SINTEF

Lecture #5 (F5)
23 February 2006

Model mappings and transformations

Brian Elvesæter, SINTEF ICT
brian.elvesater@sintef.no

Based on material developed in ATHENA (IST-507849), INTEROP (IST-508011) and MODELWARE (IST-511731)

Structure

SINTEF

- Foundations of model mappings and transformations
- Model transformation technologies and examples
- Atlas Transformation Language (ATL) tutorial with RSM example
  - Installation
  - Book2Publish example
  - RSM
- References

Based on material developed in ATHENA (IST-507849), INTEROP (IST-508011) and MODELWARE (IST-511731)

SINTEF

Foundations of model mappings and transformations

About MOF

SINTEF

- MOF is short for Meta-Object Facility
- OMG standard
- MOF is constructed to store and manage metamodels and model instances of these.
- One can view MOF as a generic storage which lets you define a (meta)model of what to store, and MOF will manage storage of data according to this (meta)model.
- Is now used in various UML tools.
- MOF support for Java in the form of JMI (Java Metadata Interface)

MOF usage

- Core technology for flexible model repositories
  - A commonly used term for denoting a structured storage for information of various kinds.
- Define new metamodels in a standardized way.
- Base technology for different modelling tools (UML).

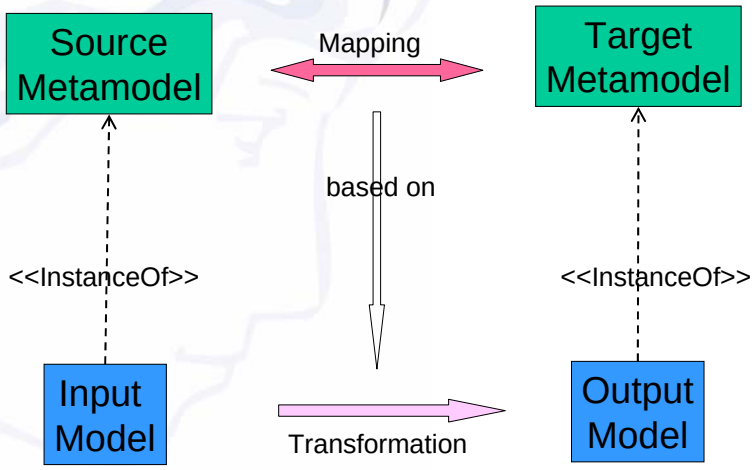
Model mapping (1/2)

- Mapping is performed by defining relations between two models
- The relations can be
  - 1-to-1, n-to-1, 1-to-n or n-to-n
- Mapping is performed in “design time”

Model mapping (2/2)

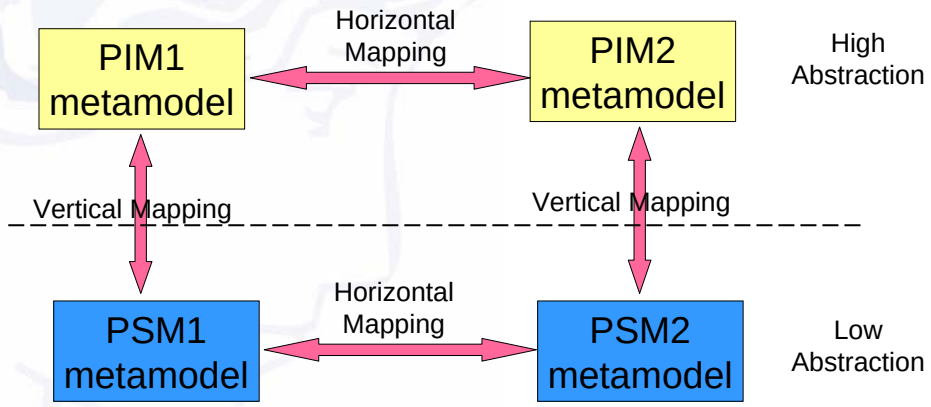
- The mapping is defined with models one meta-level higher than the input and output of the transformation.
- The mapping is used to perform transformation of instances of the mapped models.
- “The mapping describes the rules used for the transformation”.

Mapping and transformation



Different mappings

SINTEF



Transformations

SINTEF

- Takes input and produces output
  - One-way process
- Transforms according to a predefined mapping
- Transformation is used in “run time”
- Two main categories of transformation
  - Vertical transformation
  - Horizontal transformation

Horizontal transformation

SINTEF

- Source model has the same level of abstraction as target model
  - Not to be confused with “meta-levels”
- Examples of horizontal transformation
  - Refactoring
  - Merging

Vertical transformation

SINTEF

- Source model is at a different level of abstraction than the target model
- Examples of vertical transformation
  - Refinement (specialization)
    - PIM→PSM transformations
  - Abstraction (generalization)

MOF QVT (Query/View/Transformation)

SINTEF

- High-level
  - Definition of query/view/transformation language for MOF
  - Important part of realizing the OMG MDA vision
- Part of the MOF 2.0 specification work in OMG
  - MOF 2.0 Facility / Object Lifecycle RFP
  - MOF 2.0 IDL RFP MOF 2.0 Versioning RFP
  - MOF 2.0 Query / View / Transformation RFP
  - MOF 2.0 Core
- We focus on the transformation part
  - But we will need both the Q and the V for it to work

Involved partners

SINTEF

- Codagen
- Compuware
- DSTC
- France Telecom
- IBM
- INRIA
- Interactive Objects
- Kings College London
- Softteam
- Sun Microsystems
- Tata Consultancy Services
- Thales
- TNI-Valiosys
- University of Paris VI
- University of York

QVT - Query/View/Transformation

SINTEF

- Queries are performed on input models for finding specific elements or information
  - Example: *Return all classes*
- Views are models derived from other models
  - The result of a query is a kind of view
- Transformations takes a model as input and creates a new model
  - Example: PIM → PSM
  - Defined in Relations language or Core language
  - Uses Queries and Views

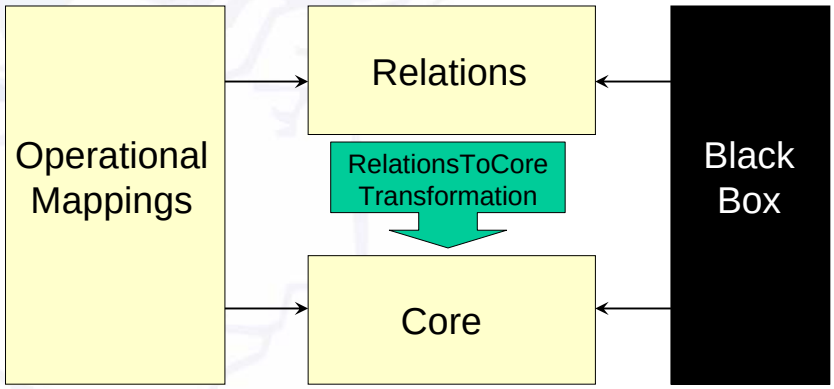
How to define a transformation?

