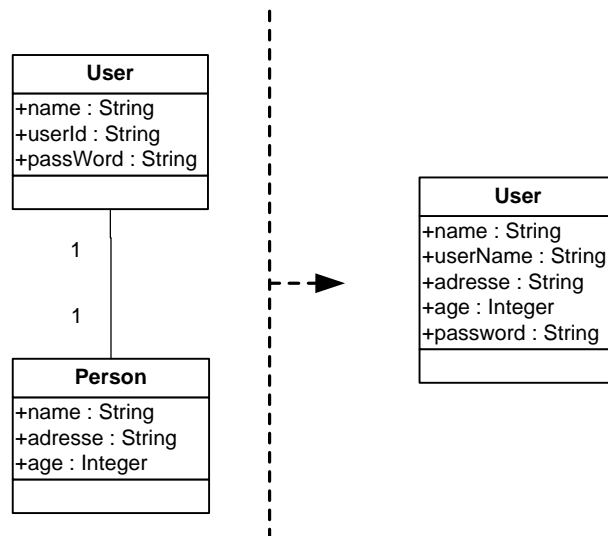
	ATL Transformation Catalogue of Model Transformations	Author Eric Simon eric.simon3 <at> gmail.com
	Documentation	Aug 9th 2006

1.	ATL TRANSFORMATION EXAMPLE: REMOVAL OF ASSOCIATION CLASSES	1
2.	ATL TRANSFORMATION OVERVIEW.....	1
2.1.	DESCRIPTION	1
2.2.	PURPOSE	2
2.3.	RULES SPECIFICATION	2
2.4.	ATL CODE.....	4
3.	REFERENCES	7

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

“Classes may define only part of a coherent concept, other parts may be expressed in different classes and their commonalities have not been recognised. This transformation merges such classes into a single class.”

	ATL Transformation Catalogue of Model Transformations	Author Eric Simon eric.simon3 <at> gmail.com
	Documentation	Aug 9th 2006

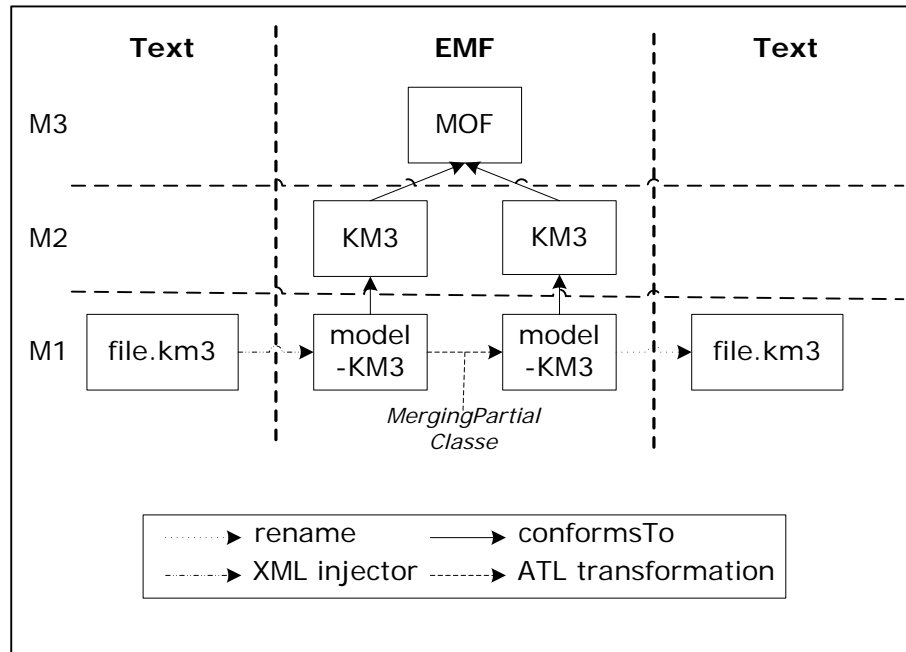


Fig 1. Overview of the transformation


2.2. Purpose

The purpose of this transformation is to merge the class link by a association one-one.


