface user specific extensions can be realized (e.g. user friendly editing functions)
- Meta-model support (MOF 1.4) is included in EA

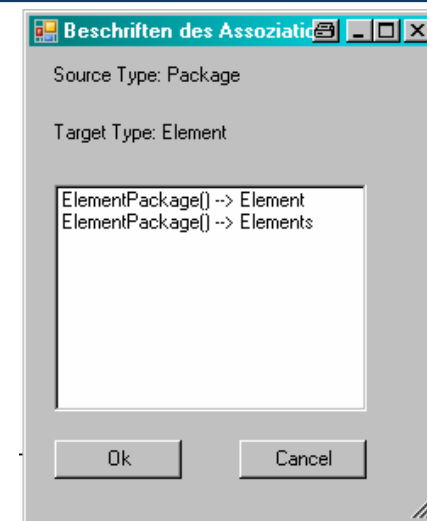
## Results 1: QVT editing support (3)



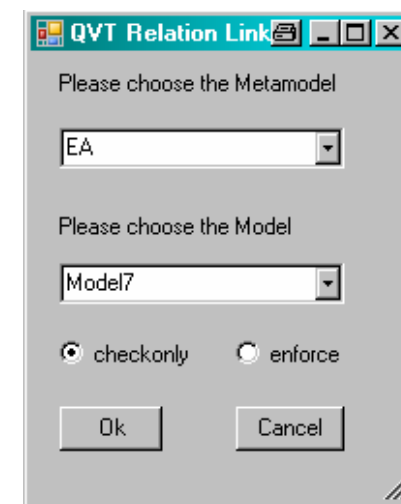
QVT toolbox



Dialog for WHEN and WHERE



Association selector



Model selector

# Result 2: Generating the textual representation

- A generator to derive the textual representation could be realized in two days by using the EA add-in interface
  - Implemented using C#.net
- The Add-in writes the textual representation to a file
- The import of existing textual QVT relations was not implemented, but it should be possible to realize it if a QVT text-parser is available
- By using the textual export the relations can be documented in a compact form

### Result 3: Realization of a QVT interpreter (1)

- A prototype of such an interpreter was realized within a few weeks
- The prototype is able to use an EA and a XML meta-model
- Advantages of the current implementation:
  - By using the reflection interface of .net all required information for accessing the EA model can be extracted directly from the meta-model on runtime (Collection and Attribute names)
- Disadvantages of the current implementation
  - Only EA to EA and EA to XML transformations are executable
  - Currently no complete support of the QVT standard (e.g. full OCL support) is included in the implementation
  - The implementation is not performance optimized

### Result 3: Realization of a QVT interpreter (2)

- The interpreter prototype supports a subset of transformations and allows the evaluation of these transformation rules
- In expectation of QVT transformation engines from research projects and industry in the near future, the further development of the prototype was stopped until further notice and work was invested to improve the QVT editor
- By using tool specific aspects the implementation of such interpreters could be simplified and a working transformation engine was realizable in short time

## Conclusion

- In a case study the implementation of the QVT relational standard with Enterprise Architect was analyzed
  
- As results
  - a working prototype of a graphical QVT editor
  - a generator to derive the textual representation
  - and a limited prototype of a QVT interpreter to evaluate the transformationscould be realized within Enterprise Architect within short time
  
- It is now possible to create QVT relations in practical use with a standard CASE tool - first for documentation purposes - and in future for executing required model-to-model transformations as well

Any questions...