SINTEF

- The abstract syntax must be defined as a metamodel in MOF
- Concrete syntax expressed as text (program code) or models
- Variations
  - Declarative
  - Imperative

QVT overview

SINTEF



QVT overview – Declarative part

SINTEF

- Relations
  - Declarative specification of the mapping between MOF models
  - Equivalent with Java code (high-level)
- Relations to Core transformation
  - Equivalent with compiling Java code into byte code
- Core
  - Equivalent with Java byte code (low-level)

QVT overview – Imperative part

SINTEF

- Operational mappings
  - Standard language for defining Relations (or Core) in an imperative way
- Black box
  - Offers the possibility to plug-in code from any programming language with a MOF binding

QVT in practice – most transformation

SINTEF

- Given a defined metamodel for source and target
  - Metamodels must be well-defined according to MOF (models on level M2)
- One can define transformations between these two worlds in a general, standardized manner
  - The transformation can be used on all instances of the source metamodel
  - The result will be an instance of the target metamodel

SINTEF

Model transformation technologies and examples

Technology overview

SINTEF

- Multiple technologies for mapping and transformation
  - Java
  - IBM Model Transformation Framework (MTF)
  - Atlas Transformation Language (ATL)
- Not many QVT-based ones
  - QVT support in Borland Together Architect 2006

Java

SINTEF

- Technologies such as MDR and EMF allow for generation of Java API toward arbitrary metamodels
- Using these APIs mappings can be defined in a Java program
- The transformation is performed by running the Java program on a model instance given as input
- Drawback
  - API generation for metamodels needed
- Pros
  - Well known language for mapping definition

IBM MTF

SINTEF

- MTF was developed in order to experiment with QVT related concepts
- MTF is not QVT compliant
- MTF rules describe the relationships between the input and the output metamodel
- Provides functionality to define mappings between metamodels
- And execute the model transformations

ATL

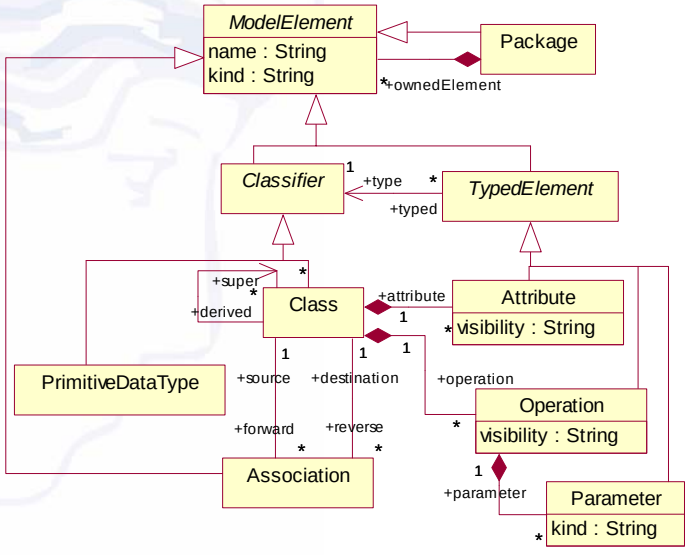
SINTEF

- The Atlas Transformation Language (ATL) is a hybrid language (a mix of declarative and imperative constructions) designed to express model transformations as described by the MDA approach.
- It is not QVT, but similar and with the corresponding functionality
- A transformation model in ATL is expressed as a set of transformation rules.
- The recommended style of programming is declarative.
- OCL is used to expression constraints on rules
  - Guards (constraints) on the entry point for a rule
- Different kinds of M3/M2 (meta)metamodel technology supported: Netbeans MDR and EMF Ecore
  - Can use either EMF or MDR metamodels as input and output.
- Can also be used to produce textual output.

Example: UML → RDBMS

SINTEF

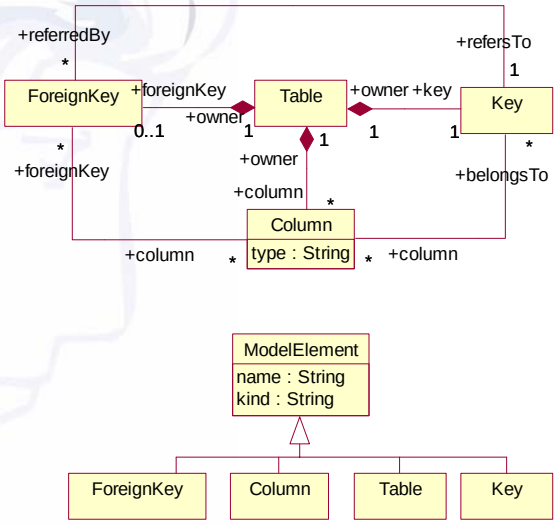
Parts of UML metamodel



Example: UML → RDBMS

SINTEF

Simple RDBMS metamodel



Example: Informal rules

SINTEF

- A persistent class maps to a table, a primary key and an identifying column.
- Attributes of the persistent class map to columns of the table: an attribute of a primitive datatype maps to a single column; an attribute of a complex data type maps to a set of columns corresponding to its exploded set of primitive datatype attributes; attributes inherited from the class hierarchy are also mapped to the columns of the table.
- An association between two persistent classes maps to a foreign key relationship between the corresponding tables.

Example: Java transformation (1/2)

SINTEF

```
public Model model2model(org.eclipse.uml2.Model inModel){
        Model retval = RdbmsFactory.eINSTANCE.createModel();
        Iterator it = inModel.getOwnedElements().iterator();
        while (it.hasNext()){
                Object element = it.next();
                if(element instanceof Class){
                        Class toProcess = (Class)element;
                        retval.getElements().add(class2table(toProcess));
                }
        }
        return retval;
}

public Table class2table(Class inClass){
        Table retval = RdbmsFactory.eINSTANCE.createTable();
        retval.setName(inClass.getName());
        Iterator it = inClass.getOwnedAttributes().iterator();
        while (it.hasNext()){
                Object attrib = it.next();
                if(attrib instanceof Property){
                        Property toProcess = (Property)attrib;
                        retval.getColumns().add(attrib2column(toProcess));
                }
        }
        return retval;
}
```


Example: Java transformation (2/2)

SINTEF

```
public Column attrib2column(Property inAttr){
        Column retval = RdbmsFactory.eINSTANCE.createColumn();
        retval.setName(inAttr.getName());
        retval.setType(inAttr.getType().getName());
        return retval;
}
```

- It is all about traversing the model tree
  - And building an output model

Based on material developed in ATHENA (IST-507849), INTEROP (IST-508011) and MODELWARE (IST-511731)

Example: MTF transformation

SINTEF

```
/* UML Packages map to EPackages */
relate umlpkg2ecore( uml:Package pkg, ecore:EPackage epkg)
   when equals(pkg.name, epkg.name)
{
        // map sub-packages
        umlpkg2ecore( over pkg.ownedMember, over epkg.eSubpackages),

        // map classifiers
        umlclassifier2ecore( over pkg.ownedMember, over epkg.eClassifiers )
}

/* UML Classifiers map to EClassifiers */
abstract relate umlclassifier2ecore( uml:Classifier class,
        ecore:EClassifier eclass)
   when equals(class.name, eclass.name)


/* Map UML Class to EClass */
relate umlclass2ecore extends umlclassifier2ecore( uml:Class class,
        ecore:EClass eclass)
{
        // check that super classes map to each other
   check umlclass2ecore(over class.superClass, over eclass.eSuperTypes)
}
```


