1. ATL Transformation Example: Disaggregation

This example is extract from Catalogue of Model Transformations by K. Lano. Section 2.8: Disaggregation, page 23.

2. ATL Transformation overview

2.1. Description

“A class is factored into component classes.”
2.2. Purpose

“A class may become large and unmanageable, with several loosely connected functionalities. It should be split into several classes, such as a master/controller class and helper classes, which have more coherent functionalities and data.”

2.3. Rules specification

The transformation has the same metamodel for the source and the target: KM3. However, we choose two different names: KM3 and KM3Target, indeed there is a confusion with the rule ocl: KM3!<nameElement>->allInstances() which returns all the class appertain to the source and the target.

- 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.
- 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.
- For each DataType element, another DataType element is created with the following elements:
  - the attributes name and location are the same,
• 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.

• For each `Enumeration` element, another `Enumeration` element is created with the following elements:
  - the attributes `name` and `location` are the same,
  - the reference `literals` and `package` are composed by the same source.

• For each `Class` element
  - another `Class` element is created with the following elements:
    - the attributes `name`, `location` and `isAbstract` are the same,
    - the references `supertypes` and `package` are the same one as the source,
    - the reference `structuralFeatures` owns the attribute which have not a metadata.
  - the `Class` elements contained by the set are created with the following elements:
    - the attributes `name`, `location` and `isAbstract` are the same,
    - the references `supertypes` and `package` are the same one as the source,
    - the reference `structuralFeatures` owns the attribute which have for metadata the name of this `Class` element.

• 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.

• For each `Reference` element, another `Reference` element is created with the following elements:
  - the attributes `name` and `isContainer` are the same,
  - the references `type`, `opposite`, `owner` and `package` are the same;
2.4. ATL Code

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

-- @comment this helper returns the metadata "commentsBefore" of name : <name>
helper context KM3!Attribute def: getMetadata(name : String) : String =
  let comment : String = self.commentsBefore->select(e | e.startsWith('-- @' + name + ' '))-
  > first() in
  if comment.oclIsUndefined() then
    OclUndefined
  else
    comment.substring(6 + name.size(), comment.size())
  endif;

-- @comment this helper returns the class set of the metadatas
helper context KM3!Class def: getClass : Set(String) =
  KM3!Attribute->allInstances()->select(c|c.commentsBefore->notEmpty())->iterate(a; acc : Set(String) = Set{}| acc->including(a.getMetadata('label'))) |

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
  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 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

2.4.5. --@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

2.4.6. --@begin rule Class
rule Class {
  from
    inputC:KM3!Class
  to
    outputC:KM3!Class {
      isAbstract <- inputC.isAbstract,
supertypes <- inputC.supertypes,
      name <- inputC.name,
      location <- inputC.location,
      package <- inputC.package,
      structuralFeatures <- inputC.structuralFeatures
    }
}
@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

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 Disaggregation

rule Disaggregation {
    from
    inputC : KM3!Class
    (inputC.structuralFeatures->select(r|r.oclIsTypeOf(KM3!Attribute))-
    >exists(a|a.commentsBefore->notEmpty()))
    using
    {
        subClasses : Set(String) = inputC.getClass;
    }
    to
    outputPrimaryClass : KM3!Class {
        isAbstract <- inputC.isAbstract,
        supertypes <- inputC.supertypes,
        name <- inputC.name,
        location <- inputC.location,
        package <- inputC.package,
        structuralFeatures <- inputC.structuralFeatures->select(a|not
        a.oclIsTypeOf(KM3!Attribute)),
        structuralFeatures <- inputC.structuralFeatures->select(a|
        a.oclIsTypeOf(KM3!Attribute))->select(a|a.commentsBefore->oclIsUndefined()),
        structuralFeatures <- subClasses->iterate(a; acc :
        Sequence(KM3!Reference)=Sequence{}| acc->append(thisModule.composition(inputC,a)))
    }
} -- @end rule Disaggregation

-- @comment this lazy rule creates a auxiliarie class and this link with the root element for each element in the metadata.
-- @begin rule composition
lazy rule composition{
    from
    inputC : KM3!Class,
    Name : String
    to
    outputRef1 : KM3!Reference {
        package <- inputC.package,
        name <- 'ref1'+Name,
        lower <- 1,
        upper <- 1,
        isOrdered <- false,
        isUnique <- false,
        owner <- inputC,
        type <- subClass,
        isContainer <- true,
        opposite <- outputRef2
    },
    subClass : KM3!Class {

isAbstract <- false,
name <- Name,
location <- inputC.location,
package <- inputC.package,
structuralFeatures <- inputC.structuralFeatures->select(a|a.oclIsTypeOf(KM3!Attribute))->select(a|a.commentsBefore->notEmpty() and a.getMetadata('label') = Name)

outputRef2 : KM3!Reference {
  package <- inputC.package,
  name <- 'ref'+inputC.name,
  lower <- 1,
  upper <- 1,
  isOrdered <- false,
  isUnique <- false,
  owner <- subClass,
  type <- inputC,
  isContainer <- false,
  opposite <- outputRef1
}

-- @end rule composition

3. References

[1] Catalogue of Model Transformations

http://www.dcs.kcl.ac.uk/staff/kcl/tcat.pdf