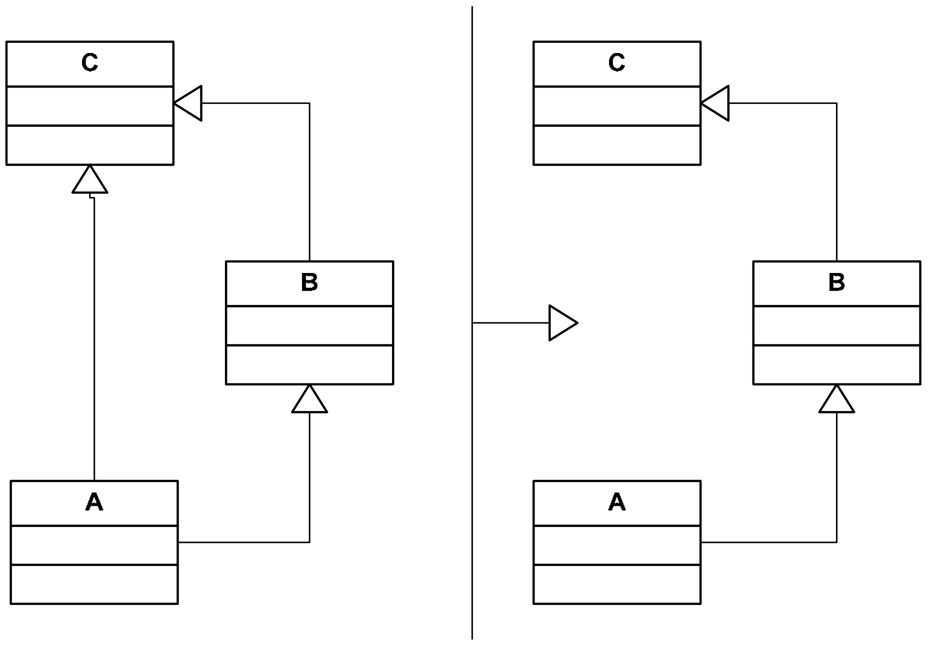
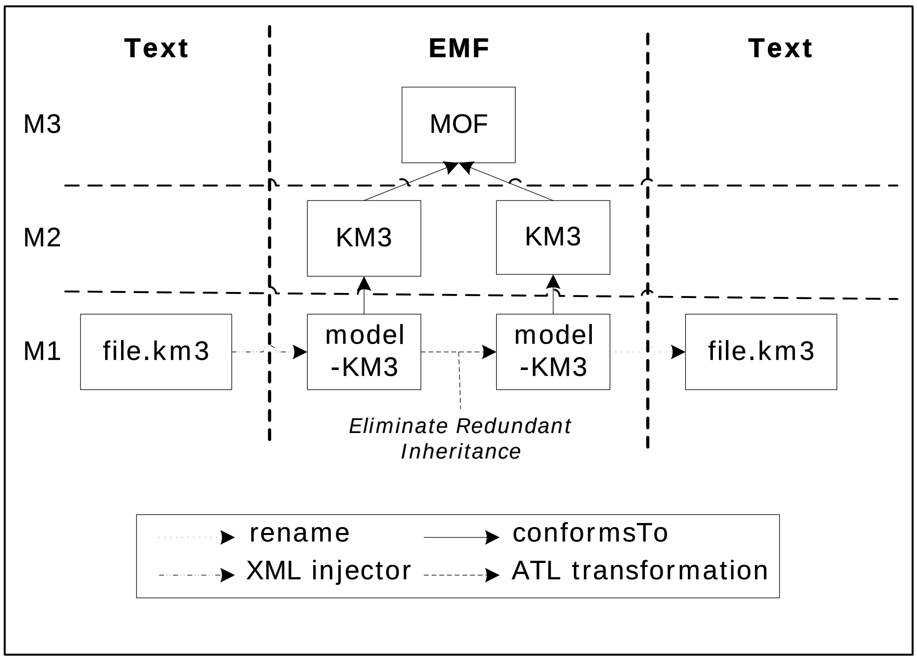
INRIA	**ATL Transformation** **Catalogue of Model Transformations**	**Author** **Baudry Julien** **Jul.baudry <*at*> gmail.com**
	Documentation	Aug 7th 2006

1. ATL Transformation Example: Eliminate Redundant Inheritance

This example is extract from Catalogue of Model Transformations by K. Lano.
Section 2.3: Eliminate Redundant Inheritance, page 15.