Example: ATL transformation (1/2)

SINTEF

```
module uml2rdbms; -- Module Template
create OUT : rdbms from IN : uml2;

rule model{
  from inMod:uml2!Model
  to
    modi:rdbms!Model(
     elements <-inMod.ownedMember
  )
}

rule class2table{
  from cl:uml2!Class (cl.oclIsTypeOf(uml2!Class))
  to
    tbl:rdbms!Table(
    name <- cl.name,
    columns <- cl.ownedAttribute
  )
}
```

Example: ATL transformation (2/2)

SINTEF

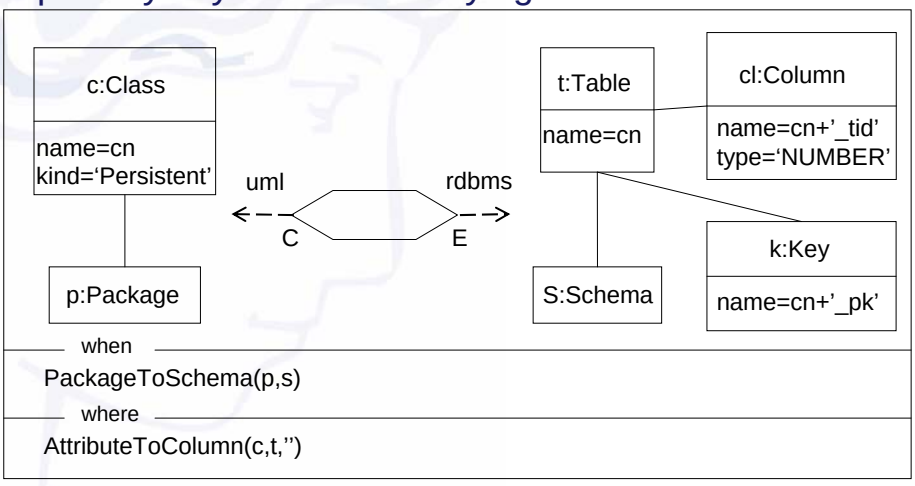
```
rule attr2col{
    from attr:uml2!Property
  to
      col:rdbms!Column(
    name <- attr.name,
    type <-attr.type.name
  )
}
```

- Compared to Java ATL provides simpler mechanisms for model traversal

Example: QVT graphical notation

SINTEF

A persistent class maps to a table,
a primary key and an identifying column



Example: QVT textual notation

SINTEF

A persistent class maps to a table,
a primary key and an identifying column

```
top relation ClassToTable {
   cn, prefix: String;
   checkonly domain uml c:Class {namespace=p:Package {},
        kind='Persistent', name=cn};
   enforce domain rdbms t:Table {schema=s:Schema {},name=cn,
        column=cl:Column {name=cn+'_tid', type='NUMBER'},
        key=k:Key {name=cn+'_pk', column=cl}};
   when {
        PackageToSchema(p, s);
   } where {
        prefix = '';
        AttributeToColumn(c, t, prefix);
   }
}
```


SINTEF

Atlas Transformation Language (ATL) tutorial with RSM example

ATL (1/2)

SINTEF

- ATL: Atlas Transformation Language
- Developed by:
  - LINA (Laboratoire D'Informatique De Nantes Atlantique)
  - Université de Nantes Faculté des Sciences et Techniques
  - Building IRIN
  - 2, rue de la Houssinière, BP 92208
  - 44322 Nantes cedex 3, France
- Proposal to the OMG MOF/QVT RFP.
- Model transformation language
  - Hybrid
    - Declarative => transformations based on simple mappings
    - Imperative => for complex mappings
  - Based on OCL
- Documentation
  - http://www.eclipse.org/gmt/atl/doc/

ATL (2/2)

SINTEF

- An ATL transformation program is composed of rules that define how
  - source model elements are matched and navigated
  - to create and initialize the elements of the target models
- The ATL programs for model to model transformation are called modules.
- Composed of
  - Header
    - transformation name
    - variables of source and target models
  - Import
    - libraries to be imported
  - Helpers
    - define (global) variables and functions.
  - Rules
    - describe the transformation from a source model to a target model

Warning

SINTEF

- To navigate from an element to its attribute,
  - write the name of the element, then “.” and the name of the attribute;
- If an identifier (variable name, metamodel name, model element name, etc.) is in conflict with a keyword,
  - it has to be marked with apostrophes;
- The ATL parser is case sensitive. This concerns
  - the file names
  - the source code itself.

SINTEF

ATL: Installation

Installation (1/2)

SINTEF

- Rational Software Modeler (RSM) 6.0
  - Based on Eclipse 3.0
- ADT (ATL Development Tools)
  - Installation guide
    - http://www.eclipse.org/gmt/atl/doc/ATL_Installation_Guide[v0.1].pdf
  - NB! Download ADT for Eclipse **3.0** and not 3.1
    - http://www.eclipse.org/downloads/download.php?file=/technology/gmt/atl/binaries/ADT-20060113/adt-20060113(Eclipse_3.0).zip
    - Unzip **`adt-20060113(Eclipse_3.0).zip`** into **`{RSMInstallDir}\eclipse`**
      - **`C:\Program Files\IBM\Rational\SDP\6.0\eclipse`**

Installation (2/2)

SINTEF

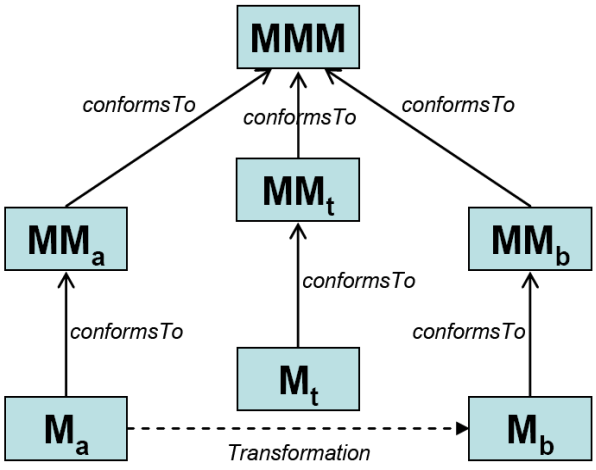
- Additional elements to download and install for ADT
  - External libraries not included as part of the standard ADT package
- Download **`antlr-2.7.5.jar`** from
  - http://www.antlr.org/download.html
  - Rename file to **`antlr.jar`**
  - Copy **`antlr.jar`** into … **`{RSMInstallDir}\eclipse\plugins\org.atl.eclipse.engine_1.0.7\lib`**
- Download **`mdr-standalone.zip`** from
  - http://mdr.netbeans.org/download/
  - Unzip into **`{RSMInstallDir}\eclipse\plugins\org.atl.engine.repositories.mdr4atl_1.0.0\lib`**
- The ADT is now installed. In Eclipse it adds
  - Plug-in
  - 2 perspectives
    - ATL perspective
    - Debug perspective

