



ATL Transformation

Catalogue of Model Transformations

Author

Baudry Julien
Jul.baudry <at> gmail.com

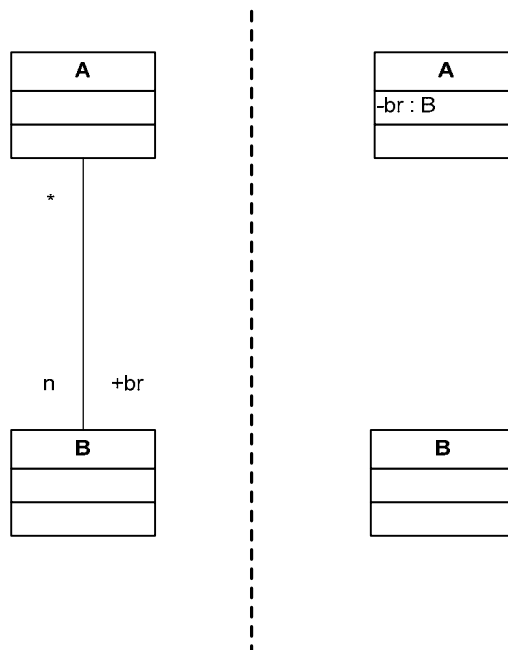
Documentation

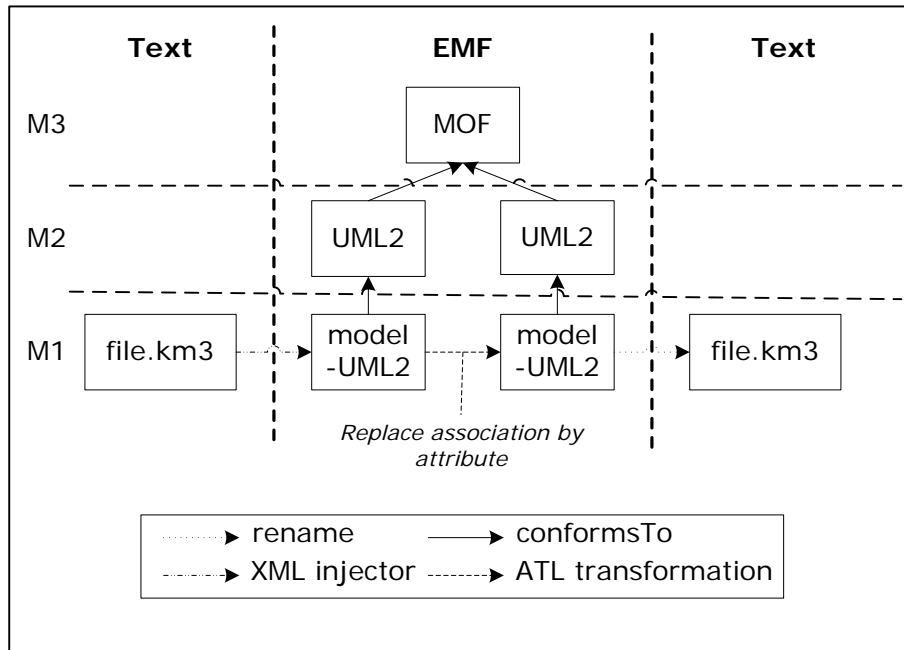
Aug 7th 2006

1.	ATL TRANSFORMATION EXAMPLE: REPLACE ASSOCIATION BY ATTRIBUTE	1
2.	ATL TRANSFORMATION OVERVIEW.....	2
2.1.	DESCRIPTION	2
2.2.	PURPOSE	2
2.3.	RULES SPECIFICATION	2
2.4.	ATL CODE.....	3
3.	REFERENCES	5

1. ATL Transformation Example: replace association by attribute

This example is extract from [Catalogue of Model Transformations](#) by K. Lano.
Section 1.5: replace association by attribute, page 4.





2. ATL Transformation overview

2.1. Description

IN UML 2.0, attributes of a class are equivalent to certain forms of association from the class. Embedding an association into a class as an attribute can be used as a step towards implementing it as an attribute of the class.


2.2. Purpose

The figure shows the equivalence of associations and attributes. Multiplicities, visibilities, the {ordered} and {unique} constraints, and the indications / that the feature is derived, can all be mapped from the association notation into identical attribute notation.

Qualified association cannot be represented as attributes, as there is no corresponding attribute notation, i.e., for specifying map data structures. In addition the multiplicity at the owning entity end of an association cannot be generally expressed in the attribute version. For an ordinary attribute this multiplicity is * and for an identity attribute (primary key) it is 0..1.

2.3. Rules specification

Our transformation has the same source and the target metamodel, UML2. We use 2 different names (UML2 and UML2target), but they refer to the same metamodel.

