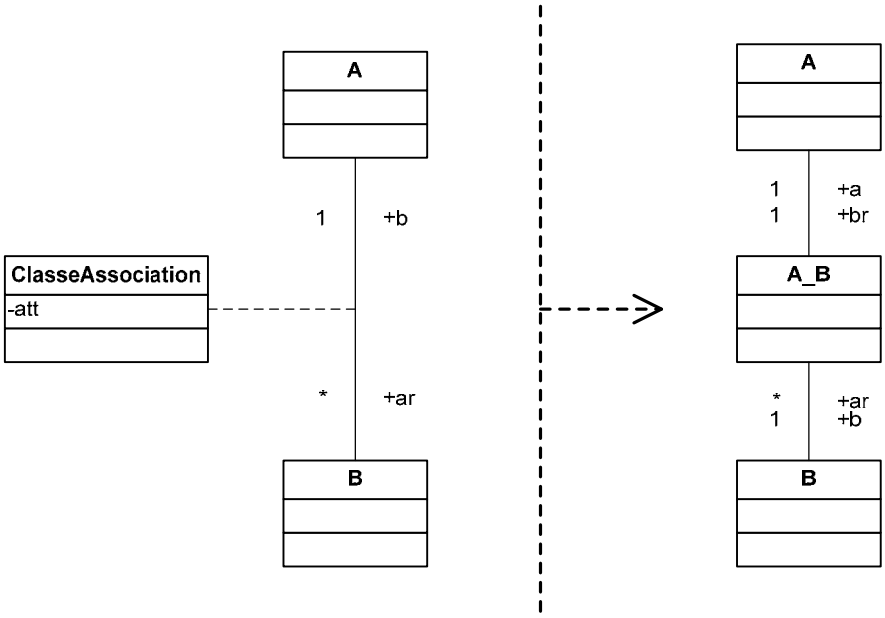
INRIA	**ATL Transformation** **Catalogue of Model Transformations**	**Author** **Eric Simon** **eric.simon3 <*at*> gmail.com**
	Documentation	Aug 7th 2006

1. ATL Transformation Example: Removal of association classes

This example is extract from Catalogue of Model Transformations by K. Lano.
Section 1.2: Removal of many-many associations, page 2.



2. ATL Transformation overview

2.1. Description

"This transformation replaces an association class with a new class and two associations."

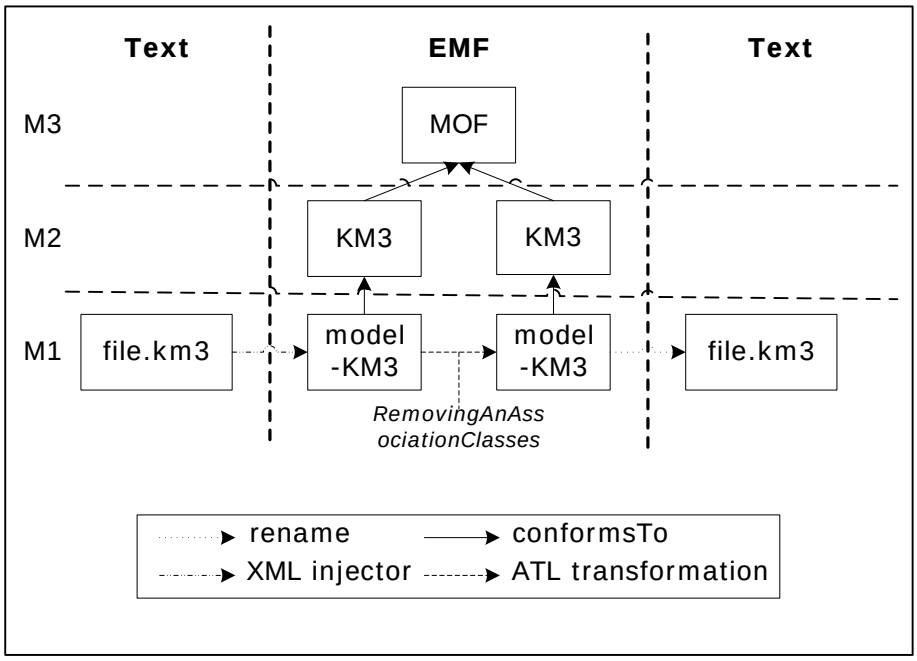


Fig 1. Overview of the transformation

2.2. Purpose

"Association classes cannot be represented directly in many object-oriented programming languages."

2.3. Rules specification

The transformation has the same metamodel for the source and the target: UML2.

- Rule Model: for each *Model* element, another *Model* element is created with the following elements:
  - the attribute *name* is the same,
  - the reference *ownedMember* is the same for the properties, but for the associations only those, which are not association classes.
- Rule DataType: for each *DataType* element, another *DataType* element is created with the following element:
  - the attribute name is the same.
- Rule LiteralNull: for each *LiteralNull* element, another *LiteralNull* element is created;
- Rule LiteralInteger: for each *LiteralInteger* element, another *LiteralInteger* element is created with the following element:
  - the attribute *value* is the same.
- Rule LiteralUnlimitedNatural: for each *LiteralUnlimitedNatural* element, another *LiteralUnlimitedNatural* element is created with the following element:
  - the attribute *value* is the same.