SINTEF

ATL: Book2Publish example

Reminder on transformation

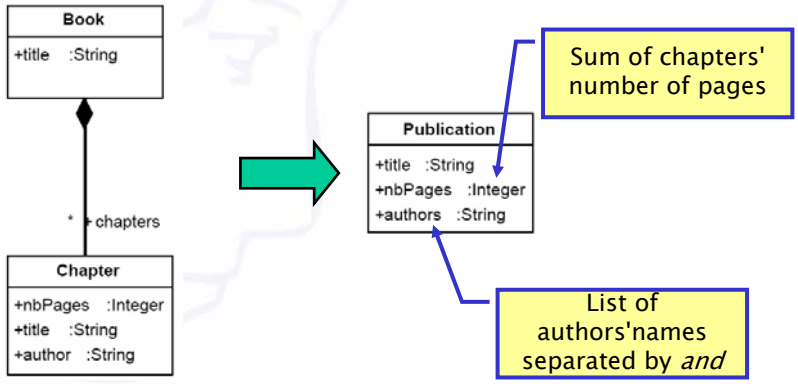
SINTEF



Goal

SINTEF

- Transform instances Book into instances of Publication



ATL transformation code

SINTEF

- Header
- Helper
- Rule

Book2Publication.atl

SINTEF

- Header

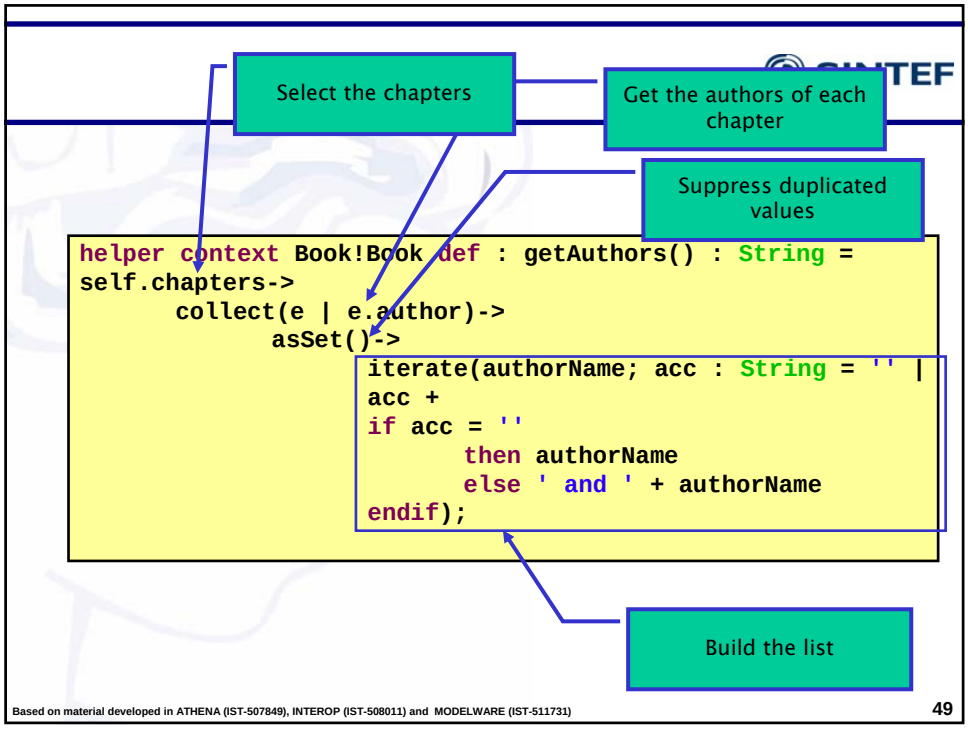
```
module Book2Publication;
create OUT : Publication from IN : Book;
```


1st helper

SINTEF

- Helper
  - To build the authors' list

```
helper context Book!Book def : getAuthors() : String =
self.chapters->
      collect(e | e.author)->
            asSet()->
                  iterate(authorName; acc : String = '' |
                  acc +
                  if acc = ''
                        then authorName
                        else ' and ' + authorName
                  endif);
```



Iterate on a collection

- To iterate on a collection

```
collection->iterate(elem : Type; acc : Type = <expression> |
                    expression-avec-elem-et-acc)
```

- `acc` is an accumulator which gets the initial value
- `elem` is an iterator which iterates on each element of the collection
- For each iteration `expression-avec-elem-et-acc` is
  - Evaluated
  - And then put into acc

2nd helper

SINTEF

- Helper
  - To get the total of pages

```
helper context Book!Book def : getNbPages() : Integer =
self.chapters->
      collect(f|f.nbPages)->
            iterate(pages; acc : Integer = 0 |
                  acc + pages);
```


Rules

SINTEF

```
rule Book2Publication {
from
      b : Book!Book (
            b.getNbPages() > 2
                  )
to
      out : Publication!Publication (
            title   <- b.title,
            authors <- b.getAuthors(),
            nbPages <- b.getNbPages()
            )
}
```

SINTEF

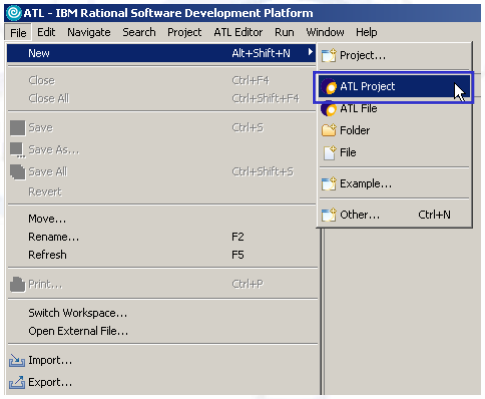
ATL: RSM

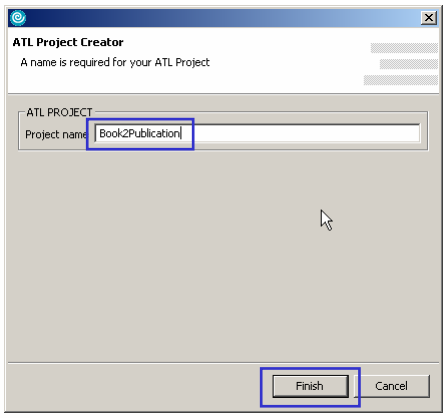
RSM example steps

SINTEF

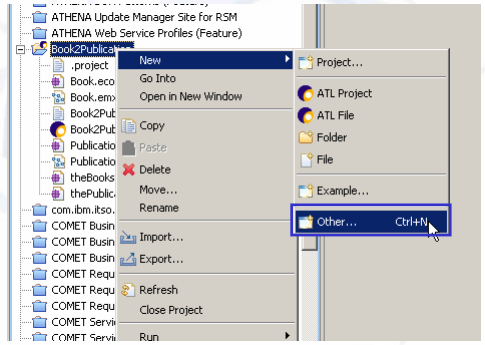
1. Create an ATL project **Book2Publication**
2. Create the **`Book.emx`** metamodel (UML model)
3. Export and import the **`Book.emx`** metamodel as **`Book.ecore`**
4. Create the **`Publication.emx`** metamodel (UML model)
5. Export and import the **`Publication.emx`** metamodel as **`Publication.ecore`**
6. Create an ATL file **`Book2Publication.atl`**
7. Write the ATL transformation
8. Create a source model **`theBooks.ecore`** containing book instances
9. Configure the ATL transformation
10. Run the ATL transformation