2. ATL Transformation overview

2.1. Description

If a class inherits another by two or more different paths of inheritance, remove all but one path, if possible.

2.2. Purpose

A redundant inheritance conveys no extra information, but complicates the model. The previous figure shows a typical situation where class A directly inherits from class C, and also indirectly via a more specific class B. The first inheritance is redundant and can be removed. (In some languages, such as Java, such inheritances would actually be invalid).

2.3. Rules specification

First of all, we must find a solution to detect a redundant inheritance. We use two helpers. The first one is *getAllSuperTypes ()*. The result is a sequence of class element, which corresponds to all the direct or indirect supertypes of the class. If a class inherit from another class by two different ways, it appears two times in the sequence.

The second helper *getSuperTypes ()* take all the supertypes and eliminate the element if it appears more than one time in the sequence. If it's the case, that means the class inherit from the class by another way.

Our transformation has the same source and the target metamodel, KM3. We use 2 different names (KM3 and KM3target), but they refer to the same metamodel.

- For a Metamodel element, another Metamodel element is created :
  - with the same name and location,
  - Linked to the same contents.

- For a Package element, another Package element is created :
  - with the same name,
  - Linked to the same contents.

- For a Class element, another Class is created :
  - with the same name,
  - Abstract if the source class is abstract,
  - Linked to the supertypes given by the helper *getSuperTypes ()*.

2.4. ATL Code

```
-- @name    Remove redundant inheritance
-- @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 remove redundant inheritance.
-- @see http://www.dcs.kcl.ac.uk/staff/kcl/tcat.pdf
-- @see section 2.3, page 15
-- @see author of article : K. Lano

module RedundantInheritance;
create OUT : KM3target from IN : KM3;

-- helper getSuperTypes
-- OUT : Sequence(KM3!Class)
-- Corresponds to all the direct or indirect supertypes of the class.
-- If a class inherit from another class by two different ways, it appears two times in the
sequence.
helper context KM3!Class def : getSuperTypes : Sequence(KM3!Class) =
   self.supertypes->iterate(a; acc:Sequence(KM3!Class)=Sequence{}|
                  if self.getAllSuperTypes->count(a)>1
                  then acc->union(Sequence{})
                  else acc->append(a)
                  endif
   );

-- helper getAllSuperTypes
helper context KM3!Class def: getAllSuperTypes : Sequence(KM3!Class) =
   if self.supertypes->isEmpty()
      then Sequence{}
   else     self.supertypes->select(c | c.supertypes->notEmpty())
               ->iterate(a; acc : Sequence(KM3!Class)=Sequence{} |(acc-
>including(a.getSuperTypes)))
               ->union(
               self.supertypes->iterate(a; acc : Sequence(KM3!Class)=Sequence{} | acc-
>including(a))
         ).flatten()
   endif;
```

```
--@begin rule Metamodel
rule Metamodel {
   from
      inputMm:KM3!Metamodel
   to
      outputMm:KM3target!Metamodel (
         location <- inputMm.location,
         contents <- inputMm.contents
      )
}
--@end rule Metamodel

--@begin rule Package
rule Package {
   from
      inputPkg:KM3!Package
   to
      outputPkg:KM3target!Package (
         name <- inputPkg.name,
         contents <- inputPkg.contents
      )
}
--@end rule Package

--@begin rule Class
rule Class {
   from
      inputA:KM3!Class
   to
      outputA:KM3target!Class (
         name <- inputA.name,
         isAbstract <- inputA.isAbstract,
         structuralFeatures <- inputA.structuralFeatures,
         supertypes <- inputA.getSuperTypes
      )
}

--@end rule Class
```

3. References

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