	ATL Transformation Catalogue of Model Transformations	Author Baudry Julien Jul.baudry <at> gmail.com
	Documentation	Aug 7th 2006

- For a Model element, another Model element is created :
 - with the same name, visibility and packageableElement_visibility,
 - Linked to the same ownedMember.
- For a Class element, another Class element is created :
 - with the same name, visibility and packageableElement_visibility,
 - with the same properties, isAbstract, isLeaf and isActive,
 - Linked to the same ownedAttribute.
- For a Property element, another Property element is created :
 - with the same name, visibility and packageableElement_visibility,
 - with the same properties, isDerived, isDerivedUnion, isLeaf, isOrdered, isReadOnly and isUnique,
 - Linked to the same ownedAttribute.
- For a LiteralNull/ LiteralInteger/ LiteralUnlimitedNatural element, another LiteralNull/ LiteralInteger/ LiteralUnlimitedNatural element is created :
 - with the same name and value.

To transform an association into an attribute, we only need to search from associations who refers to the current class, and to find the right property referring to the associated class. These properties become element of the class, as attributes. This is the job of the helper getProperties.


2.4. ATL Code

```
-- @name      Equivalence of attributes and associations
-- @version   1.0
-- @domains   Catalogue of Model Transformations
-- @authors   Baudry Julien (jul.baudry<at>gmail.com)
-- @date      2006/08/02
-- @description The purpose of this transformation is to replace association by equivalent
attributes.
-- @see http://www.dcs.kcl.ac.uk/staff/kcl/tcat.pdf
-- @see section 1.5, page 4
-- @see author of article : K. Lano

module UML2Transformations; -- Module Template
create OUT : UML2target from IN : UML2;

helper context UML2!Class def : getProperties : Sequence(UML2!Properties) =
    UML2!Association.allInstances()->select(a|a.endType->includes(self))
    ->iterate(a;acc : Sequence(UML2!Property) = Sequence {}|
        acc->including(a.ownedEnd->select(p|p.type <> self)
            ->first())
;

--@begin rule model
rule model {
    from
        inputModel : UML2!Model
    to
        outputModel : UML2target!Model (
            name <- inputModel.name,
```

	ATL Transformation Catalogue of Model Transformations	Author Baudry Julien Jul.baudry <at> gmail.com
	Documentation	Aug 7th 2006

```

visibility <- inputModel.visibility,
packageableElement_visibility <- inputModel.packageableElement_visibility,
ownedMember <-inputModel.ownedMember
)
}
--@end rule model


--@begin rule class
rule class {
  from
    inputClass : UML2!Class
  to
    outputClass : UML2target!Class (
      name <- inputClass.name,
      visibility <- inputClass.visibility,
      packageableElement_visibility <- inputClass.packageableElement_visibility,
      isAbstract <- inputClass.isAbstract,
      isLeaf <- inputClass.isLeaf,
      isActive <- inputClass.isActive,
      ownedAttribute <- inputClass.getProperties->union(inputClass.ownedAttribute)
    )
}
--@end rule class

--@begin rule property
rule property {
  from
    inputProperty : UML2!Property
  to
    outputProperty : UML2target!Property (
      isDerived <- inputProperty.isDerived,
      isDerivedUnion <- inputProperty.isDerivedUnion,
      isLeaf <- inputProperty.isLeaf,
      isOrdered <- inputProperty.isOrdered,
      isReadOnly <- inputProperty.isReadOnly,
      isStatic <- inputProperty.isStatic,
      isUnique <- inputProperty.isUnique,
      name <- inputProperty.name,
      visibility <- inputProperty.visibility,
      lowerValue <- inputProperty.lowerValue,
      upperValue <- inputProperty.upperValue
    )
}
--@end rule property

--@begin literal null
rule literalNull {
  from
    inputLiteral : UML2!LiteralNull
  to
    outputLiteral : UML2target!LiteralNull (
      name <- inputLiteral.name,
      value <- inputLiteral.value
    )
}
--@end literal null

--@begin literal integer
rule literalInteger {
  from
    inputLiteral : UML2!LiteralInteger
  to
    outputLiteral : UML2target!LiteralInteger (
      name <- inputLiteral.name,

```

	ATL Transformation Catalogue of Model Transformations	Author Baudry Julien Jul.baudry <at> gmail.com
	Documentation	Aug 7th 2006

```

        value <- inputLiteral.value
    )
}
--@end literal integer

--@begin literal unlimited natural
rule literalUnlimitedNatural {
    from
        inputLiteral : UML2!LiteralUnlimitedNatural
    to
        outputLiteral : UML2target!LiteralUnlimitedNatural (
            name <- inputLiteral.name,
            value <- inputLiteral.value
        )
}
--@end literal unlimited natural

```

3. References

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