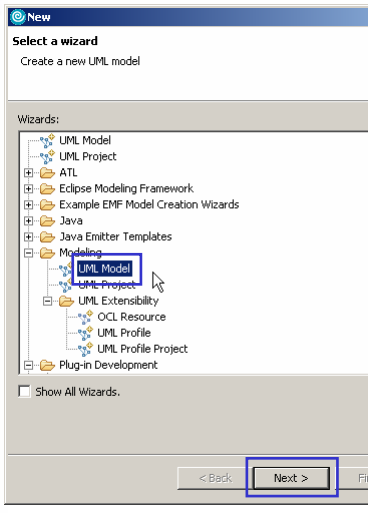
1. Create an ATL project



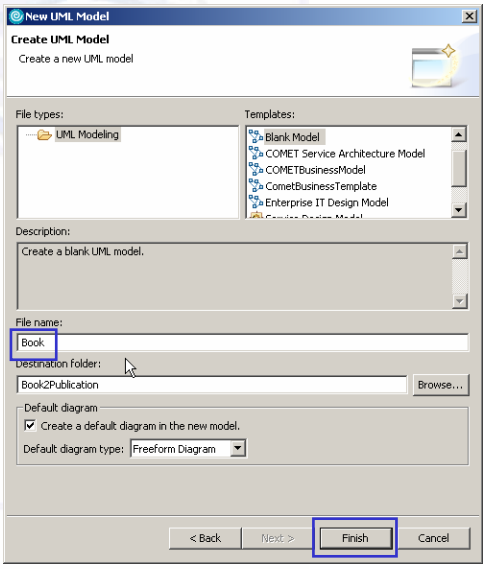


2a. Create the Book metamodel

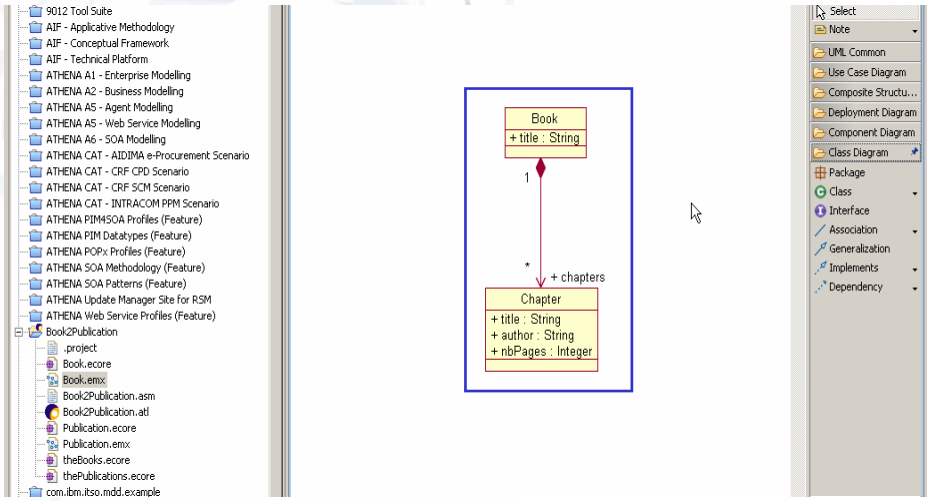




2b. Create the Book metamodel

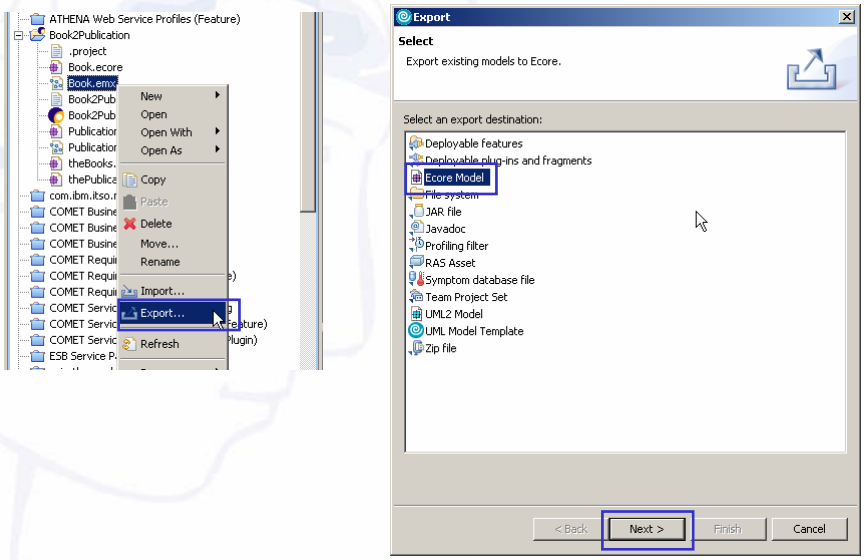


2c. Create the Book metamodel



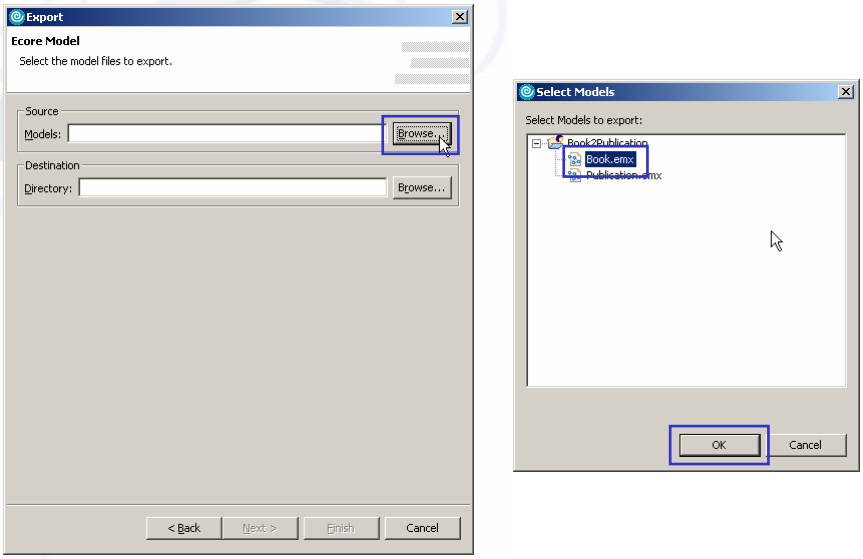
3a. Export the Book metamodel as .ecore

SINTEF

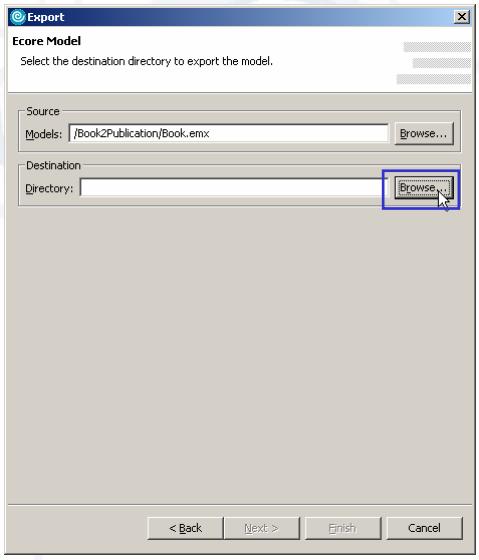


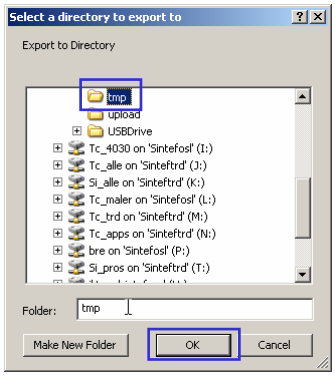
3b. Export the Book metamodel as .ecore

SINTEF



3c. Export the Book metamodel as .ecore

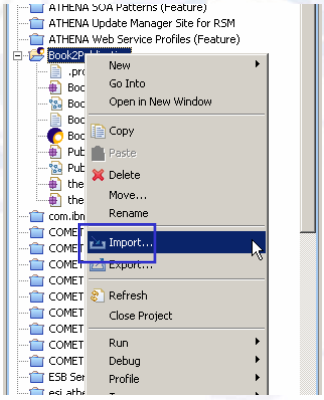


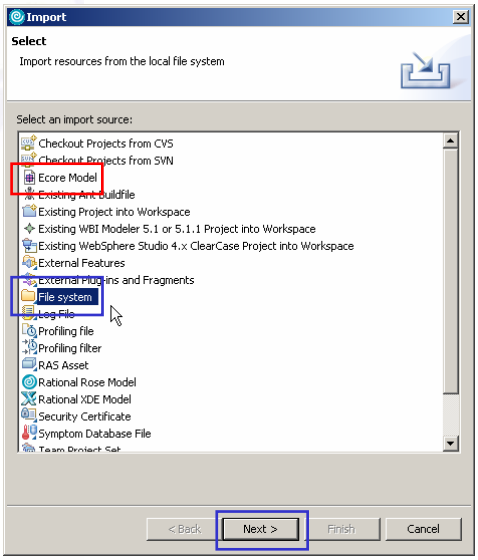


Book.ecore file

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
   xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="Book"
   nsURI="http:///Book.ecore" nsPrefix="Book">
  <eClassifiers xsi:type="ecore:EClass" name="Book">
   <eStructuralFeatures xsi:type="ecore:EAttribute" name="title" lowerBound="1"
eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
       defaultValueLiteral=""/>
   <eStructuralFeatures xsi:type="ecore:EReference" name="chapters" upperBound="-1"
       eType="#//Chapter" containment="true"/>
  </eClassifiers>
  <eClassifiers xsi:type="ecore:EClass" name="Chapter">