2.3. Rules specification

The transformation has the same metamodel for the source and the target: KM3.

- Rule [Metamodel](#): for each *Metamodel* element, another *Metamodel* element is created with the following elements:
 - the attribute *location* is the same,
 - the reference *contents* is the same.
- Rule [Package](#): for each *Package* element, another *Package* element is created with the following elements:
 - the attribute *name* is the same,
 - the reference *contents* is the same.
- Rule [DataType](#): for each *DataType* element, another *DataType* element is created with the following elements:
 - the attributes *name* and *location* are the same,
- Rule [EnumLiteral](#): for each *EnumLiteral* element, another *EnumLiteral* element is created with the following elements:
 - the attributes *name* and *location* are the same,
 - the references *enum* and *package* are composed by the same source.
- Rule [Enumeration](#): for each *Enumeration* element, another *Enumeration* element is created with the following elements:

	ATL Transformation Catalogue of Model Transformations	Author Eric Simon eric.simon3 <at> gmail.com
	Documentation	Aug 9th 2006

- the attributes *name* and *location* are the same,
 - the reference *literals* and *package* are composed by the same source.
- Rule [Class](#): for each *Class* element
 - If the *Class* element contained a reference which is not contained by a one-one association
 - another *Class* element is created with the following elements:
 - the attributes *name*, *location* and *isAbstract* are the same,
 - the references *structuralFeatures*, *supertypes* and *package* are the same.
- Rule [Attribute](#): for each *Attribute* element, another *Attribute* element is created with the following elements:
 - the attributes *name*, *lower*, *upper*, *isOrdered* and *isUnique* are the same source value,
 - the references *package*, *owner* and *type*, are filled in with the same value respectively.
- Rule [Reference](#): for each *Reference* element
 - If the *Reference* element is not contained by a one-one association
 - another *Reference* element is created with the following elements:
 - the attributes *name* and *isContainer* are the same,
 - the references *type*, *owner*, *opposite* and *package* are the same;
- Rule [Merging](#): for each pair of *Reference* element which is considered like a one-one association
 - a *Class* element is created with the following elements:
 - the elements of both *Class*, which are linked by this pair of *Reference*, composed this new *Class* element

	ATL Transformation Catalogue of Model Transformations	Author Eric Simon eric.simon3 <at> gmail.com
	Documentation	Aug 9th 2006

2.4. ATL Code

```

module MergingPartialClasses; -- Module Template
create OUT : KM3 from IN : KM3;

helper context KM3!Reference def: isMerginable : Boolean =
  self.lower = 1 and self.upper = 1 and not self.isContainer
  ;

helper def: assoMap : Map(KM3!Reference, Sequence(KM3!Reference)) = Map{};
rule isAlreadyConsidered(ref1 : KM3!Reference, ref2 : KM3!Reference) {

  do {
    if (not thisModule.assoMap.get(ref2).oclIsUndefined()) {
      if (thisModule.assoMap.get(ref2)->includes(ref1)) {
        true;
      }
      else {
        if (not thisModule.assoMap.get(ref1).oclIsUndefined()) {
          thisModule.assoMap <-
thisModule.assoMap.including(ref1,thisModule.assoMap.get(ref1)->including(ref2));
          false;
        }
        else {
          thisModule.assoMap <- thisModule.assoMap.including(ref1, Sequence{ref2});
          false;
        }
      }
    }
    else {
      if (not thisModule.assoMap.get(ref1).oclIsUndefined()) {
        thisModule.assoMap <-
thisModule.assoMap.including(ref1,thisModule.assoMap.get(ref1)->including(ref2));
        false;
      }
      else {
        thisModule.assoMap <- thisModule.assoMap.including(ref1, Sequence{ref2});
        false;
      }
    }
  }
}

```

2.4.1. --@begin rule Metamodel

```

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


```

2.4.2. --@begin rule Package

```

rule Package {
  from
    inputPkg:KM3!Package