- Rule Property: a property is either an attribute, when it is in class, or an extremity of an association. For each *Property* element
  - If it is not an extremity of an association class
    - another *Property* element is created with the following elements:
      - the attribute *name* is the same,
      - the reference *type* is the same one as the source.
- Rule Class: for each *Class* element
  - If it is not an association class
    - another *Class* element is created with the following elements:
      - the attributes *name* and *isActive* are the same,
      - the references *ownedOperation*, *nestedClassifier*, *ownedReception* and *ownedAttribute* are the same one as the source.
- Rule AssociationClass2Class: for each Association*Class* element
  - a *Class* element is created with the elements (attributes and references) which it composes,
  - two associations which bind to both classes,
  - four properties which are the extremities of associations.

2.4. ATL Code

```
module RemovingAnAssociationClass; -- Module Template
create OUT : UML2 from IN : UML2;

2.4.1.  -- @begin Model
rule Model {
   from
      inputM : UML2!Model
   to
      outputM : UML2!Model (
         name <- inputM.name,
         ownedMember <- inputM.ownedMember,
         ownedMember <- inputM.ownedMember ->
         select(a | a.oclIsTypeOf(UML2!AssociationClass))->
         collect(c|Sequence {thisModule.resolveTemp(c, 'outputAsso1')})->flatten(),
         ownedMember <- inputM.ownedMember ->
         select(a | a.oclIsTypeOf(UML2!AssociationClass))->
         collect(c|Sequence {thisModule.resolveTemp(c, 'outputAsso2')})->flatten()
      )
}
-- @end Model

2.4.2.  -- @begin DataType
rule DataType {
   from
      inputC : UML2!DataType
   to
      outputC : UML2!DataType (
      name <- inputC.name
      )
}
-- @end DataType

2.4.3.  -- @begin LiteralNull
rule LiteralNull {
   from
      inputLN : UML2!LiteralNull
   to
      outputLN : UML2!LiteralNull
}
-- @end LiteralNull

2.4.4.  -- @begin LiteralInteger
rule LiteralInteger {
   from
      inputLI : UML2!LiteralInteger
   to
      outputLI : UML2!LiteralInteger (
         value <- inputLI.value
      )
}
-- @end LiteralInteger

2.4.5.  -- @begin LiteralUnlimitedNatural
rule LiteralUnlimitedNatural {
   from
      inputLUN : UML2!LiteralUnlimitedNatural
   to
      outputLUN : UML2!LiteralUnlimitedNatural (
         value <- inputLUN.value
      )
```

```
}
-- @end LiteralUnlimitedNatural

2.4.6.  -- @begin Property
rule Property {
   from
      inputC : UML2!Property
      (
         not (
            inputC.class_.oclIsTypeOf(UML2!AssociationClass)
         and
            inputC.association.oclIsTypeOf(UML2!AssociationClass)
            )
      )
   to
      outputC : UML2!Property (
         name <- inputC.name,
         type <- inputC.type
      )
}
-- @end Property

2.4.7.  -- @begin Class
rule Class {
   from
      inputC : UML2!Class
      (not inputC.oclIsTypeOf(UML2!AssociationClass))
   to
      outputC : UML2!Class (
         name <- inputC.name,
         ownedOperation <- inputC.ownedOperation,
         nestedClassifier <-  inputC.nestedClassifier,
         isActive <-  inputC.isActive,
         ownedReception <-  inputC.ownedReception,
         ownedAttribute <- inputC.ownedAttribute
      )
}
-- @end Class

-- @comment this rule replace a association class by a new class between two associations.
2.4.8.  -- @begin AssociationClass2Class
rule AssociationClass2Class {
   from
      inputA : UML2!AssociationClass
   to
      outputClass : UML2!Class (
         name <- inputA.name,
         ownedAttribute <- inputA.ownedAttribute->select(a|a.association->oclIsUndefined())
      ),
      outputDef1 : UML2! LiteralInteger (
         value <- 1
      ),
      outputPro11 : UML2!Property (
         owningAssociation <- outputAsso1,
         name <- inputA.memberEnd->at(1).name,
         upperValue <- inputA.memberEnd->at(1).upperValue,
         lowerValue <- inputA.memberEnd->at(1).lowerValue,
         defaultValue <- inputA.memberEnd->at(1).defaultValue
      ),
      outputPro12 : UML2!Property (
         owningAssociation <- outputAsso1,
         name <- inputA.memberEnd->at(2).type.name->toLower(),
         defaultValue <- outputDef1
      ),
```

```
        outputAsso1 : UML2!Association(
           memberEnd <- outputPro11,
           memberEnd <- outputPro12

        ),
        outputDef2 : UML2! LiteralInteger (
           value <- 1
        ),
        outputPro21 : UML2!Property (
           owningAssociation <- outputAsso2,
           name <- inputA.memberEnd->at(2).name,
           upperValue <- inputA.memberEnd->at(2).upperValue,
           lowerValue <- inputA.memberEnd->at(2).lowerValue,
           defaultValue <- inputA.memberEnd->at(2).defaultValue
        ),
        outputPro22 : UML2!Property (
           owningAssociation <- outputAsso2,
           name <- inputA.memberEnd->at(1).type.name->toLower(),
           defaultValue <- outputDef2
        ),
        outputAsso2 : UML2!Association(
           memberEnd <- outputPro21,
           memberEnd <- outputPro22
        )
}
-- @end  AssociationClass2Class
```

3. References

[1] Catalogue of Model Transformations
http://www.dcs.kcl.ac.uk/staff/kcl/tcat.pdf