   <eStructuralFeatures xsi:type="ecore:EAttribute" name="title" lowerBound="1"
eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
       defaultValueLiteral=""/>
   <eStructuralFeatures xsi:type="ecore:EAttribute" name="author" lowerBound="1"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EString"
defaultValueLiteral=""/>
   <eStructuralFeatures xsi:type="ecore:EAttribute" name="nbPages" lowerBound="1"
       eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>
  </eClassifiers>
</ecore:EPackage>
```

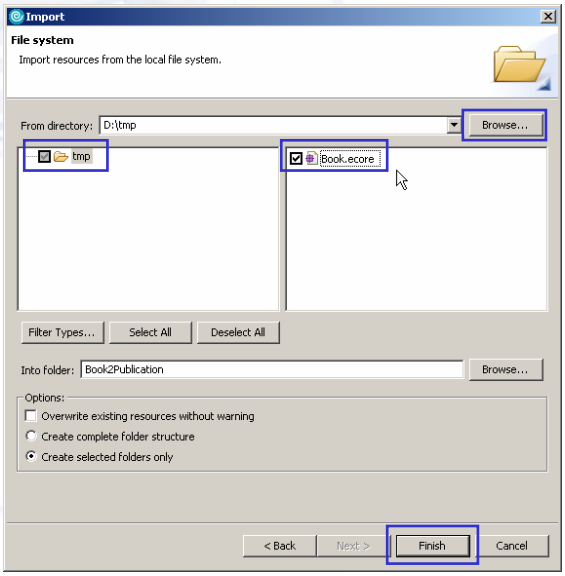
3d. Import the Book metamodel as .ecore





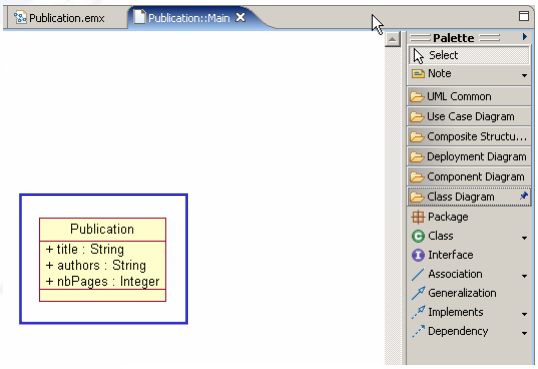
NB! Choose "File system"
If you choose "Ecore Model", RSM proposes to replace your ".emx' model !!!!!!

3e. Import the Book metamodel as .ecore

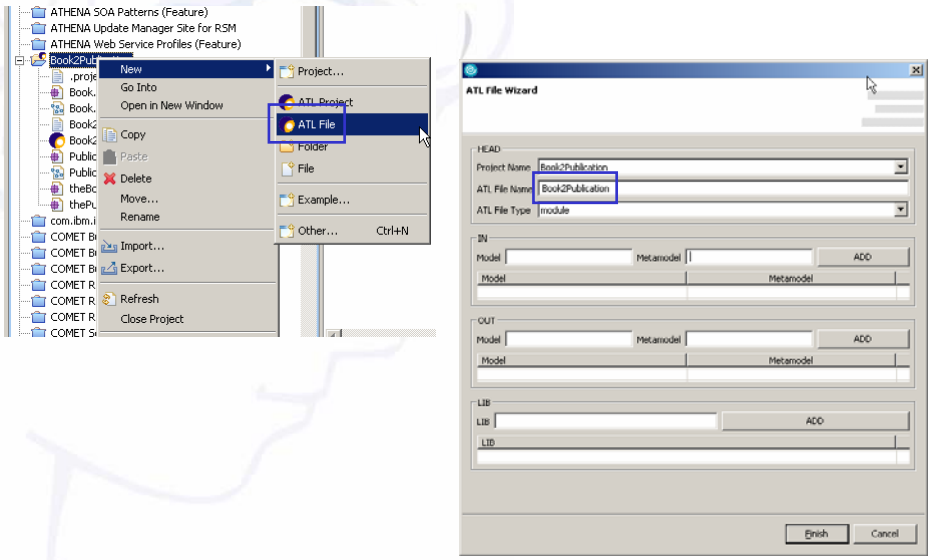


4-5. Create the Publication metamodel and export/import as .ecore

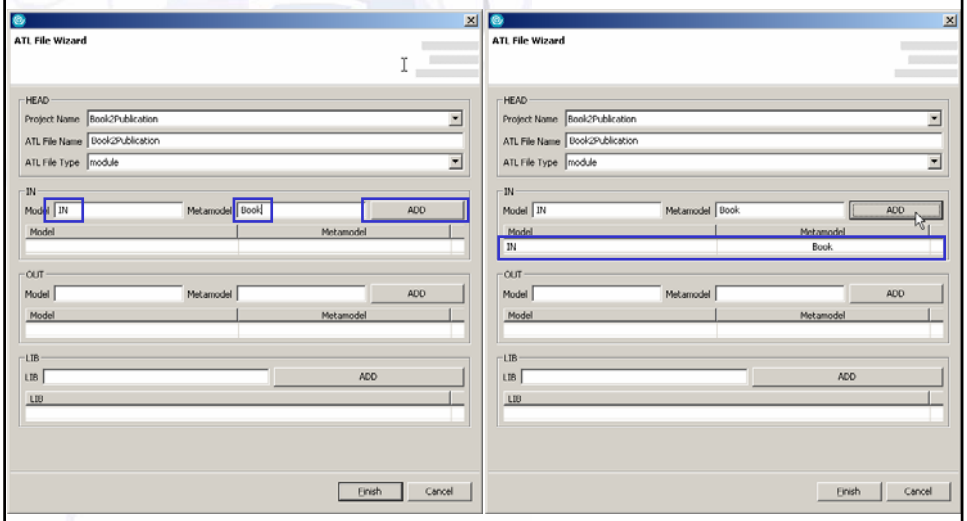
- Follow the same procedure as for the Book metamodel (steps 2 and 3) to create and export/import the **Publication** metamodel as .ecore



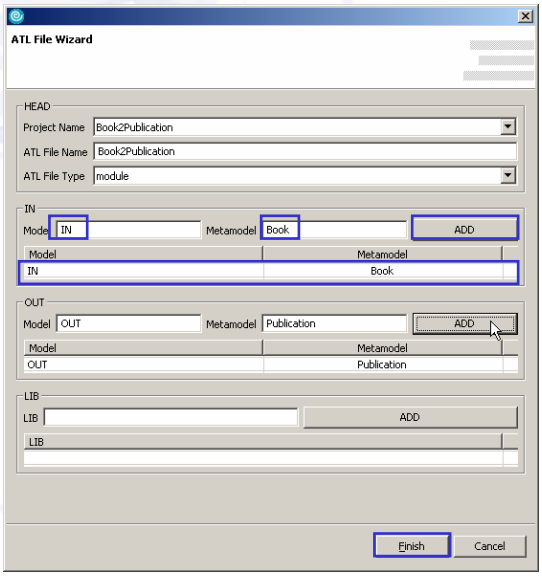
6a. Create an ATL file



6b. Set IN metamodel



6c. Set OUT metamodel



7. Write the ATL transformation

SINTEF

Book.emx | Book::Main | Publication.emx | Publication::Main | Book2Publication.atl