```

	ATL Transformation Catalogue of Model Transformations	Author Eric Simon eric.simon3 <at> gmail.com
	Documentation	Aug 9th 2006

```

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

2.4.3. --@begin rule DataType
rule DataType {
  from
    inputData:KM3!DataType
  to
    outputData:KM3!DataType(
      name <- inputData.name,
      location <- inputData.location
    )
}
--@end rule DataType

2.4.4. --@begin rule EnumLiteral
rule EnumLiteral {
  from
    inputL:KM3!EnumLiteral
  to
    outputL:KM3!EnumLiteral (
      name <- inputL.name,
      location <- inputL.location,
      enum <- inputL.enum,
      package <- inputL.package
    )
}
--@end rule EnumLiteral

2.4.5. --@begin rule Enumeration
rule Enumeration {
  from
    inputEnum:KM3!Enumeration
  to
    outputEnum:KM3!Enumeration (
      name <- inputEnum.name,
      location <- inputEnum.location,
      package <- inputEnum.package,
      literals <- inputEnum.literals
    )
}
--@end rule Enumeration

2.4.6. --@begin rule Class
rule Class {
  from
    inputC:KM3!Class
    (not inputC.structuralFeatures->select(a|a.oclIsTypeOf(KM3!Reference))->exists(r|
r.isMerginable and r.opposite.isMerginable))
  to
    outputC:KM3!Class (
      isAbstract <- inputC.isAbstract,
      supertypes <- inputC.supertypes,
      name <- inputC.name,
      location <- inputC.location,
      package <- inputC.package,
      structuralFeatures <- inputC.structuralFeatures
    )
}

```



ATL Transformation

Catalogue of Model Transformations

Author

Eric Simon
eric.simon3 <at> gmail.com

Documentation


Aug 9th 2006

```
--@end rule Class

2.4.7. --@begin rule Attribute
rule Attribute {
  from
    inputAttr : KM3!Attribute
  to
    outputAttr : KM3!Attribute (
      package <- inputAttr.package,
      name <- inputAttr.name,
      lower <- inputAttr.lower,
      upper <- inputAttr.upper,
      isOrdered <- inputAttr.isOrdered,
      isUnique <- inputAttr.isUnique,
      owner <- inputAttr.owner,
      type <- inputAttr.type
    )
}
--@end rule Attribute

2.4.8. --@begin rule Reference
rule Reference {
  from
    inputRef : KM3!Reference
    ( not (inputRef.isMerginable and inputRef.opposite.isMerginable))
  to
    outputRef : KM3!Reference (
      package <- inputRef.package,
      name <- inputRef.name,
      lower <- inputRef.lower,
      upper <- inputRef.upper,
      isOrdered <- inputRef.isOrdered,
      isUnique <- inputRef.isUnique,
      owner <- inputRef.owner,
      type <- inputRef.type,
      isContainer <- inputRef.isContainer,
      opposite <- inputRef.opposite
    )
}
--@end rule Attribute

2.4.9. --@begin rule Merging
rule Merging {
  from
    inputA : KM3!Reference,
    inputB : KM3!Reference
    (
      inputA.opposite = inputB
      and inputA.isMerginable
      and inputB.isMerginable
      and inputA <> inputB
      and not thisModule.isAlreadyConsidered(inputA, inputB)
    )
  to
    outputA : KM3!Class (
      package <- inputA.owner.package,
      name <- inputA.owner.name->concat(inputB.owner.name),
      isAbstract <- inputA.owner.isAbstract,
      structuralFeatures <- inputA.owner.structuralFeatures-
>select(b|b.ocIsTypeOf(KM3!Reference))->select(a| not a.isMerginable),
      structuralFeatures <- inputA.owner.structuralFeatures-
>select(b|b.ocIsTypeOf(KM3!Attribute)),
```

	ATL Transformation Catalogue of Model Transformations	Author Eric Simon eric.simon3 <at> gmail.com
	Documentation	Aug 9th 2006

```

    structuralFeatures <- inputB.owner.structuralFeatures-
>select(b|b.ocliIsTypeOf(KM3!Reference))->select(a| not a.isMerginable),
    structuralFeatures <- inputB.owner.structuralFeatures-
>select(b|b.ocliIsTypeOf(KM3!Attribute))
)
}
--@end rule Merging

```

3. References

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