```
module Book2Publication; -- Module Template
create OUT : Publication from IN : Book ;

helper context Book!Book def : getAuthors() : String =
    self.chapters->collect(e | e.author)->
        asSet()->
            iterate(authorName; acc : String = '' |
                    acc +
                    if acc = ''
                        then authorName
                        else ' and ' + authorName
                    endif);

helper context Book!Book def : getNbPages() : Integer =
    self.chapters->collect(f | f.nbPages)->
        iterate(pages; acc : Integer = 0 |
                acc + pages);

rule Book2Publication {
from
    b : Book!Book (
        b.getNbPages() > 2
    )
to
    out : Publication!Publication (
        title <- b.title,
        authors <- b.getAuthors(),
    nbPages <- b.getNbPages()
    )
}
```

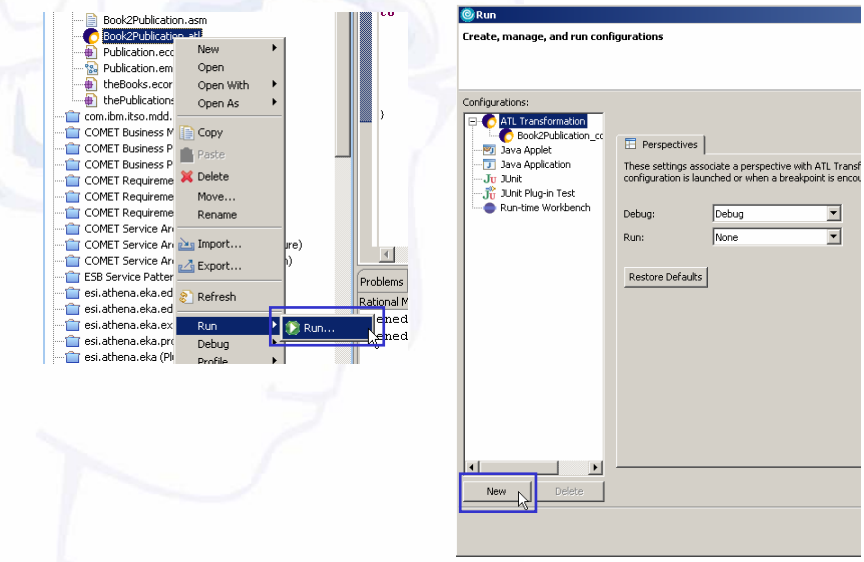

8. Create source model

SINTEF

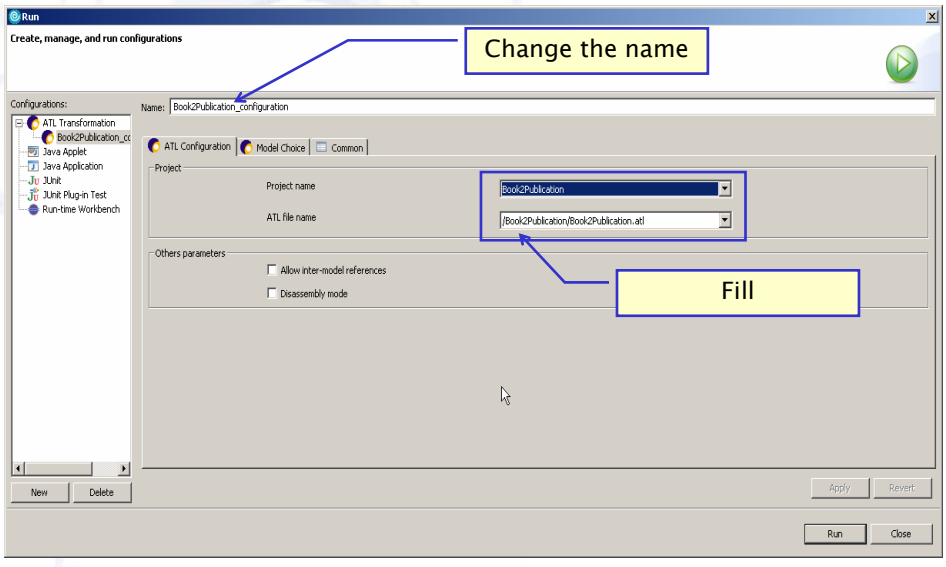
- Set of book instances **theBooks.ecore**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0"
        xmlns:xmi="http://www.omg.org/XMI"
        xmlns:Book="http:///Book.ecore">
<Book:Book title="Easy ATL">
  <chapters title="chapter 1" nbPages="5" author="Jean-Pierre"/>
  <chapters title="chapter 2" nbPages="6" author="Reyes"/>
</Book:Book >
<Book:Book title="ATL for Dummies">
  <chapters title="chapter 1: A" nbPages="13" author="Reyes"/>
  <chapters title="chapter 2: T" nbPages="17" author="Arne"/>
  <chapters title="chapter 3: L" nbPages="20" author="Jean-Pierre"/>
</Book:Book >
</xmi:XMI>
```

9a. Configure the ATL transformation

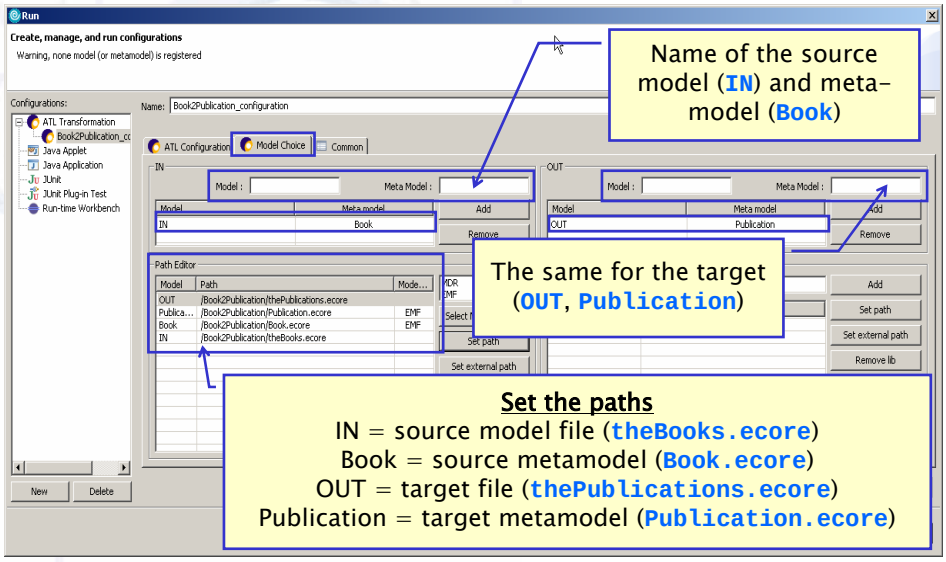


9b. Configure the ATL transformation



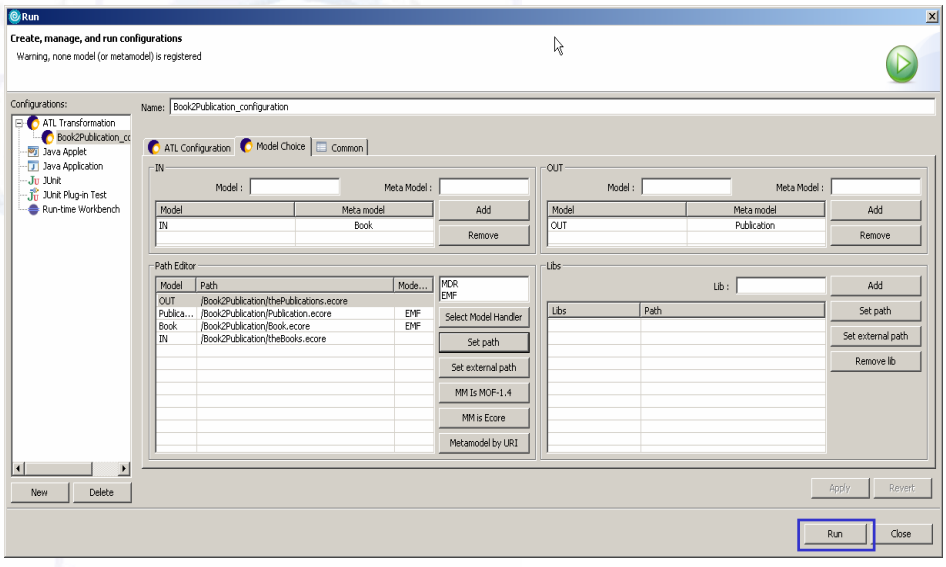
9c. Configure the ATL transformation

SINTEF



10. Run the ATL transformation

SINTEF



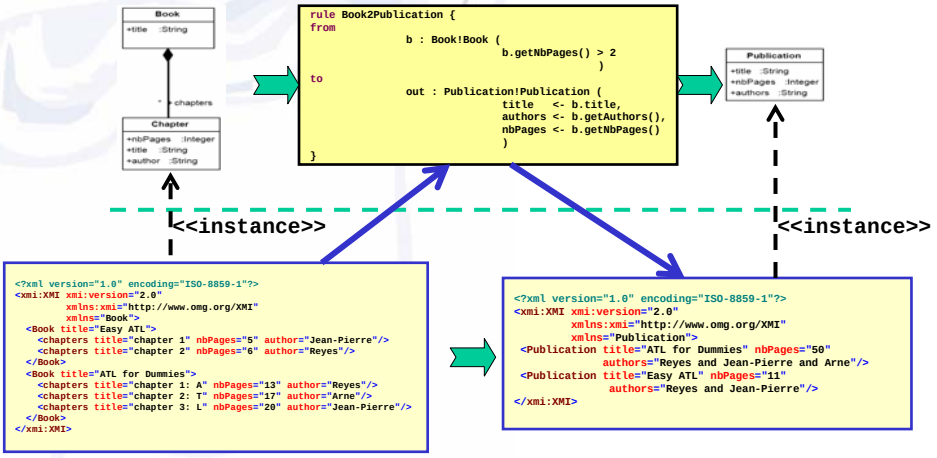
Result: thePublications.ecore

SINTEF

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
  xmlns:Publication="http:///Publication.ecore">
 <Publication:Publication title="ATL for Dummies"
                          authors="Reyes and Jean-Pierre and Arne"
                          nbPages="50"/>
  <Publication:Publication title="Easy ATL"
                          authors="Reyes and Jean-Pierre"
                           nbPages="11"/>
</xmi:XMI>
```


To summarize

SINTEF



References

References

- P. Swithinbank, M. Chessell, T. Gardner, C. Griffin, J. Man, H. Wylie, and L. Yusuf, "Patterns: Model-Driven Development Using IBM Rational Software Architect", IBM, Redbooks, December 2005. http://www.redbooks.ibm.com/redbooks/pdfs/sg247105.pdf
- OMG, "Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification", Object Management Group (OMG), Document ptc/05-11-01, November 2005. http://www.omg.org/docs/ptc/05-11-01.pdf
- INRIA, "ATL - The Atlas Transformation Language Home Page". http://www.sciences.univ-nantes.fr/lina/atl/
- Eclipse.org, "ATL Home page". http://www.eclipse.org/gmt/atl/
- IBM, "Model Transformation Framework". http://www.alphaworks.ibm.com/tech/mtf/
- IBM, "Model Transformation with the IBM Model Transformation Framework". http://www-128.ibm.com/developerworks/rational/library/05/503_sebas/